SmartPlant Schema Editor

User's Guide



Version 2016 (6.0)

March 2016

DSPF1-PE-200008Q

PART OF HEXAGON

INTERGRAPH®

Copyright

Copyright © 2002-2016 Intergraph® Corporation. All Rights Reserved. Intergraph is part of **Hexagon**.

Including software, file formats, and audiovisual displays; may be used pursuant to applicable software license agreement; contains confidential and proprietary information of Intergraph and/or third parties which is protected by copyright law, trade secret law, and international treaty, and may not be provided or otherwise made available without proper authorization from Intergraph Corporation.

Portions of the user interface copyright 2012-2014 Telerik AD.

U.S. Government Restricted Rights Legend

Use, duplication, or disclosure by the government is subject to restrictions as set forth below. For civilian agencies: This was developed at private expense and is "restricted computer software" submitted with restricted rights in accordance with subparagraphs (a) through (d) of the Commercial Computer Software - Restricted Rights clause at 52.227-19 of the Federal Acquisition Regulations ("FAR") and its successors, and is unpublished and all rights are reserved under the copyright laws of the United States. For units of the Department of Defense ("DoD"): This is "commercial computer software" as defined at DFARS 252.227-7014 and the rights of the Government are as specified at DFARS 227.7202-3.

Unpublished - rights reserved under the copyright laws of the United States.

Intergraph Corporation
305 Intergraph Way
Madison, AL 35758

Documentation

Documentation shall mean, whether in electronic or printed form, User's Guides, Installation Guides, Reference Guides, Administrator's Guides, Customization Guides, Programmer's Guides, Configuration Guides and Help Guides delivered with a particular software product.

Other Documentation

Other Documentation shall mean, whether in electronic or printed form and delivered with software or on Intergraph Smart Support, SharePoint, or box.net, any documentation related to work processes, workflows, and best practices that is provided by Intergraph as guidance for using a software product.

Terms of Use

a. Use of a software product and Documentation is subject to the End User License Agreement ("EULA") delivered with the software product unless the Licensee has a valid signed license for this software product with Intergraph Corporation. If the Licensee has a valid signed license for this software product with Intergraph Corporation, the valid signed license shall take precedence and govern the use of this software product and Documentation. Subject to the terms contained within the applicable license agreement, Intergraph Corporation gives Licensee permission to print a reasonable number of copies of the Documentation as defined in the applicable license agreement and delivered with the software product for Licensee's internal, non-commercial use. The Documentation may not be printed for resale or redistribution.

b. For use of Documentation or Other Documentation where end user does not receive a EULA or does not have a valid license agreement with Intergraph, Intergraph grants the Licensee a non-exclusive license to use the Documentation or Other Documentation for Licensee's internal non-commercial use. Intergraph Corporation gives Licensee permission to print a reasonable number of copies of Other Documentation for Licensee's internal, non-commercial use. The Other Documentation may not be printed for resale or redistribution. This license contained in this subsection b) may be terminated at any time and for any reason by Intergraph Corporation by giving written notice to Licensee.

Disclaimer of Warranties

Except for any express warranties as may be stated in the EULA or separate license or separate terms and conditions, Intergraph Corporation disclaims any and all express or implied warranties including, but not limited to the implied warranties of merchantability and fitness for a particular purpose and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of such disclaimer. Intergraph believes the information in this publication is accurate as of its publication date.

The information and the software discussed in this document are subject to change without notice and are subject to applicable technical product descriptions. Intergraph Corporation is not responsible for any error that may appear in this document.

The software, Documentation and Other Documentation discussed in this document are furnished under a license and may be used or copied only in accordance with the terms of this license. THE USER OF THE SOFTWARE IS EXPECTED TO MAKE THE FINAL EVALUATION AS TO THE USEFULNESS OF THE SOFTWARE IN HIS OWN ENVIRONMENT.

Intergraph is not responsible for the accuracy of delivered data including, but not limited to, catalog, reference and symbol data. Users should verify for themselves that the data is accurate and suitable for their project work.

Limitation of Damages

IN NO EVENT WILL INTERGRAPH CORPORATION BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL INCIDENTAL, SPECIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO, LOSS OF USE OR PRODUCTION, LOSS OF REVENUE OR PROFIT, LOSS OF DATA, OR CLAIMS OF THIRD PARTIES, EVEN IF INTERGRAPH CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

UNDER NO CIRCUMSTANCES SHALL INTERGRAPH CORPORATION'S LIABILITY EXCEED THE AMOUNT THAT INTERGRAPH CORPORATION HAS BEEN PAID BY LICENSEE UNDER THIS AGREEMENT AT THE TIME THE CLAIM IS MADE. EXCEPT WHERE PROHIBITED BY APPLICABLE LAW, NO CLAIM, REGARDLESS OF FORM, ARISING OUT OF OR IN CONNECTION WITH THE SUBJECT MATTER OF THIS DOCUMENT MAY BE BROUGHT BY LICENSEE MORE THAN TWO (2) YEARS AFTER THE EVENT GIVING RISE TO THE CAUSE OF ACTION HAS OCCURRED.

IF UNDER THE LAW RULED APPLICABLE ANY PART OF THIS SECTION IS INVALID, THEN INTERGRAPH LIMITS ITS LIABILITY TO THE MAXIMUM EXTENT ALLOWED BY SAID LAW.

Export Controls

Intergraph Corporation's software products and any third-party Software Products obtained from Intergraph Corporation, its subsidiaries, or distributors (including any Documentation, Other Documentation or technical data related to these products) are subject to the export control laws and regulations of the United States. Diversion contrary to U.S. law is prohibited. These Software Products, and the direct product thereof, must not be exported or re-exported, directly or indirectly (including via remote access) under the following circumstances:

a. To Cuba, Iran, North Korea, Sudan, or Syria, or any national of these countries.

b. To any person or entity listed on any U.S. government denial list, including but not limited to, the U.S. Department of Commerce Denied Persons, Entities, and Unverified Lists, http://www.bis.doc.gov/complianceandenforcement/liststocheck.htm, the U.S. Department of Treasury Specially Designated Nationals List, http://www.treas.gov/offices/enforcement/ofac/, and the U.S. Department of State Debarred List, http://www.pmddtc.state.gov/compliance/debar.html.

c. To any entity when Licensee knows, or has reason to know, the end use of the Software Product is related to the design, development, production, or use of missiles, chemical, biological, or nuclear weapons, or other un-safeguarded or sensitive nuclear uses.

d. To any entity when Licensee knows, or has reason to know, that an illegal reshipment will take place.

e. Any questions regarding export or re-export of these Software Products should be addressed to Intergraph Corporation's Export Compliance Department, Huntsville, Alabama 35894, USA.

Trademarks

Intergraph, the Intergraph logo, Intergraph Smart, PDS, SmartPlant, SmartMarine, FrameWorks, I-Sketch, IntelliShip, ISOGEN, SmartSketch, SPOOLGEN, SupportManager, SupportModeler, Sapphire, and FreeView are trademarks or registered trademarks of Intergraph Corporation or its subsidiaries in the United States and other countries. Microsoft and Windows are registered trademarks of Microsoft Corporation. MicroStation is a registered trademark of Bentley Systems, Inc. Other brands and product names are trademarks of their respective owners.

Contents

Preface **9**

SmartPlant Foundation Product Documentation 9

What's New in SmartPlant® Schema Editor? **12**

SmartPlant Schema Editor Overview **17**

SmartPlant Schema Overview **18**

Class Definitions 19
Model Definitions 20
Shared Object Definitions 21
Interface Definitions 22
Property Definitions 26
Enumerated Lists 27
Units of Measurement 31
Relationship Definitions 31
Edge Definitions, Graph Definitions, and View Definitions 33
Edge Definitions 33
Graph Definitions 37
View Definitions 39
Meta Schema 41
Component Schemas 42
XML Files 43
UML Notation 45

Forward and Backward Schema Compatibility **47**

Schema Editor User Interface **48**

Standard Workflows Window 49
Schema Editor Toolbars 53
Schema Editor - Standard Overview 55
Open a Startup File 56
Open a Schema 56
Open Other Schema Files 56
Set the Active Schema File 56
Open a Tool Schema 56
Synchronize Another Tool Map Schema with Tool Metadata 57
Search for Schema Objects 58
Search for Tool Schema Objects 59
Define Mapping 59
Other Workflows Dialog Boxes 60
Personalized Menus 63
Schema Editor Roles 64
Create a Personalized Menu Configuration 65

Open a Personalized Menu Configuration File 67
Define the Storage Location for Your Personalized Menu Configurations 67
Modify an Existing Personalized Menu Configuration 68
Layouts 69
Create a Layout 69
Schema Editor Naming Rules 71
Define Naming Rules for Objects 78

Opening and Using Files in the Schema Editor 82

Configuration Managed Files (CMFs) 82
Viewable Configurations 84
Editable Configurations 84
Multiple Editable Configurations 84
Single Active Configuration with New Objects in Multiple Configurations 84
Pseudo Files 85
Special Configuration Commands 86
Allow Multiple Users to Edit the Same CMF File 88
Session Files 92
Open a Session File 92
Create a Session File 92
File Configurations 93
Create a File Configuration 93
Open a File Configuration 94
Modify a File Configuration 94
Set Dependencies for a Schema File 95
Opening Files 96
Open a Schema 97
Set the Active Schema File 98
Open a Tool Map Schema 98
Set the Active Tool Schema File 99
Open a Data File 99
Open a Data File Set 99
Compare a Data File Set 100

Navigating Files and Data in the Schema Editor 101

Tree View: An Example 103
Tree/Table View: An Example 105
Tree/Viewable UML View: An Example 107
Tree/Editable UML View: An Example 108
Create a UML Diagram 111
Align Objects in Editable UML Views 118
Remove Objects from Editable UML Views 118
Clear Editable UML Views 118
Open Saved UML Views 119
Auto-Generate UML Diagrams Based on Published Data 120
Tree/MultiTab View: An Example 122
View Options 123
Set View Options 123

Finding Objects in the Schema Editor 124

Find Objects Using Search Criteria 124
Find Objects by Class 125
Find Objects Using Simple Criteria 128
Keyword Search 129
Find Objects Using Predefined Named Query 133
Find Objects by Interface 135
Simple Interface-Based Query 135
Full Interface-Based Query 137
Filtered Interface-Based Query 140

SmartPlant Schema Documentation 144

View SmartPlant Schema Documentation 145

Extending the SmartPlant Schema 146

Naming Conventions in the SmartPlant Schema 146
Patterns and Abstract Roles 147
SmartPlant Schema Modification Rules 156
Schema modification rules 158
Create New Schema Objects 168
Create a Class Definition 169
Create a Component Schema 170
Create an Interface Definition 171
Create a Property Definition 172
Create an Enumerated List 173
Create a Unit of Measure List 175
Create a Relationship Definition 176
Create an Edge Definition 177
Create a Graph Definition 180
Create a View Definition 182
Create a Relationship Between Schema Objects 186
Modifying Objects and Relationships in the Schema 187
Modify Existing View and Graph Definitions 187
Extending the Schema for SmartPlant Integration 191
Open the CMF File for Editing 192
Create new objects and relationships in the schema 198
Check in the updated CMF file 198
Extract the schema file to load 198
Load schema changes into the database 199
Regenerate the component schema files 203
Import Existing Customizations in Extensions.xml into the CMF File 203
Move Customizations from Project to Project 205
Extend the Existing Stand-Alone SmartPlant Foundation Schema 206
Create a schema extension file 207
Add a new schema file to a configuration file 208
Create new objects and relationships in the software 209
Load stand-alone schema changes into SmartPlant Foundation 209
Creating an alternate plant breakdown structure (PBS) 210
Remove the existing PBS structure 212

Check out and launch the CMF file 213
Define a custom PBS hierarchy from a template 214
Define a custom PBS hierarchy 215
Check in the CMF and load schema changes 218
Generate component schema files 220
Generate authoring schema and load files 221
Load alternate PBS objects 224
Create a new plant and structure 225
Associate a vault with the new plant 226
Register and publish the PBS 226

Using Excel to Export and Import Schema 227

Export the Schema to Excel 228
Export Schema Containers 228
Export Pseudo File Configurations 228
Export a Displayed View 229
Edit in Excel 231
Excel (.xlsm) File 231
UIDs File 236
Import Data into Schema Editor 237
Import from Excel 237

Publish and Retrieve File Types for Tools 239

Externalize Authoring Tool Metadata 240
Class Definitions (ClassDef) 241
Component Schemas (CompSchema) 241
DocumentType Class Definition 242
EnumEnum Class Definition 242
Enumerated Lists (EnumListType) 242
SPMapClassDef 244
Tool Class Definition 244
ToolSchema Class Definition 244
InterfaceDef Instances 244
MenuCmd Instances 246
Menu Instances 248
RelDef Instances 248
Componentization Relationship Definition 248
Create a New Document Type that Existing Tools Can Publish and Retrieve 251
Update Document Types that Existing Tools Can Publish and Retrieve 255
Generate Authoring Tool Metadata Files 257

SmartPlant Mapping Overview 258

Tool Map Schemas 259
Tool Schema Models 260
Tool Map Schema Objects 261
Mapping Relationship Definitions 266
Cardinalities for Mapping 267
Types of Mapping for Map Class Definitions 271
Using View Definitions to Define Mapping 275

Mapping Directly Against the SmartPlant Schema 277
Map Enumerated List and Map Unit of Measure List Mapping 283
Defining Authoring Tool Mapping 289
Mapping with Metadata Adapters 291
Viewing Authoring Tool Mapping 307
Work Processes for Mapping for Tools with Metadata Adapters 327
Map Publish and Retrieve Properties in the Map Environment 391
Modify Reference Files with the Integrated Reference File Environment 408
Map Publish and Retrieve Properties in a Graphical View 414
Work Process for Mapping for Tools without Metadata Adapters 427

Comparison 463

Compare Files 464
Full Comparison: An Example 465
Detect Tombstones 468
Detect New UIDs for Named Objects 469
Tombstone Comparison Results: An Example 469
Selectively Update 471

Validate Schemas 473

Validate Files 473
Automatic Validation 476
Set Validation Message Filtering 476

Verification of Tool Map Schema and Data Files 478

Verify Publish Tool Map Schemas 479
Verify Publish Data Files 481

Glossary 482

Index 487

Preface

This document is a user's guide for the SmartPlant Schema Editor. This user's guide contains the most commonly used procedures and examples for using the Schema Editor, as well as overviews and command reference information.

This document is intended for advanced users or system administrators who need to understand the SmartPlant schema, tool map schemas, meta data, and data files for integration and for SmartPlant Foundation modeling.

SmartPlant Foundation Product Documentation

SmartPlant Foundation documentation is available as Help and as PDF files. To view printable guides for SmartPlant Foundation, click **Help > SmartPlant Foundation Printable Guides** in the software.

Intergraph gives its customers permission to print as many copies of the delivered PDF files as they need for their non-commercial use. Do not print the PDF files for resale or redistribution.

Integration Documentation

Installation and Overviews

- *SmartPlant Enterprise Hardware and Software Recommendations* - Provides information about the hardware recommendations and software requirements of the authoring tools that are part of the SmartPlant Enterprise suite.
- *Integration Setup Guide* - Provides information about setting up and using SmartPlant Enterprise products in an integrated environment. Also provides integration troubleshooting information.
- *SmartPlant Enterprise Overview* - Provides an overview of SmartPlant Enterprise and integration concepts and data exchange among integrated SmartPlant Enterprise authoring tools.
- *SmartPlant Enterprise Data Exchange Diagrams* - Includes full-size graphics that describe the exchange of data among SmartPlant Enterprise authoring tools. These graphics are also included in the *SmartPlant Enterprise Overview.*

User and Administrative Guides

- *Integration User's Guide* - Provides information about using SmartPlant Enterprise products in an integrated environment. Also provides a basic understanding of integration and common integration tasks.
- *Integration Administrator's Guide* - Provides information for system administrators on how to set up and configure an integrated environment.
- *SmartPlant Enterprise Citrix Configuration Guide* - Provides information on configuring integrated SmartPlant® Enterprise applications to run in a Citrix environment.
- *Schema Editor User's Guide* - Provides instructions for using the Schema Editor to view and extend the SmartPlant schema, view and modify authoring tool mapping, and compare schema files.

Configuration Guides

- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant Electrical* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant Electrical.
- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant Instrumentation* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant Instrumentation.
- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant P&ID* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant P&ID.
- *SmartPlant Enterprise SmartPlant Adapter for Intergraph Smart™ 3D* - Describes the SmartPlant Adapter details, functionality and configuration in Smart 3D.
- *SmartPlant Enterprise SmartPlant Adapter for PDS 3D* - Describes the SmartPlant Adapter details, functionality, and configuration in PDS 3D.

Schema and Mapping Guides

- *Integration: A Gentle Introduction to the SmartPlant Schema* - Provides an introduction to the SmartPlant schema and the data model that describes how the SmartPlant schema is constructed.
- *Integration: A Gentle Introduction to SmartPlant Schema Patterns, Process & Practice* - Provides an introduction to patterns in the SmartPlant schema.
- *SmartPlant Mapping User's Guide* - Provides guidelines, procedures, and examples of mapping between the SmartPlant schema and integrated authoring tools.
- *Schema Editor User's Guide* - Provides instructions for using the Schema Editor to view and extend the SmartPlant schema, view and modify authoring tool mapping, and compare schema files.

Intergraph Work Process Guides

Intergraph Work Process guides map the process, power, and marine industries' basic work processes to SmartPlant Enterprise and SmartPlant Marine Enterprise solutions. This documentation helps your organization transition from previous work practices to using Intergraph tools as the new way of executing projects. These documents are available on *https://smartsupport.intergraph.com* (*https://smartsupport.intergraph.com*) under **View Documentation > Work Process Guides**.

Enterprise Work Processes describe an entire engineering discipline or process at two different levels, including:

- Providing a swim lane diagram to define the typical roles and high-level processes involved in a project
- Showing how Intergraph tools are used to produce required deliverables

Integration Capability Statements describe a specific work process among a set of Intergraph tools, including:

- Explaining the out-of-the-box capabilities of the tools and their recommended use
- Providing a swim lane diagram to show how the tools interact in the work process
- Stating critical requirements and precautions
- Enterprise Work Processes and Integration Capability Statements offer a better understanding of how Intergraph tools work together and how to adapt the tools to improve existing work processes.

Tool Integration Documentation

Intergraph Smart™ 3D Documentation

- *How to Integrate Smart 3D Material Take Off & Nesting* - Describes the material take off integration between Smart 3D and SmartPlant Materials.
- *Smart 3D Integration Reference Guide* - Provides information about installing, configuring, and using Smart 3D in an integrated environment.

Integration Programming Documentation

- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant Electrical* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant Electrical.
- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant Instrumentation* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant Instrumentation.
- *SmartPlant Enterprise SmartPlant Adapter for SmartPlant P&ID* - Describes the SmartPlant Adapter details, functionality and configuration in SmartPlant P&ID.
- *SmartPlant Enterprise SmartPlant Adapter for Intergraph Smart™ 3D* - Describes the SmartPlant Adapter details, functionality and configuration in Smart 3D.
- *SmartPlant Enterprise SmartPlant Adapter for PDS 3D* - Describes the SmartPlant Adapter details, functionality, and configuration in PDS 3D.
- *SmartPlant Adapter Interfaces Reference Guide* - Provides information for tool developers about the interface classes used to communicate data between the authoring tool and the common user interface and the common user interface and the tool adapter.
- *SmartPlant Common UI Interfaces Reference Guide* - Provides information for tool developers about the API exposed through the EFCommonUI.dll, which is used by authoring tool developers to access integration functionality.
- *SmartPlant Metadata Adapter Reference Guide* - Provides information for tool developers about programming the metadata adapter. The tool metadata adapter allows communication between the SmartPlant Schema Editor and the tool database to manage mapping between the tool and the SmartPlant schema.
- *SmartPlant Schema Component API Reference Guide* - Provides information for developers about the Schema Component API.

Customer Support

For the latest support information for this product, use a World Wide Web browser to connect to *http://support.intergraph.com* (*http://support.intergraph.com*). Also, you can submit any documentation comments or suggestions you might have on the Intergraph support site.

What's New in SmartPlant® Schema Editor?

NOTE If a SmartPlant Foundation release is not listed below, that release has fixes only and no new features for this particular application. For a complete list of the new features in a release, refer to the *SmartPlant Foundation Core Functionality & Enterprise Integration Capabilities Release Bulletin*. For more information about fixes included in a release, see the Readme file delivered with the SmartPlant Foundation setup.

Version 2014 R3 (5.3)

- Information on schema evolution rules has been added. For more information, see *SmartPlant Schema Modification Rules* (on page 156).
- The auto-create map enumerated list feature now handles hierarchical enumerated lists for authoring tools. For more information, see *Create Mapping for Enumerating Lists in Map Environment* (on page 395). (CR-AM-73338)

Version 2014

- You can now add a display name in several dialog boxes in the tool map schema. For more information, see *Work Processes for Mapping for Tools with Metadata Adapters* (on page 327).
- Added relevant information from the retired *SmartPlant Modeling Guide*. For more information, see *Patterns and Abstract Roles* (on page 147).

Version 2009 R4 SP1

- Typographical corrections were made to this document.

Version 2009 R4

- Work processes for managing enumerated lists, such as code lists, have been improved. The Integrated Map Environment allows you to map, view, and manage objects that are mapped into (publish) and out of (retrieve) the SmartPlant schema from tool map schemas. The enumerated lists are automatically updated in the tool data dictionary and mapped to the SmartPlant schema. For more information, see *Map Publish and Retrieve Properties in the Map Environment* (on page 391).
- The Integrated Reference File Environment allows you to view and modify data in a set of schema and tool schema reference files. You can copy or replace data from a reference file into an active target file. This functionality allows you to quickly start a new project by referencing existing data and selectively copying it to the new project. For more information, see *Modify Reference Files with the Integrated Reference File Environment* (on page 408).
- You can now create or modify an alternate PBS hierarchy template. This functionality allows you to quickly start a project by using templates to create plant breakdown structures. Seven templates are included, and custom templates can be created in the Schema Editor. For more information, see *Creating an alternate plant breakdown structure (PBS)* (on page 210).

- A new command, **Construct schema views for data objects**, has been added to the **Editable UML** view. This command examines all the objects in the data file and generates a UML diagram for each class definition for which there is at least one instance in the data file. System implementors can generate diagrams of customized schema automatically. For more information, see *Auto-Generate UML Diagrams Based on Published Data* (on page 120).
- You can now disable and enable existing mapping relationships in the schema without deleting existing relationships. This means implementors can alter data flows without deleting and recreating established mappings. For more information, see *Disable or Enable an Existing Property Relationship in the Schema* (on page 361).
- You can now edit view definitions and graph definitions from the **Edit Map Class Definition** dialog box. For more information, see *Edge Definitions, Graph Definitions, and View Definitions* (on page 33).
- You can now open a second data file set to compare the differences and similarities, and save the comparison file separately. For more information, see *Compare a Data File Set* (on page 100).
- UML diagrams for published schema for SmartPlant P&ID were added. For more information, see *UML Diagrams for Published Schema for SmartPlant P&ID*.

Version 2009 R3 SP3

- Typographical corrections were made to this document.

Version 2009 R3 SP2

- You can automatically correlate an enumerated list from a tool with an enumerated list in the schema. See *Automatically Correlate a Tool Enumerated List with a Schema Enumerated List* (on page 359).
- Schema Editor includes a new feature that automatically maps multiple enumerations existing in an authoring tool to the SmartPlant Schema. In the previous release, automatic mapping was possible only with an enumeration. This feature benefits the administrator by expediting the schema mapping process.
- You can export schema data from Schema Editor to Microsoft Excel to make changes. You can import the edited schema from Excel to Schema Editor. The following data can be exported to Excel: any schema displayed in a tabular view, Pseudo file configuration (CMF), and Schema container (XML file or CMF active configuration). See *Using Excel to Export and Import Schema* (on page 227) in the *SmartPlant Schema Editor's Guide*.
- Schema Editor includes a new feature that supports filtering by a component schema only that set of properties are available for mapping. Filtering by component schema narrows the set of properties available to the user to only those properties that are used within the particular component schema.
- A number of new features have been added to the Schema Editor to facilitate faster loading of the schema into SmartPlant Foundation:
  - Added ContainerID as a property for pseudo-file configurations and modified the edit form for configurations to support the definition of this property.
  - Added the **Tools > Schema > Generate Component Schema** command and a new API to generate a single component schema (as opposed to all component schemas).

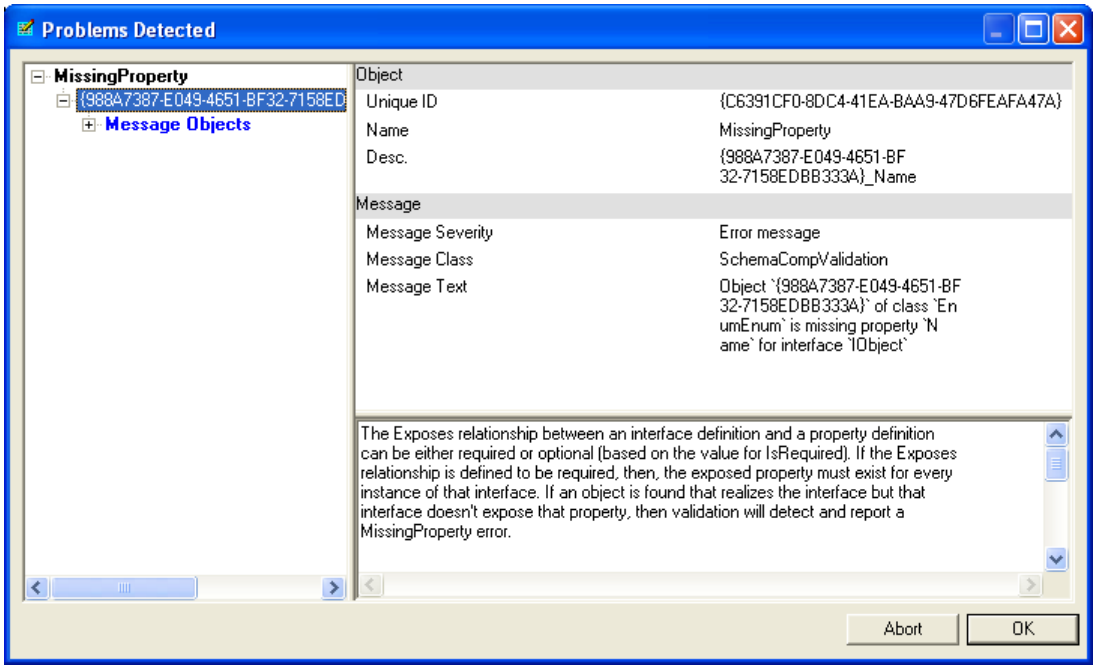
- Added SPFMeta attribute to Container element and added support for automatic loading of SmartPlant Foundation metadata inside Schema Editor based on this attribute value.

Version 2009

- The Schema Editor displays validation errors and messages in a new user-friendly interface to help you more easily diagnose problems.
- The **Selectively Update** command allows you to compare schemas, tool map schemas, and data files, then selectively choose which objects to add, update, or delete. Schema Editor also provides the ability to compare either the CMF files or XML files between two different configurations.
- A new **Layout** menu allows you to build your own Schema Editor user interface by opening multiple intelligent controls that update automatically when you select an object on one of the forms.
- The **Auto-create** command can be used to automatically add new properties to view definitions. This feature only allows you to add new properties to a view definition and does not update existing properties on view definitions.
- The **New SmartPlant Enumerated List with Correlated Entries** command establishes publish and retrieve relationships or only a publish relationship for the SmartPlant enumerated lists. A message appears allowing you to click **Yes** to establish both publish and retrieve relationships or click **No** to establish only a publish relationship.

Version 2008 SP 1

- The Schema Editor provides usability improvements through an updated user interface that uses tabs and toolbars to help you navigate in the software. You can access this new user interface by clicking **Workflows > Standard Workflows** in the Schema Editor or by launching the Schema Editor from the following location on the **Start** menu: **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor - Standard**. The previous Schema Editor user interface, featuring the Overall Workflows dialog box, is still available by clicking **Workflows > Overall Workflows** or by clicking **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor - Developer**. For more information, see *Standard Workflows Window* (on page 49), *Schema Editor Toolbars* (on page 53), and *Schema Editor - Standard Overview* (on page 55).
- You can use the new **Edit > Turn AutoValidation On** command to perform real-time validation on any changes that you make to the schema or tool map schemas when you end a transaction (complete a command or action) in the Schema Editor. The **Problems Detected** dialog box displays any validation errors with detailed explanations of the errors and their solutions.



For more information about the validation rules the Schema Editor uses to validate files, see the *Validation* section in the *SmartPlant Schema Component API Reference Guide*.

- Every time you make a change to the contents of any CMF, XML, or CFG file in the Schema Editor, the Schema Editor sets a validation flag to false. The next time you save the file, the Schema Editor prompts you to validate the file. Once you have validated the file and no errors have been found, the Schema Editor sets the validation flag to true. For CMF files, all configurations must be validated with no errors before the entire CMF file is considered validated.
- The new **Layout** menu allows you to build your own Schema Editor user interface by opening multiple intelligent controls that update automatically when you select an object on one of the forms.
- You can create name display rules from the **MetaSchema Extension** environment in the Schema Editor. Name display rules define how names of objects are displayed in the Schema Editor user interface. You can also create name display rules by double-clicking the **NameDisplayRule** node in any meta schema view in the Schema Editor. For more information, see *Schema Editor Naming Rules* (on page 71).

Version 2008

- The SmartPlant Schema Editor is the application used to modify the SmartPlant Foundation data model.
- The Schema Editor supports modifying multiple SmartPlant schema configurations at the same time using configuration (CMF) files.
- The SmartPlant schema can be managed as an object in SmartPlant Foundation. To edit the SmartPlant schema for integration, you can check out the schema CMF file in the SmartPlant Foundation Desktop Client and then launch the Schema Editor by right-clicking the CMF file.

- The Schema Editor also supports session files. Session files allow you to save registered tool information, file configuration information, tool map schema information for loaded map files, and displayed windows and dialog boxes that you want to restore when you reopen the Schema Editor. When you exit the Schema Editor, the software prompts you to save your session file so that you can return to your work easily. You can create, modify, and open session files in the Schema Editor using the **File > Session Files** commands. When you edit the CMF file from SmartPlant Foundation in version 2008 and above, the Schema Editor automatically saves a session file and uses it to determine the state of the Schema Editor when you launch it again from the Desktop Client.
- To provide a comprehensive representation of the mapping supported by an authoring tool adapter, the Schema Editor provides a new "for information only" map relationship. This allows authoring tool developers to identify hard-coded mapping for documentation purposes. You can flag a map relationship for information only by clicking the **For Information Only** button  on the **Publish** and **Retrieve** tabs in the **Edit Map Class Definition** window. ForInformationOnly is a property on the IMapRel interface.
- The Schema Editor now supports comparison for retrieve mapping. You can also specify properties that you do not want to map for retrieve. Previously, this functionality was only available for publish.
- During mapping, you can now select existing application enumerations or properties and automatically create the corresponding SmartPlant enumerations or properties. During creation of these objects, the Schema Editor also maps the application objects to the automatically-generated SmartPlant schema objects by creating map relationships between the two.
- The Schema Editor provides auto-correlation during mapping, which allows users to view a list of the best matches, based on a default algorithm, for mapping between map property definitions, map enumerated lists, and map unit of measure lists and existing objects in the schema. The **Auto-map** button is available on the **Publish** and **Retrieve** tabs on edit dialog boxes for map class definitions, map enumerated list definitions, and map unit of measure definitions.
- You can now select multiple view definitions for publish and retrieve in the Mapping Environment.
- During synchronize between the authoring tool meta data and the authoring tool map file, you can export the synchronization information into an Excel spreadsheet.
- The Schema Editor now supports merging a property definition into another, existing property definition. Merging property definitions adds a Transform object to the schema file that is copied to the transformations file when that file is created by the Schema Editor.
- When you export tool-to-tool mapping to Excel, the tool map file names appear at the top above each set of Tool columns to make it easier to read the mapping spreadsheet.
- If you select specific class definitions for searching, the Schema Editor saves the selected class definitions for your next search.
- Defining data sheets for SmartPlant Foundation is now done by system administrators in the Desktop Client.

SECTION 1

SmartPlant Schema Editor Overview

The SmartPlant Schema Editor allows you to perform various tasks required for configuring the SmartPlant schema and authoring tool map schemas for integration. The Schema Editor is also useful for familiarizing yourself with the SmartPlant schema and its class definitions, interface definitions, relationship definitions, and property definitions.

SmartPlant Foundation system administrators also use the Schema Editor to define the SmartPlant Foundation data model. Relationships between schema objects and other SmartPlant Foundation objects must be created in the Desktop Client, however.

Because there are so many different ways to use the Schema Editor, the Schema Editor documentation contains documentation for the most frequently used tasks in the Schema Editor. These tasks include:

- Using the most common Schema Editor views for the SmartPlant schema, tool map schemas, and data files
- Using file configurations to manage open files in the Schema Editor
- Using user interface configurations to customize the Schema Editor user interface
- Extending the SmartPlant schema
- Viewing and exporting authoring tool mapping
- Defining property mapping between authoring tool map schemas and the SmartPlant schema
- Comparing files

Depending on how you opened the Schema Editor, you may have opened a personalized menu configuration by default when you opened the software. The commands that you have access to in the Schema Editor depend on your personalized menu configuration. For more information about user personalized menus and the default Schema Editor roles, see *Personalized Menus* (on page 63) and *Schema Editor Roles* (on page 64).

SECTION 2

SmartPlant Schema Overview

The SmartPlant schema is an XML file that describes the structure of the XML files generated by the authoring tools in much the same way as a data dictionary describes the structure of a database. As tools publish documents in XML format, those documents must adhere to the format defined by the schema to ensure that the XML data can be loaded into SmartPlant Foundation and retrieved into the other authoring tools.

To work with the SmartPlant schema, you must understand the SmartPlant model. The SmartPlant schema uses a class/interface/relationship model. This model has the following characteristics:

- Component schemas define a group of classes.
- Class definitions define the schema information for concrete objects.
- Class definitions realize interface definitions.
- Interface definitions can imply other interface definitions.
- The schema hierarchy is defined by implies relationships between interfaces.
- Interface definitions expose property definitions.
- Relationships are defined between interface definitions, not between class definitions.
- Enumerated lists contain enumerated entries.

Shared object definitions are a group of similar class definitions.

The SmartPlant schema can be represented in UML (Unified Modeling Language). For more information about the UML model, see *UML Notation* (on page 45).

For more information about the SmartPlant schema model, see the *SmartPlant Schema Editor User's Guide* and *A Gentle Introduction to the SmartPlant Schema*, delivered with the SmartPlant Foundation Help.

See Also

Meta Schema (on page 41)
Tool Map Schemas (on page 259)
XML Files (on page 43)

Class Definitions

A class definition is a named description of a set of objects that support or realize the same interface definitions and share the same property definitions and relationships. In the schema, class definitions can represent physical things, such as pumps, or conceptual things, such as projects.

Class definitions have the following characteristics:

- Every class definition belongs to one and only one component schema.
- Every class definition has a primary interface definition that defines the set of possible roles (interface definitions) for the class definition.
- Every instance of a class definition is instantiated by a class factory.

Class Definitions and Interface Definitions

Class definitions offer different roles with which the software can interact. Class definitions expose or realize their roles through abstract entities called interface definitions.

Class definitions have a Realizes relationship with interface definitions. This means that instances of a particular class definition support or implement the realized interface definitions. For example, the Vessel class definition realizes the IVessel interface definition. The IVessel interface definition exposes property definitions, such as RatedVolume, for class definitions that realize the IVessel interface definition.

Class definitions must also realize implied interface definitions. If an interface definition implies another interface definition, then any class definition that realizes the first interface definition must also realize the implied interface definition.

Class Definitions and Property Definitions

Class definitions are made up of property definitions and can share property definitions, but not the data associated with the properties. Interface definitions expose the property definitions for class definitions by combining the property definitions in a named collection called an interface definition.

Specializations of Class Definitions

You can also create a specialization of a class definition. When you create a class definition specialization, you are creating a new class definition using an existing class definition as a template or starting point. By using a prior class definition as the template, all interfaces from the class definition being used as the starting point are copied to the new class definition. The new class definition also has all the methods and properties attached to these interface definitions.

When you create a class definition specialization, an interface with the same name as the class definition preceded by an **I** prefix is created and attached to the new class definition using the Realizes relationship. Any property definitions that you select when you create the class definition are associated to the new interface by an Exposes relationship.

See Also

SmartPlant Schema Modification Rules (on page 156)
Create a Class Definition (on page 169)

Model Definitions

A model is a non-redundant description of an object. For example, a 3D plant model is a non-redundant 3D description of a plant (non-redundant = non-excessive).

A model definition (ModelDef) is the schema information used to describe the model. Model definitions in the SmartPlant schema are roughly based on the POSC- CAESAR model.

The primary function of a model definition is to identify those objects that can be shared through a ModelClass relationship. Objects can only be shared if they exist within the same model definition. Objects in different model definitions can be related, but they cannot be shared.

Model definitions define domains of data. The SmartPlant schema contains seven model definitions:

- **MetaModel** - Used only by the class definitions in the meta schema. You should never use this model for schema.
- **OrganizationModel** - Used for class definitions that are organizational concepts. Examples would be User, Company, Department, Manufacturer, Vendor, and so on.
- **WorkBreakdownModel** - Used for class definitions that provide organizations of work. This includes the class definitions that you see in the WBS documents such as Project and Contract as well as all the document class definitions defined in the schema.
- **FacilityModel** - Used for class definitions that represent abstract concepts in a process plant. Examples include Plant, Area, Unit, System, Discipline (although you could put Discipline in the OrganizationModel instead.)
- **PlannedMaterialModel** - Used for class definitions that represent real objects that you plan to purchase to construct your design. This is the "design" arena in which engineering tools like Aspen Basic Engineering, SmartPlant P&ID, and SmartPlant Instrumentation work. In a traditional SmartPlant Foundation model, this would be the "tag".
- **TypicalMaterialModel** - Used for class definitions that represent real objects offered by manufacturers or vendors. When an engineer looks for an object to fulfill the requirements of his design, he might look through a manufacturer catalog. This is the domain of this model. In a traditional SmartPlant Foundation model, this would be the "model", "catalog", or "standard equipment".
- **ActualMaterial** - Used for class definitions that represent what is purchased and assembled to construct the plant. In a traditional SmartPlant Foundation model, this would be the "asset".

Evolution for objects is normally from planned facility (FacilityModel) to planned material (PlannedMaterialModel) to actual material (ActualMaterialModel). Catalog parts are part of the typical material model. Meta schema objects are part of the meta model. The organization model contains objects that are outside the normal set of designed objects; these objects describe the organization rather than the design. The work breakdown model contains object associated with the work being performed rather than the objects for that work.

NOTE All model definitions are predefined in the meta schema and cannot be modified.

See Also

Shared Object Definitions (on page 21)

Shared Object Definitions

Shared object definitions are used to group together similar class definitions that define the same object in different domains. Class definitions that can be shared have a Sharing relationship with shared object definitions.

Shared Object Definition Example

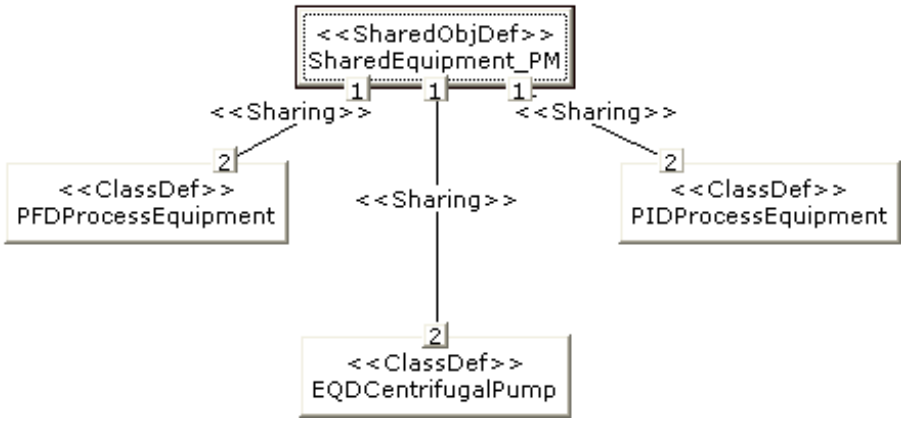
In concurrent engineering, two different authoring tools can create and publish the same object. For example, Aspen Basic Engineering and SmartPlant P&ID can both create and publish the same pump. In Aspen Basic Engineering, a user places a pump called P100 on a PFD and publishes the document. At the front end of the engineering process, very little is known about the pump other than the following:

- Some kind of pumping function is required.
- The pump is connected to streams.
- The streams have a certain fluid code running through them.
- The pump needs to pump at a certain volume.

When the PFD is defined, the user does not know how the pumping function with these general requirements will be accomplished. That information is left up to the P&ID portion of the workflow. In the PFD, the user sees some interfaces and properties of the pump, but not everything that defines the pump in an integrated environment.

In SmartPlant P&ID, engineers start adding value to the design by adding more information, such as the type of pump needed and so on. When SmartPlant P&ID publishes the P&ID containing the same pump (P100) as part of the workflow, the users see more interfaces because the pump object is being enriched as it moves through the design process. When the engineer updates the Equipment Data Sheet in Aspen Basic Engineering, there are thousands of properties describing the same object, further enriching it.

In each authoring tool, the pump (P100) is published with a different class definition, including PFDProcessEquipment, PIDProcessEquipment, and EQDCentrifugalPump. However, a shared object definition called SharedEquipment_PM indicates that these three class definitions all define the same object in an integrated environment. The three class definitions coexist in a sharing relationship. The shared object definition collects information to indicate this sharing relationship.



If you look at the realizes relationship for each class definition, you can see many of the same properties because, as the object moves from one tool to another, properties are updated and the information is enriched.

When an object is created in the PFD, it has a unique identifier for the object (UID). For example, PFD UID AA1 (P100) gets published. Then, SmartPlant P&ID retrieves P100, some additional work is done, and it gets published as PID UID AA2 (P100). The two publishes establish a Same As relationship. The Same As relationship indicates that object AA1 is the same as object AA2.

See Also

SmartPlant Schema Modification Rules (on page 156)
Model Definitions (on page 20)

Interface Definitions

An interface definition is a named collection of property definitions. Every interface definition is realized by one or more class definitions. Interface definitions expose the property definitions for class definitions. By sharing specific interface definitions, class definitions can also share property definitions, but not the data associated with the properties.

Interface Definitions as Roles

An interface definition represents a role for a class definition. The role defines both the properties and relationships of an object. Some interface definitions are defined to carry properties, and some are defined to carry relationships. Others may be defined merely to indicate a role.

Different class definitions can share the same interface definitions and, therefore, the same role. For example, every class definition in the schema shares the IObject interface, which means that every class definition in the schema has the role of an object. When a class definition has this role, it has an object name, an object description, an object identifier, and any other property definitions exposed by the IObject interface.

However, class definitions that share the IObject interface also realize other interface definitions that define other roles for them. These interface definitions expose other property definitions for the objects.

IObject

Every persistent object must be identified by a unique identifier (UID) which is a property on the IObject interface for that object.

Every relationship and relationship definition has a property UID1 that identifies the UID of the object at end 1 of the relationship and a second property UID2 that identifies the object at end 2 of the relationship.

The description of an interface definition includes its properties, methods and relationships. Since any interface definition is a potential end for a relationship or relationship definition, the extended definition of an interface, which includes the interface definitions directly or indirectly implied by that interface definition, must include IObject. Therefore, every interface definition must directly or indirectly Imply IObject.

For every interface definition that doesn't directly or indirectly imply IObject, an error will be reported during validation.

Interface Definitions and Relationships

Interface definitions define the relationships in which an object participates. There are no relationships between class definitions, only between interface definitions. That is because only the interface definitions are exposed to the outside world. The underlying reason for modeling class definitions and interface definitions instead of class definitions and relationships is because of the complexity of the relationships that can be formed.

In a traditional class data model, relationships go directly from one class definition to another. When the class hierarchy gets moderately deep, however, the relationships become very hard to understand. The consequence is that the modeler moves the relationships up in the hierarchy, which causes ambiguity in defining just what the relationship represents. By grouping similar characteristics together and exposing them as an abstract entity called an interface definition, it is easier to maintain precision in the relationships.

There are two relationships defined between class definitions and interface definitions:

- Realizes relationship
- Primary Interface relationship

Realizes Relationship

The first and most common relationship between class definitions and interface definitions is the Realizes relationship. A class definition Realizes an interface definition. The Realizes relationship may be required or optional. If it is required, the Schema Component will require that an instance of that class definition realize an instance of the required interface definition.

Primary Interface Relationship

The second relationship defined between class definitions and interface definitions is the Primary Interface relationship. The primary interface definition defines the set of possible roles for a class definition and should imply everything that is known about that class definition. Primary interface definitions, rather than class definitions, actually define the objects in the schema. Class definitions fulfill a subset of the roles defined by the primary interface definition.

The most important purpose for the primary interface definition is to define the complete set of roles for an object. It defines what an object is and everything that is known about it. For example, the IEquipmentOcc interface definition defines what a piece of equipment is in the schema, and IInstrumentOcc defines what an instrument is.

A class definition does not necessarily include all the roles of the primary interface definition because the class definition is the view that a single authoring tool or component schema has of that object. For instance, PIDProcessEquipment does not realize all the interface definitions in the implies hierarchy for IEquipmentOcc because SmartPlant P&ID does not manage all those interfaces, properties, and relationships, only a subset of them.

This complete set of roles for a class definition, defined by the primary interface, is used in several ways in an integrated environment:

- **Validation** - Class definitions may not realize any interface definition outside of the implies hierarchy of the primary interface definition.
- **Sharing** - Class definitions that are shared must have the same primary interface definition or must imply the same primary interface definition. For example, PFDProcessEquipment and PIDProcessEquipment have the same primary interface definition and are defined as shared. EQDReactorVessel does not have the same primary interface definition, but it does

imply the same primary interface definition (IReactorVesselOcc implies IEquipmentOcc). Therefore, EQDReactorVessel is defined as shared with PFDProcessEquipment and PIDProcessEquipment.

- **Mapping** - You can map the SmartPlant schema to an authoring tool schema using class definitions. However, you might end up mapping the same interfaces and properties over and over. For example, if your application retrieved equipment tags, you would want to retrieve PFD, P&ID, and EQD. To map by class definition, you would end up mapping to PFDProcessEquipment, PIDProcessEquipment, and dozens of EQD* class definitions. If you map to the primary interface definition for these class definitions (IEquipmentOcc), you only have to map to it once, and you can retrieve any object that realizes that interface definition.

NOTES

- The primary interface is used for schema modeling; it bears no relation to instantiation in SmartPlant Foundation. Only Realizes relationships are instantiated in SmartPlant Foundation.
- An interface must be realized to be used on a form in SmartPlant Foundation.
- In the SmartPlant schema, primary interfaces typically end in "Occ".

Implies Relationship

The definition of an object is completed through Implies relationships between interface definitions. When you consider the complete hierarchy of Implies relationships for the primary interface definition that a class definition realizes, you know everything about that object in an integrated environment.

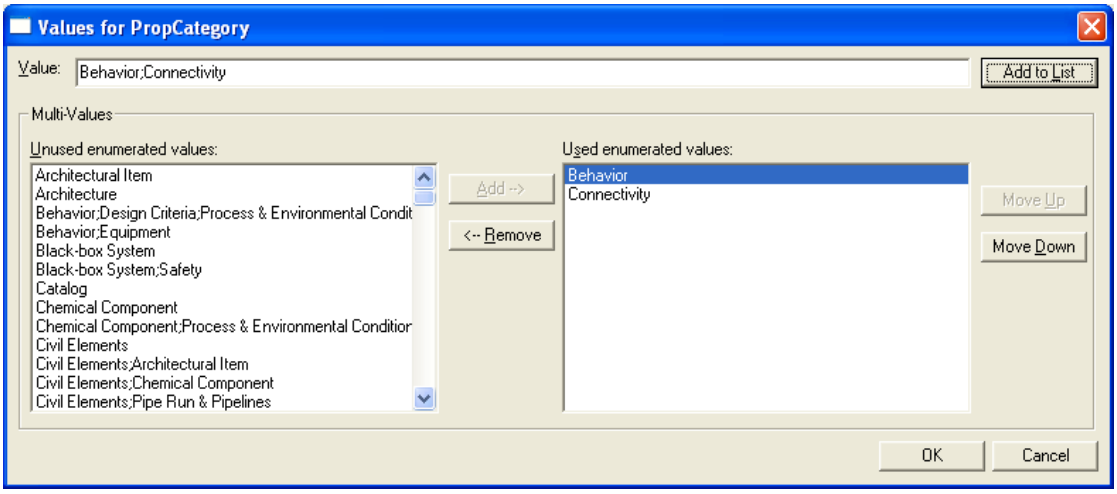
If an interface definition implies another interface definition, then any class definition that realizes the first interface definition can also realize the implied interface definition. For example, IEquipmentOcc implies the IDrawingItem. Therefore, any class definition, such as PIDProcessEquipment, that realizes IEquipmentOcc can also realize ISolidItem. If an Implies relationship between two interface definitions is required, then all class definitions that realize the first interface definition must also realize the second.

Properties Categories

The property category helps organize properties in the **Properties** window in SmartPlant Foundation. Property categories can be defined by the interface definition or edge definition. If no property definition is defined for the edge definition, properties across the edge definition are assigned a property category based on the property category defined for the interface definition that exposes the properties.

Special Handling for PropCategory Property Definition

Property Category is a special property definition in that it is a consolidated enumerated value in which the individual enumerated values are separated by semi-colons (;). When you create or edit an interface definition, select the **Browse** next to the **Property** category box. The **Values for PropCategory** dialog box allows you to define this property value.



Value - Displays enumerated values separated by the semi-colon delimiter for property value.

Unused enumerated values - Lists all the entries in the enumerated list that have not been used as part of the Property Category value

Used property values - Lists all the enumerated values that have been used as part of the Property Category value.

A number of options exist for editing the Property Category value:

- The value can be keyed into the Value box.
- Unused enumerated values can be selected and then by clicking **Add** inserted into the list of used enumerated values.
- Used enumerated values can be selected and then by clicking **Remove** deleted from the list of used enumerated values
- The order of the used enumerated values can be changed by selecting a value from this list control and then clicking **Move Up** or **Move Down**

In addition, this dialog box allows the addition of a new combination of values (or simply a new value) to the enumerated list by clicking the **Add to List**.

Interface Definitions in SmartPlant Foundation

In SmartPlant Foundation, interface definitions are used to define which methods are available and which workflows are appropriate for objects that realize the interface definition.

See Also

Schema modification rules (on page 158)
Create an Interface Definition (on page 171)

Property Definitions

All property definitions for an object are exposed through its interface definitions and never directly by the object. The property definitions that apply to a particular interface definition are defined by the Exposes relationship between objects of type InterfaceDef and objects of type PropertyDef in the schema. A particular property definition for a given class definition is typically exposed by one, and only one, interface definition.

For example, the IObject interface definition exposes the following property definitions:

- **UID** - Unique identifier for the object. This identifier is only required to be unique within the SmartPlant schema.
- **Name** - Name of the object
- **Description** - Description of the object

All objects in the schema have these properties because all class definitions realize the IObject interface definition.

Property Types

Property types define the set of possible values for a property definition. The scoped by relationship definition specifies the property type that defines acceptable values, or scopes, a particular property definition. Every property definition is scoped by one and only one property type. All properties of that property definition must be of that property type.

Each property type is actually a class definition in the meta schema that realizes IPropertyType, which allows it to be a scoping property type for property definitions.

Basic varieties of property types in the SmartPlant schema include the following:

- **Boolean** - Specifies that the property can have only one of two values: True or False.
- **Double** - Specifies that the property is a double-precision floating point number.
- **EnumListLevelType** - Specifies that the property definition is a special property type that is used to describe all enumerations at a given level in an enumeration hierarchy.
- **EnumListType** - Specifies that property values are defined by an enumerated list. For more information about enumerated list types, see *Enumerated Lists* (on page 27).
- **Int** - Specifies that property values for this property are integers.
- **String** - Specifies that values for the property can contain alphanumeric strings with some punctuation, such as periods (.), underbars (_), question marks (?), and so on.

  NOTE Commas are automatically replaced with semicolons when entered in properties of type string.
- **UoMListType** - Specifies that values for the property can be a numeric value (double) combined with a unit of measure from the appropriate unit of measure list, along with some optional criteria. For more information about unit of measure list types, see *Units of Measurement* (on page 31).
- **YMD** - Specifies that the property value is a date (year, month, day).

See Also

SmartPlant Schema Modification Rules (on page 156)
Create a Property Definition (on page 172)

Enumerated Lists

In the SmartPlant schema, some property definitions are of the enumerated list type (EnumListType). These property definitions have a list of possible string property values defined for them in an enumerated list (sometimes called a "picklist" or "lookup"). Any value for a property definition of this type must match an entry in the list of enumerated property values defined for the property type.

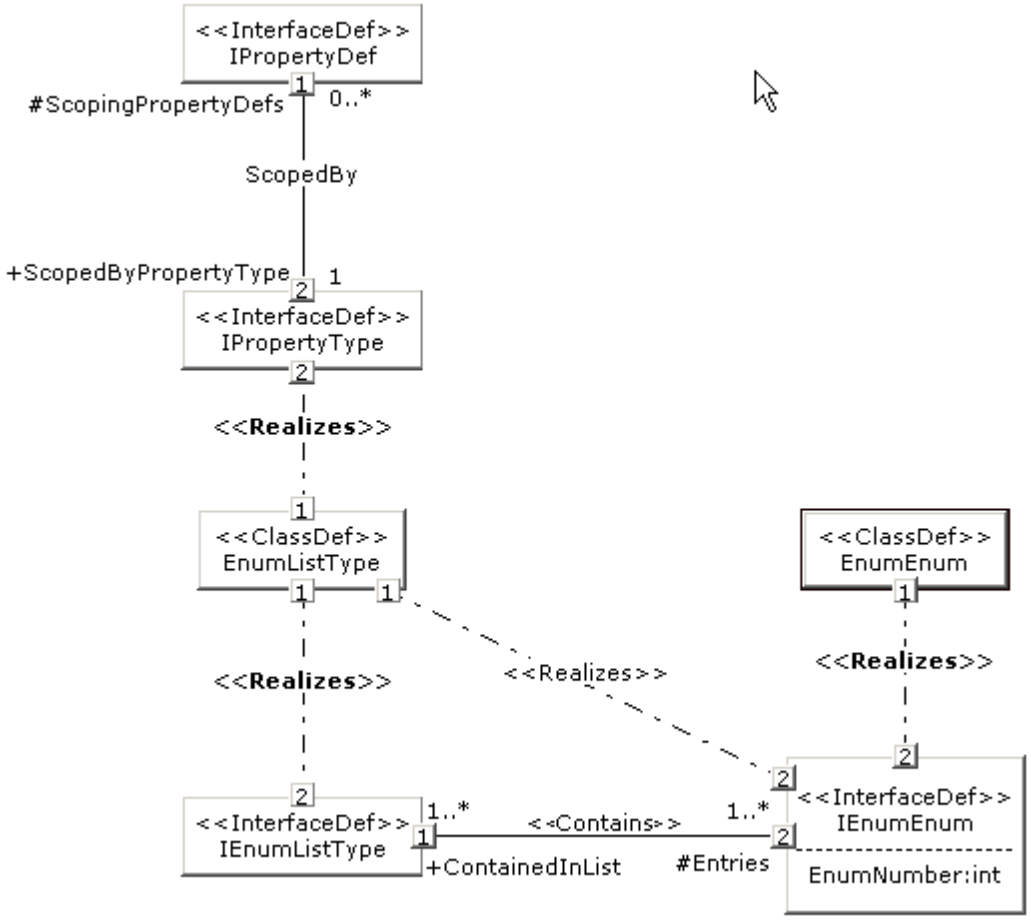
Each enumerated list contains one or more enumerated entries (EnumEnum). Each enumerated entry represents a possible value for a property scoped by that enumerated list.

Enumeration Hierarchies

Enumerated lists can contain all the possible enumerated entries for a property definition, or they can be combined to form an enumeration hierarchy. When you combine enumerated lists this way, you create child enumerated lists that are referenced by other enumerated lists. When this happens, a special property type (enumerated list level type) can be used to describe all enumerations at a given level in the enumeration hierarchy. When a property definition has an associated enumerated list level type, it effectively allows dependent picklists to be defined.

All nodes in the hierarchy that are EnumListTypes are enumerated entries for the EnumListType above them. All EnumListTypes must contain EnumEnums (list entries) or other EnumListTypes (enumerated lists).

The following UML diagram presents the SmartPlant schema model for enumerated lists and enumerated entries.



The EnumListType class definition is used for enumerated list property types. EnumListType realizes IPropertyType. EnumListType also realizes IEnumListType, the interface definition that supports enumerated list behavior. IEnumListType has a Contains relationship definition to IEnumEnum. Instances of this relationship are used to indicate the enumerated entries contained within the enumerated list.

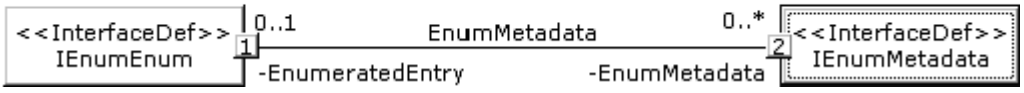
Leaf nodes within an enumerated list hierarchy are instances of the EnumEnum class definition. This class definition realizes IEnumEnum, the interface definition that supports enumerated entry behavior, and IObject, the interface that contains the short (Name) and long (Description) text values for the enumerated entry. Branch nodes within an enumerated list hierarchy are instances of the EnumListType class. EnumListType has an optional realizes relationship to IEnumEnum, which means that an enumerated list may or may not be an enumerated entry as well. For branch nodes within an enumerated list hierarchy, the EnumListType object will realize this optional interface.

NOTE By adding support for IEnumEnum to the EnumListType class definition, the need for the ConstrainsList between IEnumListType and IEnumListType and ConstrainsEntry between IEnumEnum and IEnumEnum relationship definitions has been eliminated. Therefore, these relationship definitions are no longer part of the SmartPlant meta- schema.

EnumMetadata Relationship

Sometimes enumerated entries are not just short and long text strings. In some cases enumerated entries have additional properties and/or additional behavior associated with them.

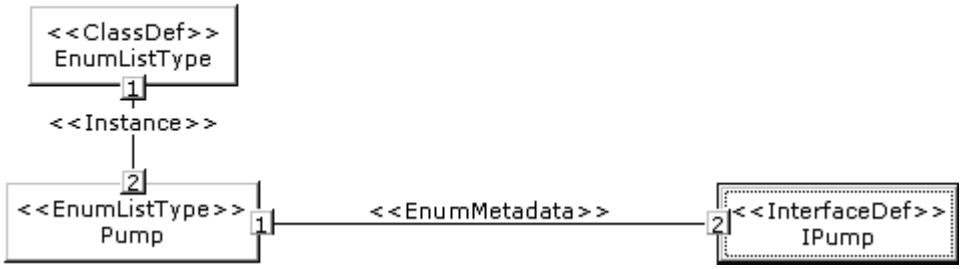
The **EnumMetadata** relationship definition is used to relate an enumerated entry (or an enumerated list that also supports the **IEnumEnum** interface) to other objects that provide additional properties and/or additional behavior to the enumerated entry.



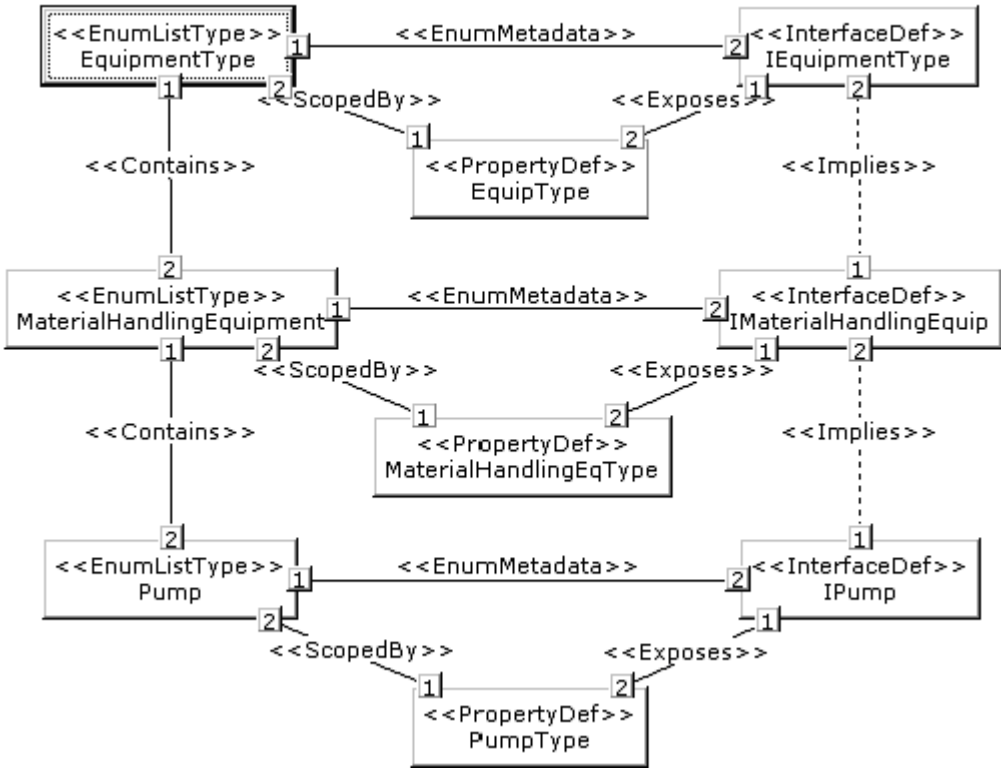
Each level in an enumerated list hierarchy is a specialization of the level above it. For example, pumps and compressors are specializations of mechanical equipment and reciprocating and centrifugal pumps are specializations of pumps. Therefore, type specialization can be considered to be an enumerated list hierarchy.

At each level of specialization in the type hierarchy, there can be **EnumMetadata** relationships to other objects that provide additional information and behavior about that enumerated entry.

One of the class definitions that supports **IEnumMetadata** is **InterfaceDef**. This means that an enumerated entry (or enumerated list that supports **IEnumEnum**) can have as **EnumMetadata** relationship to an interface definition that defines additional properties and behavior for that enumerated entry. For example, the **Pump** enumerated list could have an **EnumMetadata** relationship to **IPump**, which would then define additional properties and behavior common to pumps.



Since the various nodes in the type specialization hierarchy can have corresponding roles (interface definitions), the type specialization can have a corresponding role specialization. For the type specialization, each node in the enumerated list hierarchy would have a **Contains** relationship to its more generalized parent whereas the corresponding role specialization would have similar **Implies** relationships between the interface definitions.



The interface definition involved in the **EnumMetadata** relationship will normally have (unless it corresponds to a leaf node) an exposed property definition that is scoped by the related enumerated list. Therefore, **IPump** exposes a property **PumpType** that is scoped by the **Pump** enumerated list.

See Also

Property Definitions (on page 26)
Schema modification rules (on page 158)
Create an Enumerated List (on page 173)
Import Enumerated List Entries from a Spreadsheet (on page 174)

Units of Measurement

A unit of measure list type (UoMListType) is a special type of enumerated list with additional semantic information.

Conversion Factors

Each enumerated entry in the unit of measure list has conversion factors to convert from the SI units for that list to the specific units of measure selected by the user. In the schema, a UoMEnum contains two conversion factors: a multiplier (A) and an addend (B). These conversion factors are used to convert the value to SI units. For example, the given unit (x) is multiplied by A and then B is added to that value to determine the SI value (y = Ax + B).

A UoMListType is a composition of enumerated unit of measure entries. One entry in every unit of measure list has a relationship of HasDefaultSI with the UoMListType, which means that the entry is the default SI unit of measure for the list. For example, in the TimeUoM list, "s" (seconds) is the default SI unit of measure for time.

Other Characteristics of UOM Lists

A unit of measure list may have an associated enumerated list that identifies the possible conditions for that unit of measure list. For example, the pressure unit of measure list has an associated enumerated list that identifies whether the pressure value is absolute or gauge (@ Gauge).

A property definition that is typed as unit of measure has a double precision floating point value in the SI unit for that list. It can also have a preferred display unit of measure, a number of significant digits, and uncertainty.

See Also

Property Definitions (on page 26)
Schema modification rules (on page 158)
Create a Unit of Measure List (on page 175)

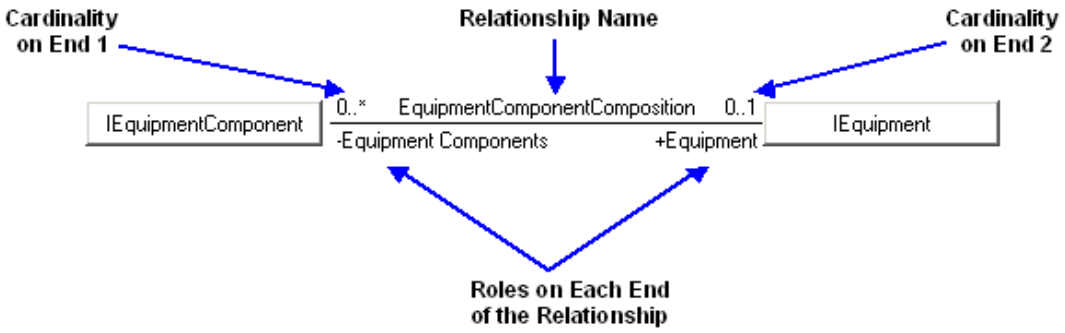
Relationship Definitions

Relationship definitions are associations between interface definitions. They identify two specific objects that fulfill the roles on each end of the relationship.

All relationship definitions in the schema are of class RelDef, which is part of the meta schema. A relationship definition has two ends, End1 and End2. The positive direction for navigating this relationship is from End1 to End2. The reverse direction is from End2 to End1. A relationship can only exist between objects that support (realize) the interface definitions at each end of the relationship definition.

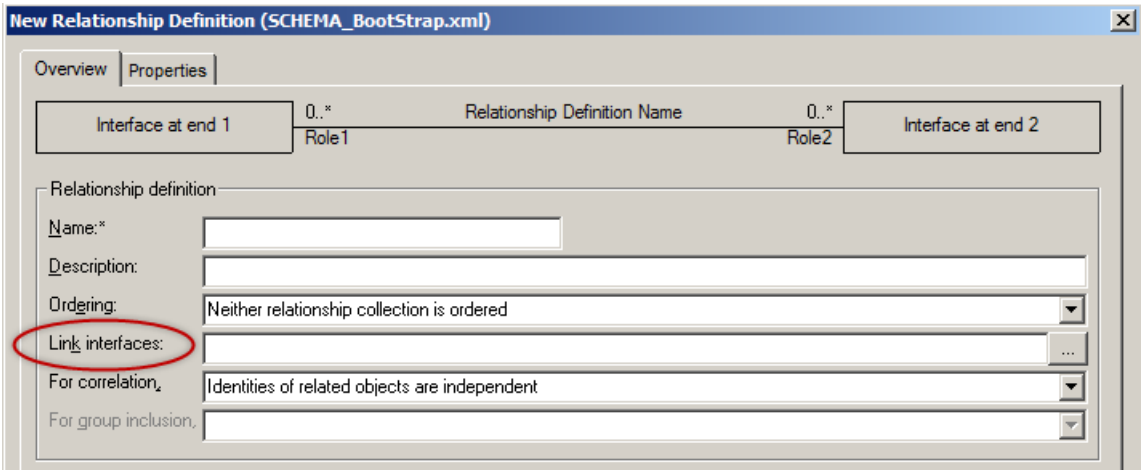
Relationship definitions also define cardinalities on each end of the relationship. Cardinalities provide the constraints for relationships by defining the minimum and maximum numbers of participants at each end of a relationship. For example, in the **EquipmentComponentComposition** relationship there can be zero to many (0..*) objects that have the role of **Equipment Components** but only zero or one (0..1) object that can have the role of **Equipment** in each instance of the relationship.

The following graphic shows the UML for the EquipmentComponetComposition relationship definition:



Link Attributes on Relationships

Relationships can also have link attributes or properties. Link attributes are defined by relating link interfaces to the relationship definition when you create the relationship definition in the **New Relationship Definition** dialog box in the Schema Editor. Because interfaces expose properties, you can use interfaces to define the properties that you want on or about the relationship definition.



For example, if you create a relationship between a company and employee, you could use the IEmployee interface to expose link attributes (properties) like employee number, salary, office number, and e-mail address on the relationship.

See Also

Schema modification rules (on page 158)
Create a Relationship Definition (on page 176)

Edge Definitions, Graph Definitions, and View Definitions

Edge definitions, graph definitions, and view definitions are used to provide different ways of viewing and traversing the data in SmartPlant Foundation.

See Also

Schema modification rules (on page 158)
Edge Definitions (on page 33)
Graph Definitions (on page 37)
View Definitions (on page 39)

Edge Definitions

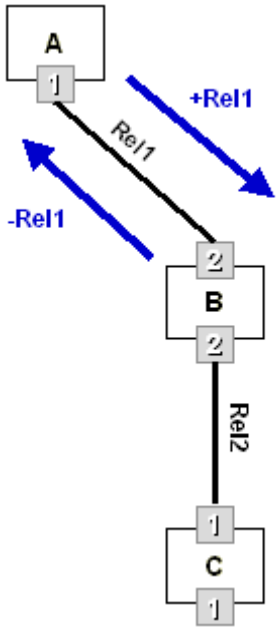
An edge definition is the definition of a path to follow to get from a starting object to one or more related objects. An edge definition can also include criteria to filter the objects across the specified relationships to only include a specific set.

Edge definitions have the following properties:

- A starting interface definition (required)
- A path definition that specifies the relationships to traverse from the starting interface definition along related interface definitions and the direction to traverse those relationships (required)
- Discrimination criteria that can be applied to objects at the end of the path (optional)
- Position criteria that can select a specific object from the set of objects that satisfy the discrimination criteria at the end of the path (optional)

Implicit Edges

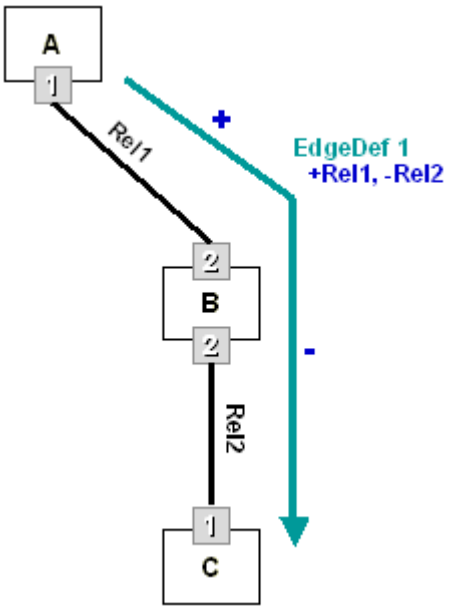
Two implicit edge definitions exist for every relationship definition in the schema: one that is positive (going from end 1 to end 2 of the relationship), and one that is negative (going from end 2 to end 1). The following graphic shows the positive and negative relationship edges (+Rel1 and -Rel1) between object A and object B.



Explicit Edge Definitions

In addition to the automatically generated edge definitions (called implicit edges), you can explicitly define edge definitions for a variety of purposes in the Schema Editor. Explicit edge definitions can be used to traverse across multiple relationships, to select only certain objects from the destination end, or to do a combination of multiple relationship traversal and selective discrimination.

The following graphic shows an explicit edge definition that traverses multiple relationships by going from object A to object C through the relationship that each object has with object B. The path for this relationship is +Rel1, -Rel2.

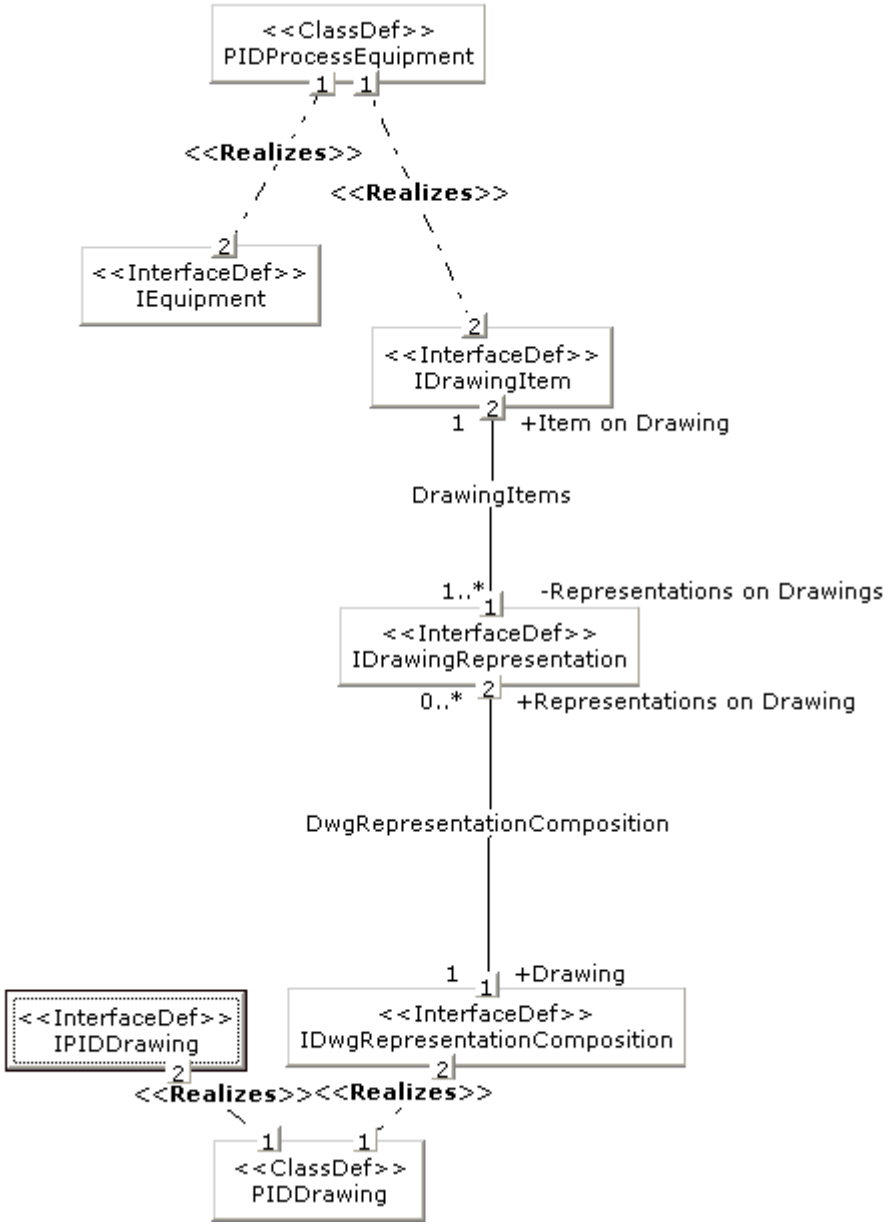


Discrimination Criteria

Edge definitions can specify discrimination criteria, including position, to help you filter the objects at the end of the path defined by the edge definition. For example, you can use position criteria in an edge definition to return the fifth nozzle on a piece of equipment by specifying a numeric position of 5 when you define the edge definition. Another option is to specify that you want the first or last object in the path by defining named criteria for the edge definition. You can also use discrimination criteria based on property values to specify that you only want to see objects with a Process Case of Minimum by creating an edge definition whose discrimination criteria included ProcessCase = Min.

Another way to use discrimination criteria to filter results is to specify that all objects at the end of the path must support a particular interface. For example, if you want to create an edge definition to display the P&ID an object is on, you could specify that all objects at the end of the path must support IPIDDrawing to make sure that only P&IDs and not other objects that participate in the DwgRepresentationComposition relationship are returned by the edge definition.

The following UML diagram shows an edge definition that uses discrimination criteria to go from a piece of equipment to the P&ID on which the piece of equipment appears:



Edge Definitions in SmartPlant Foundation

In SmartPlant Foundation, edge definitions are custom relationship expansions that appear on the shortcut menu when a user right-clicks an object that exposes the relationship referenced by the edge definition. Instead of requiring the user to go directly from object A to object B in the tree view, an edge definition can be created to allow the user to traverse through multiple intermediate objects and display an object at the end of the relationship that meets specific

criteria. For example, an edge definition could be created to expand to only the first nozzle on a vessel.

Without an edge definition, to get from object A to object D you would have to traverse from A to B, B to C, and then from C to D. With an edge definition, you could traverse directly from Object A to Object D.

System administrators might also create an edge definition to traverse directly from a plant to a unit. In SmartPlant Foundation, units are directly related to functional areas. However, a custom edge definition would allow the user to expand the plant to see the units without expanding a functional area first.

You can a set flag on the edge definition to determine whether all users have access to the edge definition. If the **Filter out edge in display** option is selected on the edge definition, then only those users in user groups that have been associated with the edge definition can see the edge definition in the SmartPlant Foundation client. Edge definitions show up in the relationships section of the shortcut menu for user groups that have access to the edge definitions.

See Also

Edge Definitions, Graph Definitions, and View Definitions (on page 33)
Graph Definitions (on page 37)
Create an Edge Definition (on page 177)

Graph Definitions

A graph consists of nodes and edges. The nodes (typically represented as circles) are connected to each other by the edges (lines). A directed graph is a special type of graph in which each of the lines goes in one direction only (typically represented using arrows). A directed graph traversal across an edge is always done in the one direction specified, never in the opposite direction.

The Schema Editor supports directed graph definitions (GraphDefs) for which the nodes are interface definitions and the edges are edge definitions. These directed graph definitions consist of a set of interface definitions (nodes) connected by a set of edge definitions (edges).

The definition of a directed graph begins by identifying the starting node (interface definition) for the graph. Once the starting interface definition has been defined, the next step in the process is to identify an edge (edge definition) to traverse. The list of edge definitions that can be used are the set of edge definitions that start at the specified interface definition or that start at an interface definition directly or indirectly implied by that interface definition.

When an edge (edge definition) is added to the graph, the node (interface definition) at the end of the edge is given a name. This name is used to identify the node for purposes of adding edges to the graph from that node.

After the starting node (interface definition) and the first edge (edge definition) have been defined, the directed graph consists of two nodes: one for the starting interface definition and one for the named interface definition at the end of the first edge. You can extend the graph definition by adding an edge to either the starting node (interface definition) or the node (interface definition) at the end of the first edge. You can continue to extend the graph definition this way until you have the graph definition that you want.

Occasionally, you may need to extend the graph from a specific node (interface definition) using an edge (edge definition) that starts at an interface definition that is not the interface definition for that node or an interface definition directly or indirectly implied by the interface definition for that node. In this case, you must identify a more "specialized" interface definition that either is or implies the starting interface definition for the edge definition that you want.

For example, consider a graph definition that starts at IEquipment and traverses the EquipmentEquipmentComponent relationship definition to reach a node corresponding to IEquipmentComponent. If you want to continue this graph definition from the node corresponding to IEquipmentComponent using a relationship definition that starts with INozzle, you must explicitly select INozzle rather than IEquipmentComponent for that node in the graph definition. Because INozzle is not implied by IEquipmentComponent, the relationships from INozzle are not, by default, available for continuing the graph definition.

Explicit Edge Definitions and Criteria

Every relationship definition results in two implied edge definitions: one for going from the interface definition at one end of the relationship definition to the interface definition at the other end and the second for traversing the opposite direction. For many graph definitions, the set of implied edge definitions is sufficient to define the graph definition. Occasionally a graph definition will require a more specific traversal than provided by the implicit edge definitions. For these cases, explicit edge definitions that define the more specific traversal are required.

Cases where explicit edge definitions are required include

- Traversing across multiple relationships as a single edge
- Traversing across a single relationship where only a subset of the objects at the other end of the relationship are required (the results need to be filtered)
- Traversing multiple relationships where filtering is needed as part of a single edge. An example of this is the traversal from a graphical object to its drawing, which traverses the DrawingItems and DwgRepresentationComposition relationships. Examples of filtering include selecting the object at a particular position (1, 2, and so on), selecting only those objects that support a particular interface, or selecting those objects that have a particular value for a property.

Directed Graphs

When a directed graph definition is applied to an object (or a set of objects), the result is a directed graph in which the nodes are objects and the edges are edges. A directed graph is therefore an instance of a directed graph definition (just as an object is an instance of a class). Where the directed graph definition consists of nodes that are interface definitions and edges that are edge definitions, the directed graph consists of nodes that are objects and edges that are edges. Therefore, a directed graph definition can start at interface definition IEquipment and traverse EquipmentEquipmentComponent to get to IEquipmentComponent. The corresponding directed graph can start at vessel V-100 and traverse to equipment components N1, N2, N3 and N4.

Graph Definitions in SmartPlant Foundation

In SmartPlant Foundation, graph definitions are used to define default expansions and alternate expansions for class definitions. They are also used to define data sheets.

If the name of a class definition is the same as the name of a graph definition, the graph definition is automatically used to define the default expansion for the class definition. For example, when you expand an object in the tree view, the graph definition with the same name as the class to which the object belongs defines the default expansion that you see.

You can also define alternate expansions for class definitions. For alternate expansions to show up on the shortcut menu when you right-click an object in the SmartPlant Foundation client, you must create a new method that uses the GetObjsbyGraphDefName client API and references

the appropriate graph definition. Default expansions created using graph definitions show up in the method section of the shortcut menu that appears when you right-click an object that exposes the starting interface defined in the graph definition.

See Also

Edge Definitions, Graph Definitions, and View Definitions (on page 33)
Edge Definitions (on page 33)
Create a Graph Definition (on page 180)

View Definitions

Directed graph definitions are useful for defining a network of objects to be traversed. While the objects resulting from traversing a graph definition (the graph for that graph definition) can be viewed as objects (the Schema Editor frequently does this), there is also a desire to see these objects from a flattened perspective.

Viewing objects from a flattened perspective can be useful for

- Viewing the graph as a table where each row in the table corresponds to one set of objects from the graph
- Viewing the graph as an Excel data sheet where cells in the data sheet correspond to specific properties from specific objects in the graph

View definitions are used to support this flattening of the directed graph. A view definition includes a selected subset of the complete set of properties that a graph definition exposes.

View definitions are SmartPlant equivalents of relational database views. A relational database view is a combination of joins and projects where the joins are used to retrieve the desired objects and the projects are used to determine which properties (columns) to include and what to call those properties. SmartPlant equivalents of relational database joins are directed graph definitions. The projection of this graph data is accomplished using view definitions.

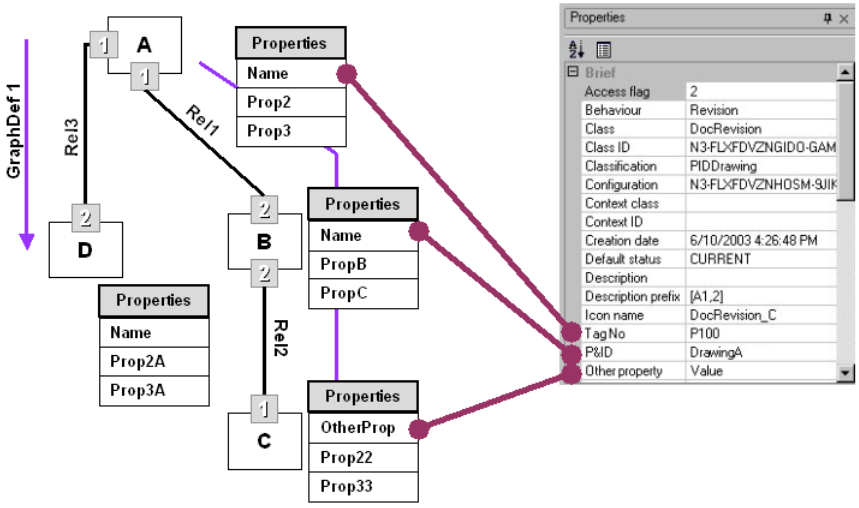
View definitions consist of the following:

- A starting interface definition
- Identification of the directed graph definition for the view definition
- A definition of the projection of property definitions from the directed graph definition, including modified display names as needed

A view definition is based on a directed graph definition and, therefore, like the directed graph definition, has a relationship to its starting interface definition. This interface definition should always be the same interface definition as that for its directed graph definition.

View Definition Example

The following graphic shows a view definition that includes three properties from three different objects exposed by a GraphDef1. The display names for the properties were modified in the view definition: **Name** for the first object was changed to **Tag No**, Name for the second object was changed to **P&ID**, and **OtherProp** was changed to **Other property**. The results are displayed in the SmartPlant Foundation Desktop Client **Properties** window.



View Definitions in SmartPlant Foundation

In SmartPlant Foundation, view definitions are used to define what users see in the **Properties** window when they select an object in the client. When you create ad-hoc reports in the SmartPlant Foundation client, you can also select the view definition that you want to use as the basis for the report as well as the properties from that view definition that you want to include in the report.

You can also use view definitions to create alternate views for classes. To create an alternate view for a class, you must do the following:

- Create the appropriate class definition in the SmartPlant schema
- Create a method that references the view definition in the Desktop Client
- Define user access for the method in the Desktop Client
- Associate the method with an interface in the Desktop Client

For all users who have access to the method, the method shows up on the shortcut menu for all class definitions that realize the interface definition.

See Also

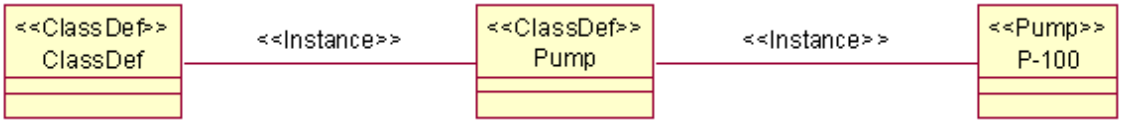
Edge Definitions, Graph Definitions, and View Definitions (on page 33)
Graph Definitions (on page 37)
Create a View Definition (on page 182)

Meta Schema

The meta schema is a set of schema objects that describe the objects in the SmartPlant schema. The schema, on the other hand, is a set of schema objects that describe data objects.

The meta schema provides the building blocks upon which the SmartPlant schema is built. The classes in the meta schema do not directly describe data that is part of the schema. Instead, these classes describe the classes in the schema.

In the following UML diagram, for example, ClassDef is the meta schema object that describes the Pump object in the schema. The Pump class in the schema is an instance of ClassDef in the meta schema. The Pump class in the schema describes P-100, which is the data. P-100 is an instance of the Pump class.



All classes in the schema are instances of the ClassDef class in the meta schema. Similarly, all relationship definitions in the schema are of class RelDef, which is part of the meta schema.

The meta schema also describes itself. For example, all classes in the meta schema are also of the class ClassDef, which is defined as part of the meta schema.

Viewing the meta schema without a schema or data is useful if you want to understand the classes that the meta schema contains.

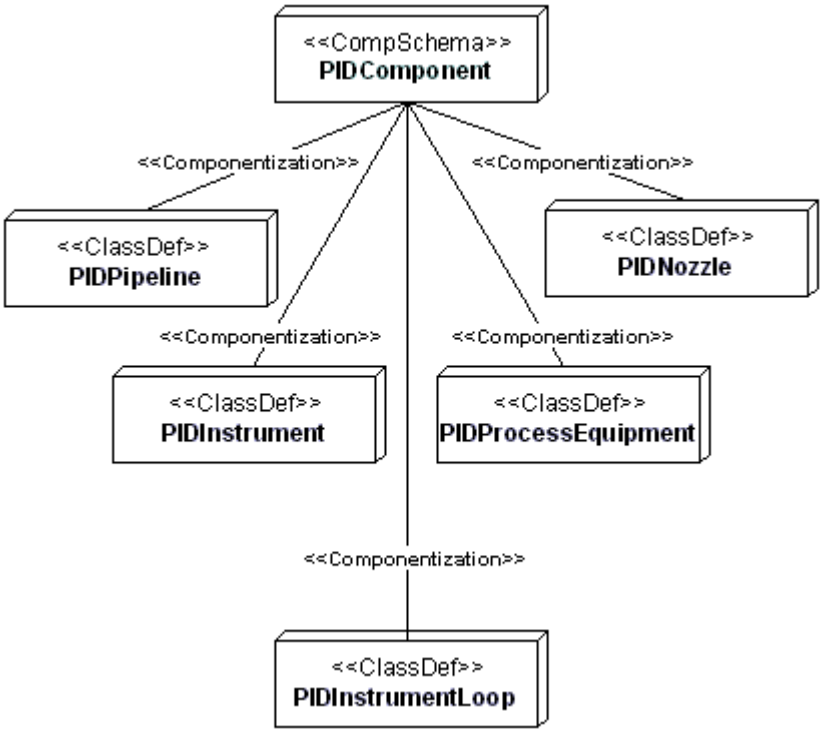
When you open the Schema Editor, the meta schema is already available for viewing before you open any other files. The meta schema is not stored in an external file and is always available in the Schema Editor.

See Also

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
Tool Map Schemas (on page 259)
XML Files (on page 43)

Component Schemas

A component schema is a subdivision of the complete SmartPlant schema. It contains the set of class definitions that are used within a specific domain or application area. For example, the **PIDComponent** component schema contains all the class definitions that are relevant to P&IDs.



Component schemas break up the SmartPlant schema into manageable chunks that correspond almost one-to-one to a document type being published by the authoring tools, such as a PFD, Equipment Data Sheet, or P&ID. At a minimum, there is a component schema defined for each authoring tool that publishes data. You can extract component schemas from the SmartPlant schema using the Schema Editor.

One of the primary purposes of the component schema is performance. When an authoring tool like SmartPlant P&ID indicates it is going to publish a certain document type, the SmartPlant Client loads only the necessary component schema instead of the entire SmartPlant schema.

Component schemas contain the following:

- The set of class definitions that have a componentization relationship with the component schema. For example, the **Vapor**, **Liquid**, **Solid**, and **BulkPhase** class definitions belong to the **PFDComponent** component schema, which means that the **PFDComponent** schema has componentization relationships with each of these class definitions.
- The interface definitions realized by those class definitions.
- The property definitions exposed by those interface definitions.
- The property types scoped by those property definitions.

IMPORTANT To modify component schemas, edit the SmartPlant schema in the Schema Editor and then regenerate the component schemas from the SmartPlant schema. Otherwise, your component schema changes could be lost.

Shared Objects

The component schemas are full of examples of shared objects that appear in more than one component schema. Virtually all of the schema data defined in a component schema, including class definitions, interface definitions, property definitions, and property types, may be shared by one or more other component schemas.

The unique identity of an object is defined by its unique identifier (UID) and does not depend in any way on its classification. Therefore, an object can be classified differently by different components and still be one shared object in the SmartPlant Foundation database. For more information about shared object definitions, see *Shared Object Definitions* (on page 21).

For example, if an object in one component schema is classified as class AA1 and the same object with the same UID is classified as class AA2 by another component schema, then the shared object in the SmartPlant Foundation database contains the information associated with the union of classes AA1 and AA2.

See Also

Create a Component Schema (on page 170)

XML Files

All documents that are published or retrieved in an integrated environment are in XML format. These XML files must conform to the structure defined by the SmartPlant schema.

When a set of objects, grouped by document, are published or retrieved, an XML file container is used to house those objects in XML format. The following XML format is used by the XML file container:

```
<Container>
   <classA>
   <interfaceX property1="value" property2="value">
   <interfaceY property1="value">
   .
   .
   .
   </classA>
   <classB>
   .
   .
   .
   </classB>
</Container>
```

The <class> elements identify objects of a particular class. Within each <class> element, there are the interfaces for that class. The <interface> elements expose the properties for the class. The property type for each property defines the acceptable values for that property in the XML file.

The following graphic displays a sample XML container file with authoring tool data as it might look in Internet Explorer:



See Also

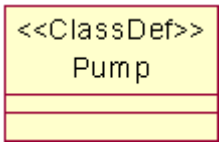
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
Meta Schema (on page 41)
Tool Map Schemas (on page 259)

UML Notation

In order to understand the UML diagrams used in the Schema Browser view, you must understand the basics of UML notation.

In UML all class objects are shown as rectangles with the name of the object and its stereotype at the top. Stereotypes define the types for classes and relationships. Stereotypes are always depicted in UML using the string <<stereotype>>, where *stereotype* is replaced by the appropriate value, such as ClassDef.

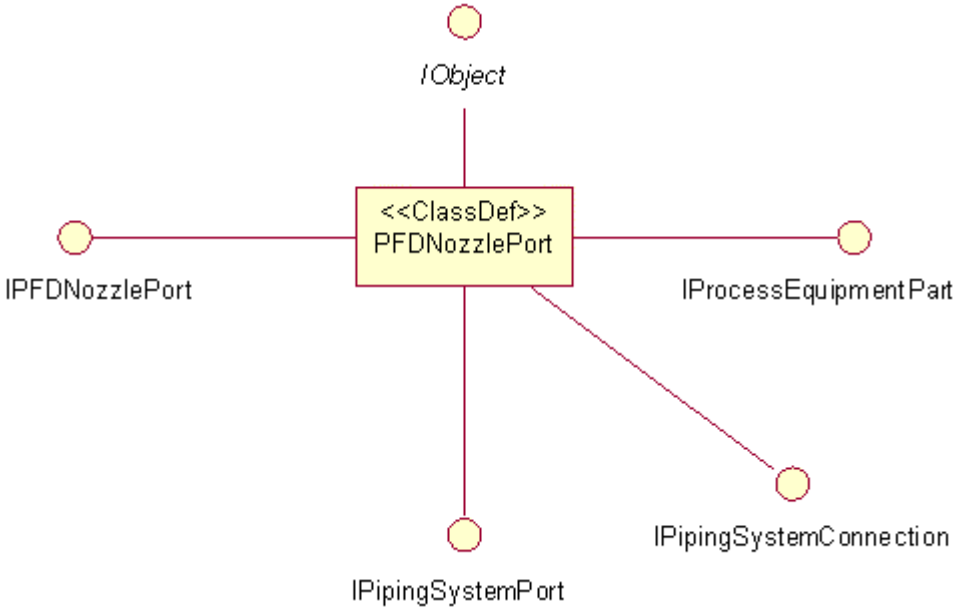
For example, Pump is stereotyped as a ClassDef, which means that Pump is a class definition in the schema.



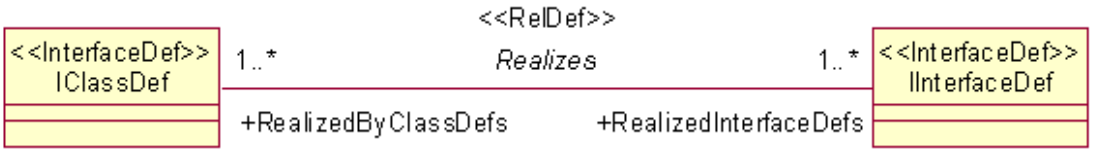
Relationships in UML are shown as lines that connect objects. In the SmartPlant schema, the UML for each relationship has a stereotype that identifies the relationship type. Therefore, the following UML indicates that a relationship of type ScopedBy exists between the UID object, which is of type PropertyDef, and the String object, which is of type StringType.



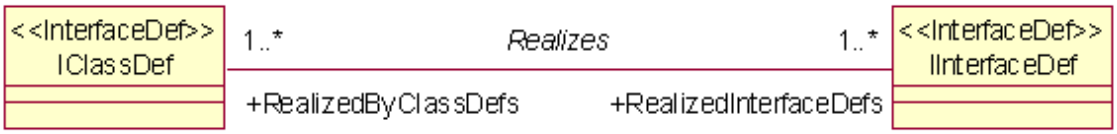
One special type of relationship, a Realizes relationship, has a special symbology within UML and may or may not have a stereotype because the stereotype is redundant with the special symbology. For example, this graphic shows the PFDNozzlePort and the interfaces it realizes.



Certain relationships, called relationship definitions, not only have a stereotype but are also named and have roles and cardinalities defined at both ends of the relationship. A relationship definition not only defines a relationship but also establishes the type for a set of other relationships. For example, the Realizes relationship is typed by the Realizes relationship definition in the following graphic.



In this example, the Realizes relationship definition has a RealizedByClassDefs role and a cardinality of one-to-many at end 1. At end 2, it has a RealizedInterfaceDefs role and one to many cardinality. Because relationship definitions can be identified by the relationship name, roles, and cardinalities, the stereotype for relationship definitions (RelDef) is frequently left out of the UML.



SECTION 3

Forward and Backward Schema Compatibility

Starting in the SmartPlant Foundation 2008 release, the Schema Editor and other SmartPlant integration components support forward and backward version compatibility. Integration supports the SmartPlant authoring tools' use of different versions of the SmartPlant schema from the version used in SmartPlant Foundation. Integrated SmartPlant tool versions 2007 Service Pack 4 and later can be integrated with SmartPlant Foundation 2008 and higher. Integration software automatically handles required transformations of data for tools. When the authoring tools release new versions, these will also be compatible with SmartPlant Foundation 2008 and higher.

When you are editing the schema using a CMF file, you can select the configurations you want to edit based on forward and backward compatibility. This allows you to add objects to one or more versions of the SmartPlant schema at the same time. However, if you publish new properties that are only available in a newer version of the SmartPlant schema to an older schema version supported by an older tool, the new properties that you add in the newer version of the schema will not be supported.

NOTE The Schema Editor, when working with application metadata adapters, relies on the version supported information in the SPMapSoftware object. If the SPMapSoftware object for a given application doesn't indicate support for the version of the schema being used, then a message will be displayed to the user warning him about proceeding.

SECTION 4

Schema Editor User Interface

In the Schema Editor, you can use the various user interface features to view the schema, tool map schemas, and data files. You can also edit the schema, tool map schemas, and data files using the Schema Editor.

IMPORTANT The commands and **Workflows** dialog box that you have access to in the Schema Editor depend on your personalized menu configuration. For more information about user personalized menus and the default Schema Editor roles, see *Personalized Menus* (on page 63) and *Schema Editor Roles* (on page 64).

Some of the key user interface features that you will use frequently in the Schema Editor include the following:

Menu Bar

The Schema Editor menu bar appears across the top of the Schema Editor application window.



NOTE Depending on your personalized menu configuration, you may not have access to specific menus or commands. For more information about user interface configurations, see *Personalized Menus* (on page 63).

The **File** menu provides standard file-based commands, such as **New**, **Open**, and **Save**.

The **Edit** menu allows you to undo and redo recent commands, edit and delete selected objects, and locate objects in the Schema Editor. You can use the **Cut**, **Copy**, **Paste/Create Relationships**, and **Paste/Copy and Edit** commands to modify open files. You can use the **Transaction Log** command to start and stop transaction logging and view processed transactions.

The **View** menu allows you to view the meta schema, schema, tool map schemas, and data files in a variety of ways. You can also display the Editor views, which allow you to modify the schema, tool map schemas, and data files, from the **View** dialog boxes displayed by commands on the **View** menu.

The **Tools** menu provides access to various data-specific, non-viewing commands. For example, you can validate files and set various viewing options using commands on the **Tools** menu.

The **Window** menu provides access to standard window manipulation commands, such as **Cascade**, **Tile Horizontally**, and **Tile Vertically**.

The **Workflows** menu allows you to open different workflows windows and dialog boxes to help you work in the Schema Editor more efficiently.

The **Help** menu provides access to the Schema Editor online Help, the printable user's guide for the Schema Editor, and information about the Schema Editor application.

Toolbars

Schema Editor toolbars provide easy access to the most commonly used commands. You can move the toolbars around in the Schema Editor window and dock them wherever you want. For descriptions of the commands on the Schema Editor toolbars, see *Schema Editor Toolbars* (on page 53).

Workflows

When you open the Schema Editor, a dialog box or window containing the most common workflows typically appears by default in the Schema Editor application window. This dialog box or window provides quick access to many of the same commands that appear on the Schema Editor menu bar.

You can use the following workflows in the Schema Editor:

- Standard Workflows
- Overall Workflows
- Basic Workflows

NOTE The **Workflows** dialog box or window that you see when you open the Schema Editor depends on your personalized menu configuration and what **Start** menu link you used to open the Schema Editor.

View Windows

In the Schema Editor, the view windows allow you to view the meta schema, schema, tool map schemas, and data files in a variety of different way. For example, you can see data in the Schema Editor in tree views, table views, properties tables, UML views, and so on. For more information about the various view windows, see example view topics for the meta schema, schema, tool map schemas, and data files.

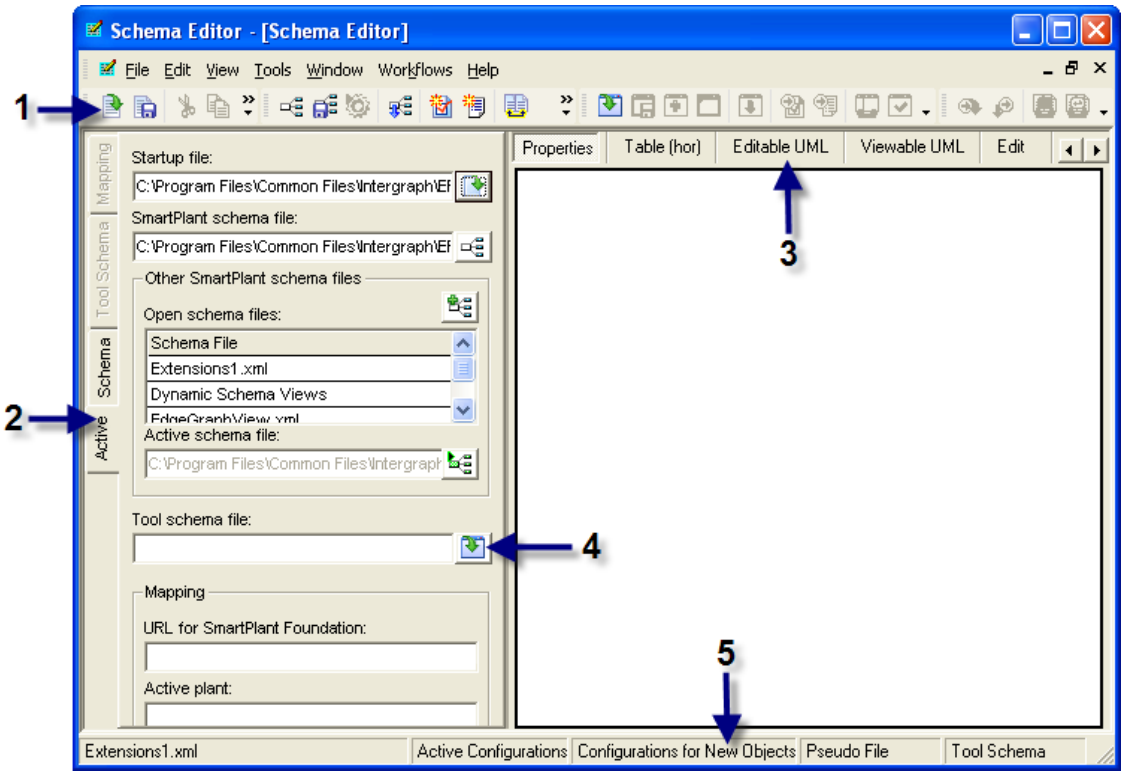
Shortcut Menus

When you right-click any object in any of the view windows, the software displays a shortcut menu. This menu displays context-sensitive commands for the selected object. These commands also appear on the **Actions** menu when an object is selected.

You can use shortcut menus to do everything from editing and creating objects and relationships in the schema, tool map schemas, and data files to changing UML views.

Standard Workflows Window

The Schema Editor **Standard Workflows** user interface has a tabbed window that provides access to the most commonly used features in the Schema Editor, such as viewing and modifying the schema and defining mapping between authoring tools. You can access this new user interface by clicking **Workflows > Standard Workflows** in the Schema Editor or by launching the Schema Editor from the following location on the **Start** menu: **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor - Standard**.



1 - Schema Editor Toolbars

Schema Editor toolbars provide easy access to the most commonly used commands. You can move the toolbars around in the Schema Editor window and dock them wherever you want. For descriptions of the commands on the Schema Editor toolbars, see *Schema Editor Toolbars* (on page 53).

2 - Vertical Tabs

Vertical tabs in the Standard Workflows window allow you to access different types of data in the Schema Editor.

Active Tab

The **Active** tab allows you to select a startup file and view related files. You can also open tool schema files, connect to SmartPlant Foundation for mapping, and synchronize tool map schemas and tool metadata for tools with metadata adapters. You can also change the active schema and open tool schemas from the **Active** tab.

Schema Tab

The **Schema** tab allows you to search the active schema for specific objects or browse the schema in the tree view. When you select an object in the tree view, information for that object is displayed on the active tab on the right side of the window.

Tool Schema Tab

The **Tool Schema** tab allows you to search the active tool schema for specific objects or browse the schema in the tree view. When you select an object in the tree view, information for that object is displayed on the active tab on the right side of the window.

Mapping Tab

The Mapping tab allows you to view loaded tool map schemas, expand them, and select tool map schema objects for which you want to define mapping.

3 - Horizontal Tabs

Horizontal tabs across the top of the **Standard Workflows** window provide easy access to different views of schema, tool schema, and mapping. To change the view, click a tab. The context of the tabs change depending on your selections on the vertical tabs on the left of the **Standard Workflows** window. You can add your own custom views to the list of horizontal tabs by right-clicking one of the tabs and clicking **Update MultiTab Options** on the shortcut menu.

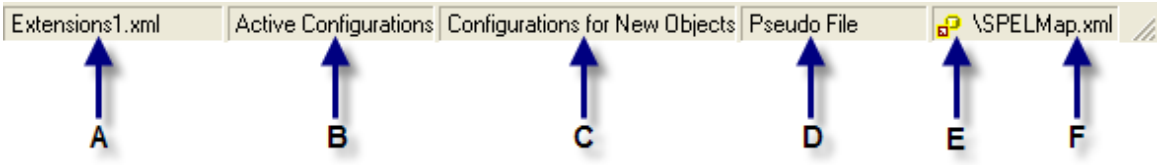
4 - Command Buttons

Command button on the Schema Editor vertical tabs allow you to identify commands easily with icons. The following standard command buttons are available on some of the tabs in the **Standard Workflows** window.

Icon	Name	Description
	**Open Startup File**	Click this to specify the startup file for the Schema Editor, such as a session file, a CFG file, a CMF file, or an XML file.
	**Open Schema File**	Click this to select a schema file to open. If the selected startup file contains a schema file, the name and path for that schema file is displayed in the **SmartPlant schema file** box.
	**Open Another Schema File**	Click this to select another schema file to open.
	**Set Active Schema File**	Click this to select the active schema file when multiple schema files are open in the Schema Editor. The active schema is the one that you are editing and searching. Although you can see multiple schemas at once, you can only edit one at a time.
	**Edit Engineering Tool Options**	Click this to select another authoring tool map schema and tool metadata that you want to synchronize.
	**Find**	Click this to search for schema objects or tool map schema objects that match the criteria you define on the corresponding tab.

5 - Status Bar

The Schema Editor status bar allows you to view active files and settings and modify those settings.



A Specifies the name of the active schema file. Double-click the file name to open another schema file.

B Allows you to define the active configurations in the Schema Editor. This setting is used in version 2008.

C Allows you to define the create configuration for objects in the Schema Editor. This setting is used in version 2008.

D Allows you to define the pseudo file for schema editing. This setting is used in version 2008.

E Indicates whether you are working in connected or disconnected mode for tool metadata adapters. Different icons represent different states of connectivity.

- Connected
- Disconnected (Disconnected mode is not supported in Schema Editor versions 2007 or 2008. It will be supported in a future service pack when the authoring tools can also support this mode.)
- Offline. This icon appears when you open the tool map schema directly from the file system.

F Specifies the name of the active tool map schema. Double-click the file name to open another tool map schema.

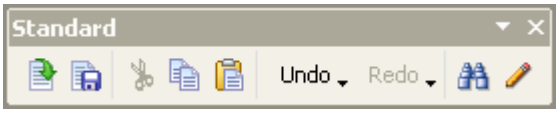
See Also

Schema Editor - Standard Overview (on page 55)
Schema Editor Toolbars (on page 53)

Schema Editor Toolbars

The following toolbars are available in the Schema Editor. Commands are available based on the files you have open and the selections you make in the Schema Editor window.

Standard Toolbar



Open Session File - Click this to open a Schema Editor session file. Session files allow you to save registered tool information, file configuration information, tool map schema information for loaded map files, and displayed windows and dialog boxes that you want to restore when you reopen the Schema Editor.

Save Session File - Click this to save the active Schema Editor session file.

Cut - Click this to remove the selected text or object from the Schema Editor. You can insert the object at a selected location using **Paste**.

Copy - Click this to copy the selected object or text in the Schema Editor.

Paste - Click this to insert the copied text or object into the selected dialog box, tree view, or other Schema Editor option or view.

Undo - Click this to cancel the last action you performed in the Schema Editor. If there are multiple actions that can be canceled, click beside the **Undo** button and select the action that you want to undo in the list.

Redo - Click this to repeat the last action you canceled in the Schema Editor. If there are multiple actions that can be repeated, click beside the **Redo** button and select the action that you want to redo in the list.

Find - Click this to search for objects in the Schema Editor. For more information about searching in the Schema Editor, see *Finding Objects in the Schema Editor* (on page 124) and *Find Objects Using Search Criteria* (on page 124).

Edit - Click this to modify the selected object in the Schema Editor tree view.

Schema Toolbar



Open Schema File - Click this to open a schema file in the Schema Editor.

Save Schema File - Click this to save the active schema file.

Set Active Configurations - Click this to set the active configuration for editing in the Schema Editor. The active configuration defines the version of the schema to which you want your changes to apply and where you want to store your schema extensions. Although the command is available in the 2007 Schema Editor, it is typically used in 2008 and greater versions of the Schema Editor for managing schema changes for integration.

View Active Schema - Click this to display the active schema file in the view you select. The active schema is the one that you are editing and searching. Although you can see multiple schemas at once, you can only edit one at a time.

Create Property Definition - Click this to create a new property definition in the schema. Depending on your startup options, your changes may be saved to an extensions file.

Create Enumerated List - Click this to create a new enumerated list definition in the schema. Depending on your startup options, your changes may be saved to an extensions file.

Compare/Display Results - Click this to compare the active schema file to another schema file that you select. For more information about comparing schema files, see *Comparison* (on page 463). Comparison results are displayed in a text editor.

Validate/Display Results - Click this to validate the active schema file and display validation results. When you validate a schema file, the Schema Editor checks the file against the meta schema and a set of semantic rules. Validation results are displayed in a text editor.

Tool Schema Toolbar



Open Tool Schema File - Click this to open a tool schema file.

Save Tool Schema File - Click this to save the active tool schema file.

Another Tool Schema File - Click this to open another tool schema file.

Set Active Tool Schema File - Click this to set the active tool schema file when multiple tool schema files are open in the Schema Editor. The active tool map schema is the file that you are editing and searching. Although you can see multiple tool map schemas at once, you can only edit one at a time.

View Tool Schema (Active) - Click this to view the active tool schema file in the view you select.

Create Map Property Definition - Click this to create a tool map property definition in the active tool map schema file.

Create Map Enumerated List - Click this to create a tool map enumerated list definition in the active tool map schema file.

Compare/Display Results - Click this to compare the active tool map schema with another tool map schema file. Comparison results are displayed in a text editor.

Validate/Display Results - Click this to validate the active tool map schema. When you validate a schema file, the Schema Editor checks the file against the SmartPlant schema and a set of semantic rules. Validation results are displayed in a text editor.

Mapping Toolbar



Edit Engineering Tool Options - Click this to select an authoring tool, load the map schema, and connect to the tool's metadata adapter. Selecting an authoring tool allows you to synchronize the tool map schema and the tool metadata and define mapping for that tool in the Schema Editor.

View Map Environment - Click this to display the Schema Editor Map Environment. The Map Environment allows you to view loaded tool schemas and meta data adapters and define mapping between the tool schemas and the SmartPlant schema.

Compare (Active Tool Schema) with Schema (Publish) - Click this to compare the active tool map schema with the SmartPlant schema to determine which map properties, map enumerated lists, and map unit of measure lists have not been mapped or flagged as not to be mapped for publish. For more information about comparing the active tool schema with the SmartPlant schema, see *Comparing Tool Map Schema with the SmartPlant Schema* (on page 304).

Compare (Active Tool Schema) with Schema (Retrieve) - Click this to compare the active tool map schema with the SmartPlant schema to determine which map properties, map enumerated lists, and map unit of measure lists have not been mapped or flagged as not to be mapped for retrieve.

Schema Editor - Standard Overview

In Schema Editor - Standard, you can access important Schema Editor features quickly and easily from a single user interface--the **Standard Workflows** window. The tabs and toolbars in the Standard Workflows window allow you to navigate and create in the schema and tool schemas without manipulating multiple windows.

NOTE If the **Standard Workflows** window is not displayed in the Schema Editor, click **Workflows > Standard Workflows** to display it before starting these tasks.

What do you want to do?

- *Open a Startup File* (on page 56)
- *Open a Schema* (on page 56)
- *Open Other Schema Files* (on page 56)
- *Open a Tool Schema* (on page 56)
- *Synchronize Another Tool Map Schema with Tool Metadata* (on page 57)
- *Search for Schema Objects* (on page 58)
- *Search for Tool Schema Objects* (on page 59)
- *Define Mapping* (on page 59)

Open a Startup File

1. On the **Active** tab, click **Open Starting File** beside the **Startup file** box.
2. Browse to the XML, CFG, CMF, or session file that you want to open.
3. Click **Open**.

TIPS

- Different startup files open differently in the Schema Editor. For example, a configuration file (CFG) can open multiple schemas and tool schemas and sets the active files. Opening a session file, on the other hand, opens schemas and tool schemas but also maintains the open dialog boxes and connections to SmartPlant Foundation that were active when the user saved the session file.
- The options on the **Active** tab change depending on the type of startup file you select to open.

Open a Schema

1. On the **Active** tab, click **Open Schema File** beside the **SmartPlant schema file** box.
2. Browse to the schema that you want to open, and click **Open**.

Open Other Schema Files

1. On the **Active** tab, click **Open Another Schema File** under **Other SmartPlant schema files**.
2. Browse to the schema that you want to open, and click **Open**.

Set the Active Schema File

1. On the **Active** tab, click **Set Active Schema File**. The active schema is the one that you are editing and searching. Although you can see multiple schemas at once, you can only edit one at a time.
2. Select the schema that you want to be active in the **Set Active File** window, and click **OK.**

Open a Tool Schema

1. On the **Active** or **Mapping** tabs, click **Open Tool Schema File**.
2. Browse to the tool schema that you want to open, and click **Open**.

TIP The active tool schema file appears in the bottom right of the Schema Editor status bar. The icon beside the tool schema name indicates whether you are working in connected or disconnected mode. In connected mode, your changes are typically saved to the tool map schema and the tool metadata database when you save the tool map schema. You can double-click the tool schema name in the status bar to open a new one.

Synchronize Another Tool Map Schema with Tool Metadata

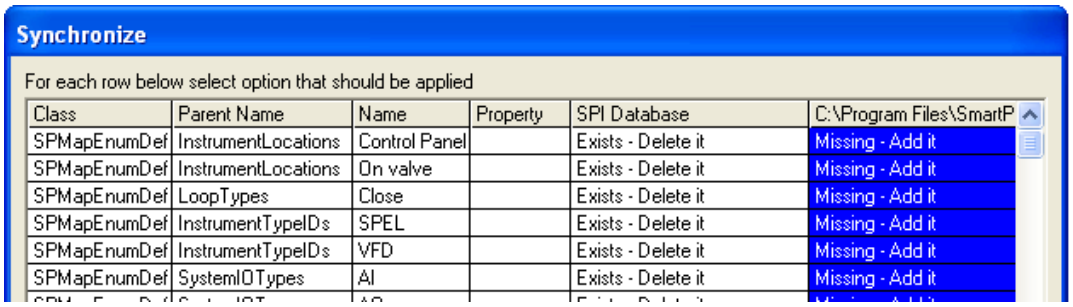
When you first connect to SmartPlant Foundation, you can select the authoring tool that you want to synchronize. To synchronize another authoring tool, use the following procedure.

1. On the **Active** or **Mapping** tabs, click **Edit Engineering Tool Options** .
2. Under **Engineering application/map schema**, select the authoring tool and tool map schema that you want to work with in the Schema Editor.

   TIP The authoring tool that you want to define mapping for must be registered with the selected SmartPlant Foundation site and plant to appear in the Engineering application/map schema list.
3. Under **Startup** options, select the **Load map schema** check box.

   The software automatically selects the **Connect to application schema** check box because the Schema Editor does not currently support working in disconnected mode with metadata adapters. In a future release, disconnected mode will allow users to modify the tool map schema without connecting to the tool's metadata adapter, which means that no synchronization will occur.

   TIP For specific details about what types of objects that you can edit in the authoring tool via the metadata adapter and how save works for the tool's metadata adapter, select a tool and then click the **Details** button.
4. Click **Finish** to review synchronization details in the **Synchronize** dialog box.



TIPS

- Connecting to the metadata adapter and generating a comparison of the tool metadata and the tool map schema can be a time consuming task, depending on your computer memory and processor speed.
- When you click **Finish**, the Schema Editor generates tool schema objects from the tool metadata in the tool database and compares them to the tool map schema file.
- If no changes have been made to the authoring tool metadata or the tool map schema since you installed the software, there may no differences to synchronize.
- Once you are familiar with synchronization and the default actions for the authoring tool you are working with, you may want to select the **Automatically synchronize schemas** check box before you click **Finish** to avoid reviewing synchronization results every time you load a tool map schema and tool metadata.
- You can click **Save to Excel** to save the synchronization results to a Microsoft Excel spreadsheet.

5. In the **Synchronize** dialog box, select the appropriate action in the **<Tool> Database** or **<Tool Map Schema Path>** column in the appropriate row for the change to update inconsistencies between the tool metadata and tool map schema. Default actions appear in blue in the appropriate column and row for each difference.

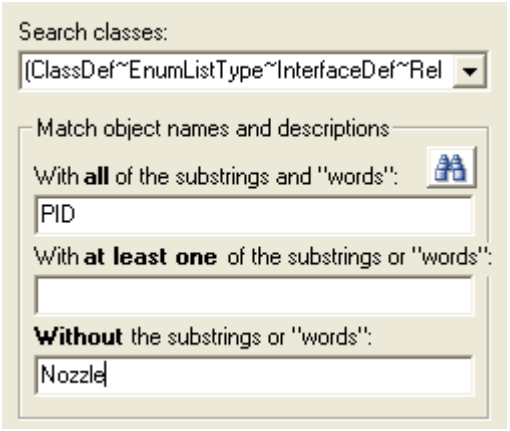
   IMPORTANT Some applications do not support both possible synchronization actions that appear in the Synchronization dialog box. For example, SmartPlant P&ID does not support deleting existing properties or enumerated lists. If properties exist in the SmartPlant P&ID tool metadata that are not in the tool map schema, then the missing properties must be added to the tool map schema. They cannot be deleted from the tool metadata database. If you select a synchronization action that is not supported by the selected authoring tool, the Schema Editor displays a message box to let you know that the action is not supported.

6. Click **OK** to synchronize the tool map schema and tool metadata.

   TIP After you synchronize the last tool map schema, the **Map Environment** appears.

Search for Schema Objects

1. On the **Schema** tab, click ▾ beside the **Search Classes** box to define the search classes that you want to include in your search.
2. Under **Match object names and descriptions**, type text you want to include and exclude from your search.



3. Click **Find** to search for objects matching your criteria in the active file.

   Search results appear in a tree view under the search criteria. Nodes are organized by the type of results, such as ClassDef and InterfaceDef.

4. Select an object in the tree view to view more information about that object on the tabs in the right pane. To switch between tabs, click the tab name at the top of the **Standard Workflows** window.

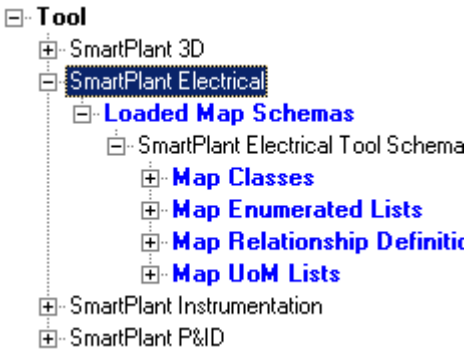
Search for Tool Schema Objects

1. On the **Tool Schema** tab, click ▾ beside the **Search Classes** box to define the search classes that you want to include in your search.
2. Under **Match object names and descriptions**, type text you want to include and exclude from your search.
3. Click **Find** to search for objects matching your criteria in the active file.

   Search results appear in a tree view under the search criteria. Nodes are organized by the type of results, such as ClassDef and InterfaceDef.
4. Select an object in the tree view to view more information about that object on the tabs in the right pane. To switch between tabs, click the tab name at the top of the **Standard Workflows** window.

Define Mapping

1. On the **Mapping** tab, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
2. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>* Tool Schema** node to display contents of the authoring tool map schema.

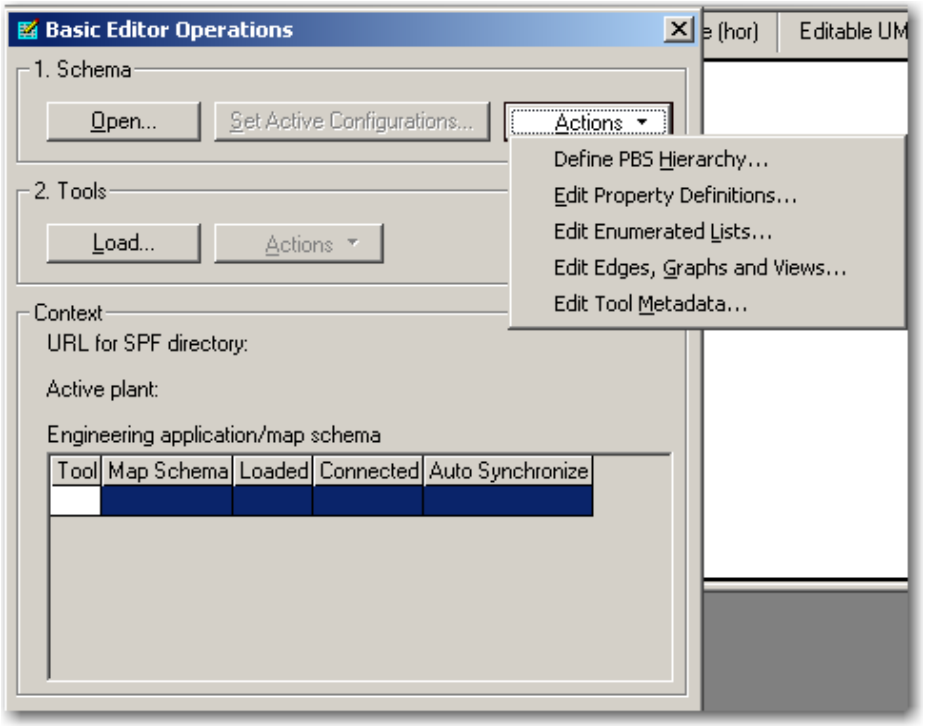


3. Expand the node for the type of map schema object that you want to create or modify.
4. Use the horizontal tabs across the top of the Standard **Workflows** window to modify mapping. For more information about defining mapping in the Schema Editor, see *Work Processes for Mapping for Tools with Metadata Adapters* (on page 327).

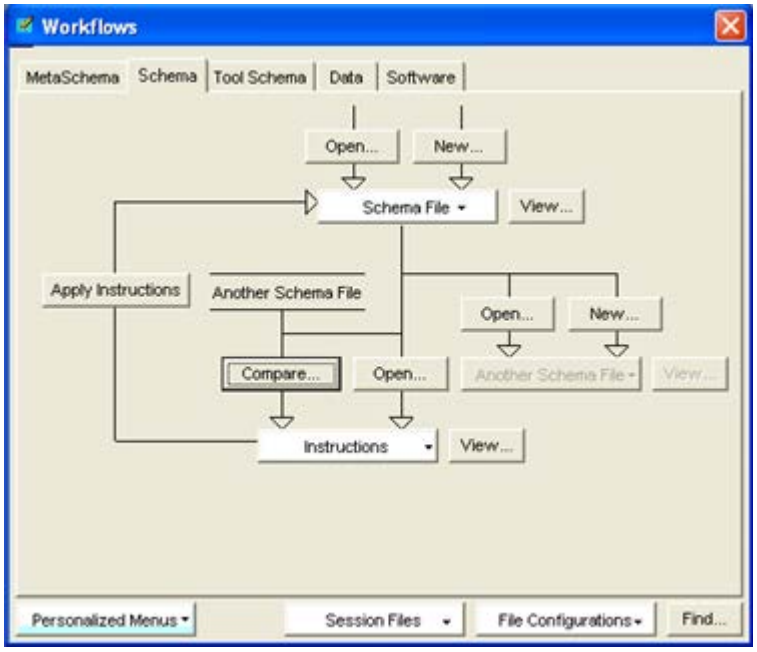
Other Workflows Dialog Boxes

In addition to the **Standard Workflows** window, there are other **Workflows** dialog boxes that are used less frequently in the Schema Editor. These **Workflows** dialog boxes are arranged using a visual representation of supported workflows in the Schema Editor. You can open these **Workflows** dialog boxes by clicking the appropriate command on the Workflows menu in the Schema Editor.

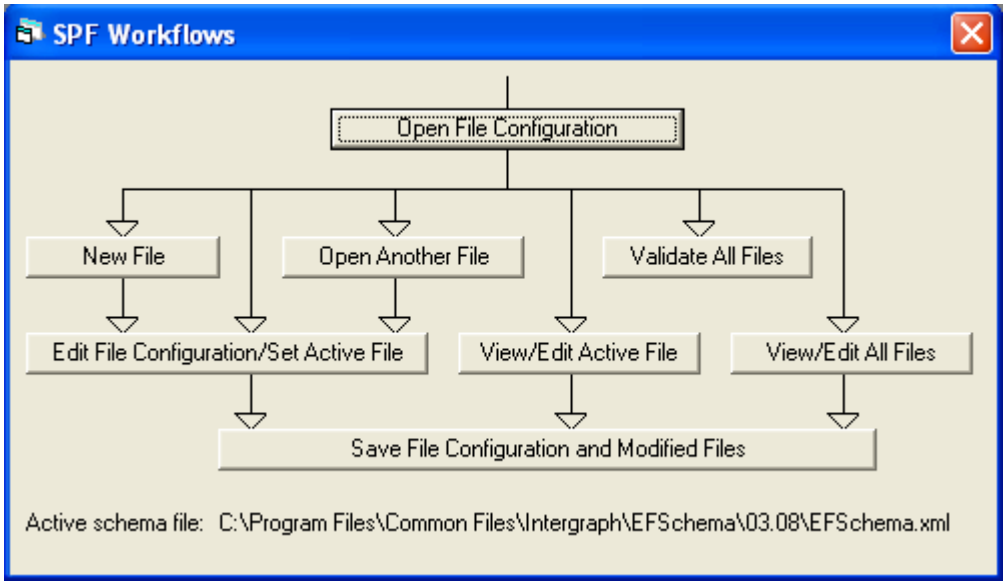
- **Basic Workflows** - Provides access to functionality that basic users of the Schema Editor need, including opening schemas and a tool map schema. After you open a file or files in the Schema Editor, you can click the **Actions** button for the file type you opened to access the basic commands that users need to interact with the file. For example, if you click **Actions** under **Schema**, you can define the PBS hierarchy in the schema and edit schema objects and tool metadata.



- **Overall Workflows** - Provides most of the commands available on the Schema Editor's menus in a workflow format. The **Workflows** dialog box provides tabs that allow you to isolate the type of functionality that you need. For example, all the commands for interacting with the schema appear on the **Schema** tab. The Overall Workflows dialog box is accessible when you open the Schema Editor from the **Schema Editor - Developer** link on the **Start** menu.

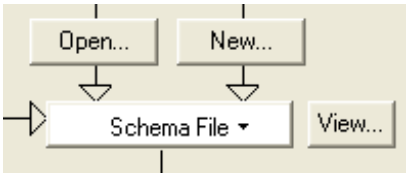


- **SPF Workflows** - Provides access to functionality that a user needs to edit the SmartPlant Foundation schema in a non-integrated environment. You can also view the object contents of a configuration and edit those contents. Once you have made your changes in Schema Editor, you can validate the schema, then save your configuration and modified files.



- The tabs in the **Workflows** dialog box may change depending on your personalized menu configuration.

When you use one of these workflows, you can use a series of **Open** buttons to open different types of files, such as schema files, tool map schema files, and data files. These **Open** buttons appear above the file type that you can open using that button. For example, **Open** buttons appear above the **Schema File**, **Tool Schema File**, and **Data File** buttons to allow you to open a schema file, tool map schema file, and data file from the **Overall Workflows** dialog box:



After you open a schema file, other buttons in the **Overall Workflows** dialog box become available to you. For example, when you open a schema file, you can then compare it with another schema or open a tool map schema file or data file.

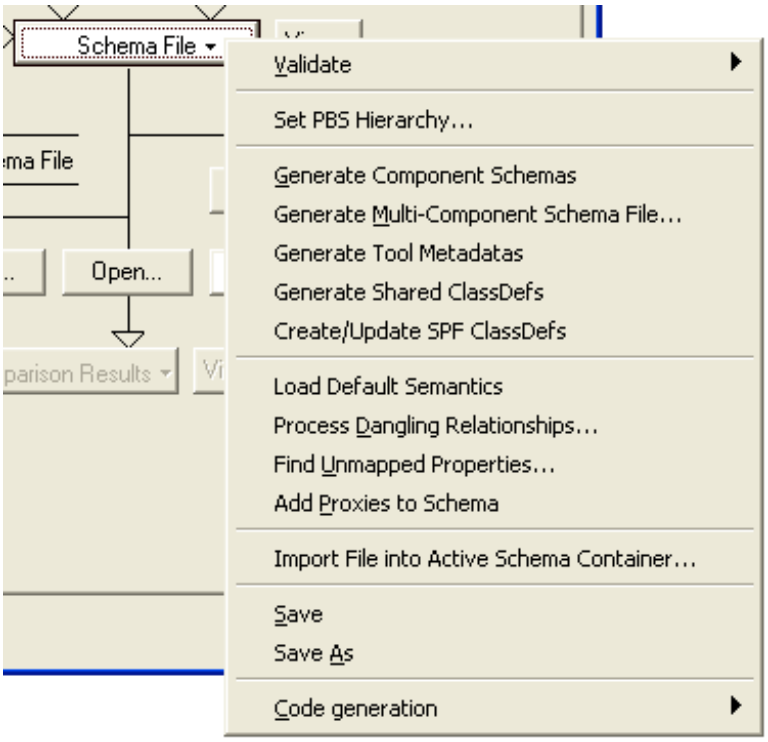
After you open a particular file type, the button for that type of data turns white. For example, if you open a schema, the **Schema File** button has a white background to indicate that you can interact with the schema file that you opened using the **Workflows** dialog box.



Some types of data are always available; therefore, the buttons that represent those data types always have light blue backgrounds. For example, when you open the Schema Editor, the meta schema and integration programming reference documentation are always available, which means that the **Meta Schema** button is always white and available.

Clicking the **View** button beside a button, such as **Meta Schema** or **Schema File**, displays the **View** dialog box for that type of data.

You can click buttons with white backgrounds to display a list of commands that are available for that type of data. These same commands are available on a submenu of the **Tools** menu. For example, if you click the **Schema File** button when a schema file is open, the software displays the following shortcut menu:



These commands are also available on the **Tools > Schema (Active)** submenu.

NOTE You can display or hide the **Workflows** dialog boxes using the **View > Workflows** command.

Personalized Menus

Personalized menus allow you to define the menu commands, startup options, find options, and view options that you use most in the Schema Editor. Personalized menus simplify the Schema Editor user interface so that you can access the commands that you need more quickly. You can also use personalized menus to define which **Workflows** dialog box or window that you want to see when you open the Schema Editor, which file configuration that you want to work with, what the default find and view options are, which edge definitions you see when you view files in the Schema Editor, and so on.

For example, if you only use the Schema Editor for mapping between an authoring tool and the SmartPlant schema, you can hide other commands and functionality that you do not use. You can save your preferences as a personalized menu configuration file.

Personalized menu configuration files are also used to define the default Schema Editor roles available on the **Start** menu. For more information, see *Schema Editor Roles* (on page 64).

If more than one person will be using the same personalized menu configuration file on a computer, you can set a user environment variable to store the configuration file. Users can

make additional modifications to the personalized menu configuration file without affecting other users.

See Also

Schema Editor User Interface (on page 48)
Create a Personalized Menu Configuration (on page 65)
Define the Storage Location for Your Personalized Menu Configurations (on page 67)
Modify an Existing Personalized Menu Configuration (on page 68)
Open a Personalized Menu Configuration File (on page 67)
Schema Editor Roles (on page 64)

Schema Editor Roles

Saved personalized menu configurations are used on the **Start** menu to limit the Schema Editor functionality that is available to users. When you install the Schema Component, the setup creates four Schema Editor links on the **Start** menu in the **Intergraph SmartPlant Foundation > Schema Component** folder.

Each link on the **Start** menu represents a particular type of Schema Editor user, or role, and opens the Schema Editor with the functionality that user needs most. Users can modify the personalized menu configurations to add or remove functionality.

The following default roles are delivered with the Schema Editor:

- **Schema Editor** - **Standard** – Provides access to standard Schema Editor functionality used by project implementation teams and customers who need to extend the SmartPlant schema and define mapping between authoring tools and the SmartPlant schema. This role allows users to work with multiple tool map schemas and multiple schema files. For more information about the Standard role, see *Standard Workflows Window* (on page 49) and *Schema Editor - Standard Overview* (on page 55).
- **Schema Editor – Developer** – Provides access to all Schema Editor functionality. This role is typically used by power users. For more information, see *Other Workflows Dialog Boxes* (on page 60).

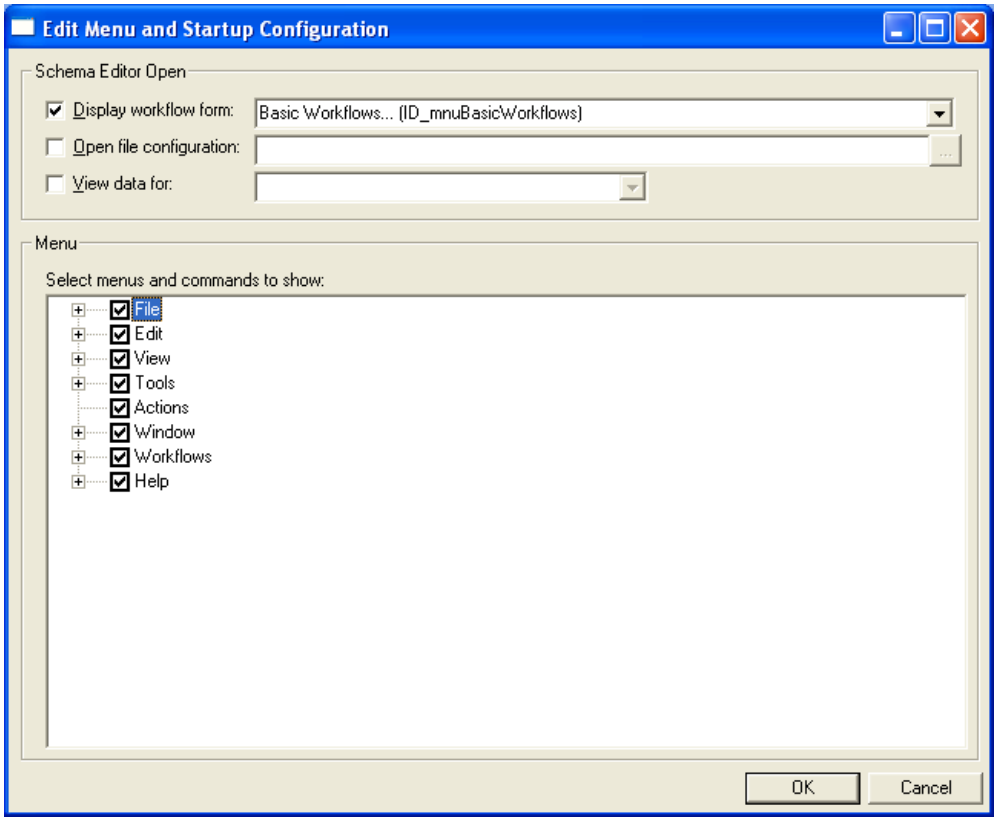
To see what menu commands and other functionality are available for a particular Schema Editor role, click the link for that role on the **Start** menu to open the Schema Editor. Then click **File > Personalized Menus > Edit Menu/Startup Configuration** or **File > Personalized Menus > Edit Find/View Options** to view the current settings for the role.

See Also

Schema Editor User Interface (on page 48)
Create a Personalized Menu Configuration (on page 65)
Define the Storage Location for Your Personalized Menu Configurations (on page 67)
Modify an Existing Personalized Menu Configuration (on page 68)
Open a Personalized Menu Configuration File (on page 67)
Personalized Menus (on page 63)

Create a Personalized Menu Configuration

1. To define the menu commands and startup options available in your personalized menu configuration, click **File > Personalized Menu > Edit Menu/Startup**.
2. If you want to display a **Workflows** dialog box when you open the Schema Editor, select the **Display workflow form** check box, and then select the name of the dialog box in the list



TIPS

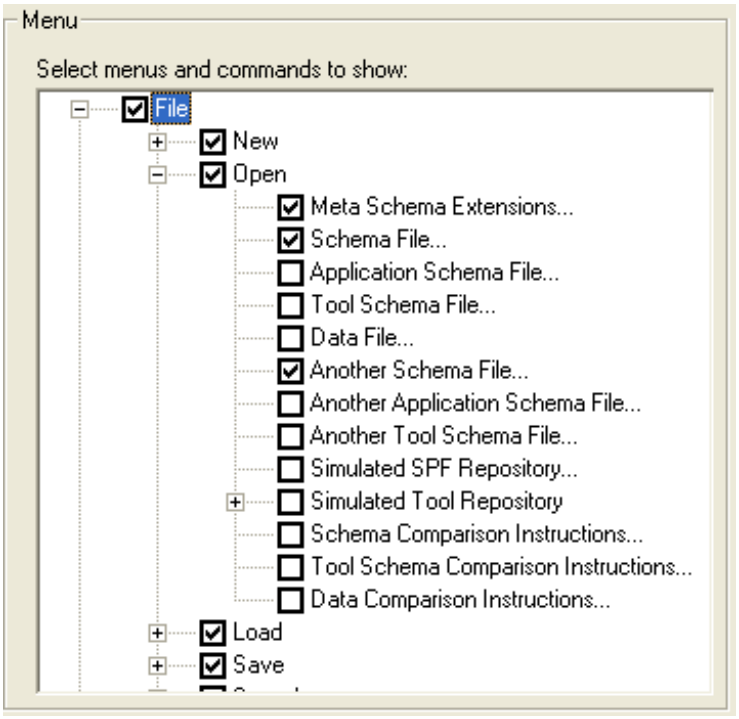
- The options under **Schema Editor Open** only apply to personalized menu files that are opened using the command line when the Schema Editor is opened. You can add a personalized menu file to your Schema Editor shortcut by right-clicking the shortcut in Windows Explorer and adding %TEFUser%\FileName.gui or the path to the personalized menu file after the path *to the S*chema Editor executable. For example, "C:\Program Files\Common Files\Intergraph\Schema Component\SchemaEditor.exe" %TEFUser%\MyMenus.gui or "C:\Program Files\Common Files\Intergraph\Schema Component\SchemaEditor.exe" C:\MyFiles\MyMenus.gui.

  The %TEFUser% option is only required if you use an environment variable to define the location of stored personalized menus. For more information, see *Define the Storage Location for Your Personalized Menu Configurations* (on page 67).

- If you selected **Overall Workflows** in the **Display workflow form** list, the tabs that are available in the **Workflows** dialog box in the Schema Editor change depending on the menu commands that you select on the **View** menu. For example, if you do not include

any of the tool map schema commands on the **View** menu, the **Tool Schema** tab does not appear in the **Workflows** dialog box.

3. If you want to work with a particular file configuration, select the **Open file configuration** check box, and then click ... to browse to the file configuration.
4. If you want to change the type of data that you can view in the Schema Editor, select **View data for**, and then select the type of data in the list.
5. Under **Select menus and commands to show**, select the check box beside the names of the menus and menu commands that you want to use in the Schema Editor.



NOTE Menus and menu commands that are not selected will still appear in the Schema Editor, but they will be unavailable to you.

6. Click **OK** to save your changes to the menu and startup configuration.
7. To define the default find and view options for your personalized menu configuration, click **File > Personalized Menu > Edit Find/View**.
8. On the **View** tab, select the type of view for which you want to define options.

   TIP The other tabs in the dialog box change depending on the type of view you select. For example, if you select a tree view, the **Tree Options** tab is available.
9. Define the view options that you want to use by default. For more information, see *Set View Options* (on page 123).
10. On the **Find** tab, define the default find options that you want to use.

    For example, if you want to search for particular types of objects by default, click **Only classes**, and then click ... to select the classes that you want to search for.
11. Click **OK** to save the find and view options you defined.

12. To save your settings as a personalized menu configuration, click **File > Personalized Menu > Save As Personalized Menu**.

See Also

Define the Storage Location for Your Personalized Menu Configurations (on page 67)
Modify an Existing Personalized Menu Configuration (on page 68)
Open a Personalized Menu Configuration File (on page 67)
Personalized Menus (on page 63)
Schema Editor Roles (on page 64)

Open a Personalized Menu Configuration File

1. Click **File > Personalized Menus > Open Personalized Menu**.
2. Browse to the .gui file that you want to open.
3. Click **Open**.

NOTES

- The display of the **Workflows** dialog box or window, your options for viewing files, your options for finding information, and the menu commands that are available to you may change when you open a personalized menu configuration file.
- By default, the delivered personalized menu configuration files that make up the default Schema Editor roles are delivered to the ..\Program Files\Common Files\Intergraph\Schema Component folder. You can change the default location of these files for each Schema Editor user on a specific computer. For more information, see *Define the Storage Location for Your Personalized Menu Configurations* (on page 67).

See Also

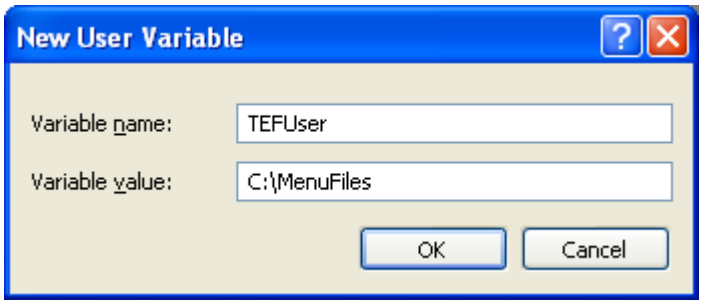
Create a Personalized Menu Configuration (on page 65)
Define the Storage Location for Your Personalized Menu Configurations (on page 67)
Modify an Existing Personalized Menu Configuration (on page 68)
Personalized Menus (on page 63)
Schema Editor Roles (on page 64)

Define the Storage Location for Your Personalized Menu Configurations

If multiple users use the Schema Editor on a single computer, you can define a unique location for storing personalized menu configurations for each user. This is especially important if multiple users want to use and modify the default roles delivered with the Schema Editor. Setting your own storage location allows you to use the different links on the **Start** menu to open a personalized menu file without requiring you to manually open the file in the Schema Editor.

1. In the Windows Control Panel, open System.
2. Click the **Advanced** tab in the **System Properties** dialog box.
3. Click **Environment Variables**.
4. Click **New** under **User variables**.
5. In the **Variable Name** box, type **TEFUser**.

6. In the **Variable Value** box, type the path to the directory where you want to store your personalized menu configuration files.



7. Click **OK**.

NOTE If you are the only user who uses the Schema Editor on your computer and you want to change the default storage location for personalized menu files, double-click **TEFUser** under **System variables**, and change the default path.

See Also

Create a Personalized Menu Configuration (on page 65)
Modify an Existing Personalized Menu Configuration (on page 68)
Open a Personalized Menu Configuration File (on page 67)
Personalized Menus (on page 63)
Schema Editor Roles (on page 64)

Modify an Existing Personalized Menu Configuration

1. Open the personalized menu configuration that you want to modify. For more information, see *Open a Personalized Menu Configuration File* (on page 67).
2. To modify the menu commands and startup options available in the personalized menu configuration, click **File > Personalized Menu > Edit Menu/Startup**.
3. Make changes in the **Edit Menu and Startup Configuration** window, and click **OK**.
4. To modify the default find and view options for your personalized menu configuration, click **File > Personalized Menu > Edit Find/View**.
5. Make changes in the **Options** dialog box, and click **OK**.

NOTE To see the changes to the Schema Editor user interface, you must close the Schema Editor and open the personalized menu configuration file.

See Also

Create a Personalized Menu Configuration (on page 65)
Define the Storage Location for Your Personalized Menu Configurations (on page 67)
Open a Personalized Menu Configuration File (on page 67)
Personalized Menus (on page 63)
Schema Editor Roles (on page 64)

Layouts

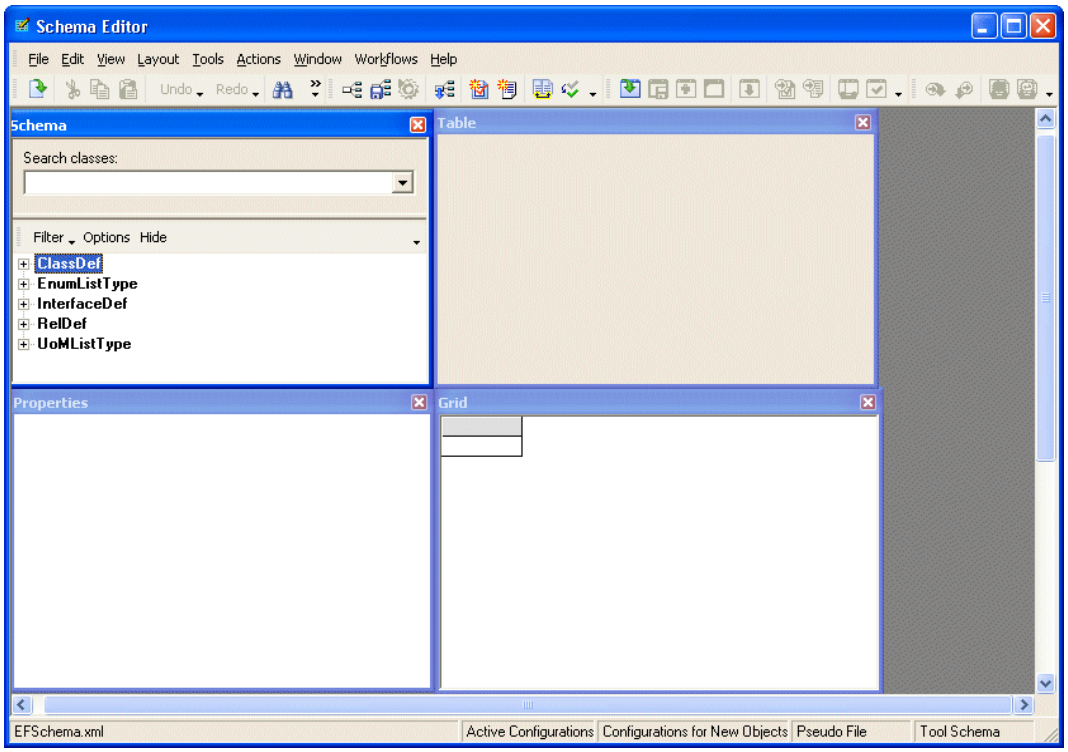
By default, Schema Editor opens with its user interface docked in a tabular layout (Schema Editor - Standard) or with the **Overall Workflows** dialog box open (Schema Editor - Developer). The new **Layout** menu allows you to build your own Schema Editor user interface by opening multiple intelligent controls that update automatically when you select an object on one of the forms.

Create a Layout

1. To define your window layouts, click **Layout** from the menu bar.

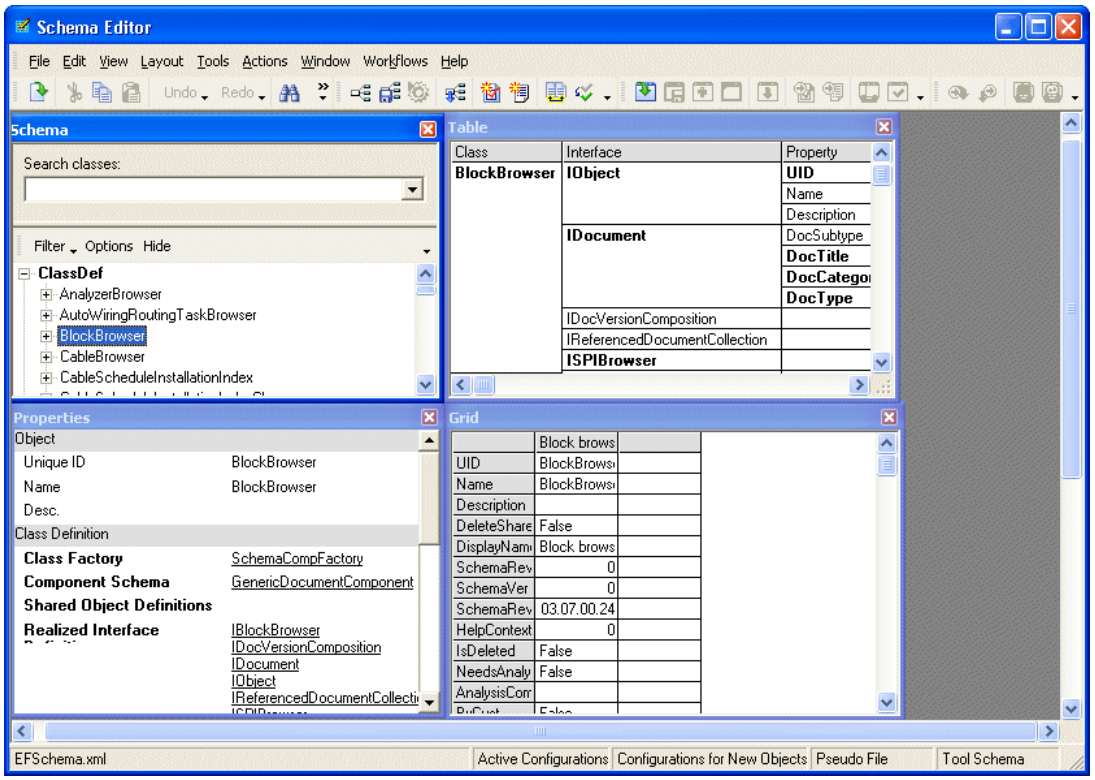
   NOTE To view the **Layout** menu in Schema Editor - Standard, click **File** > **Personalized Menu** > **Edit Menu/Startup**, then select the check box beside **Layout**.

2. Select the windows that you want to use in Schema Editor. For example, you can select **Schema**, **Tool Schema**, **Properties**, **Grid**, and **Table** as your default layout.



   NOTE The windows can be resized and repositioned.

3. Click an object in the **Schema** window or **Tool Schema** window. The information in the other windows is updated.



NOTES

- Selecting objects in windows other than **Schema** or **Tool Schema** only updates the information in that window. For example, selecting **IBlockBrowser** from the **Properties** window only updates the **Properties** window.
- Modifications made in any window updates the information in all windows. For example, changing the **Description** in the **Grid** window updates the information in all other windows.

4. To re-use the layout in Schema Editor, click **File** > **Session File** > **SaveAs Session File**.

   NOTE When you open the session file, only the position and starting data state of the layout is remembered. The last object open is not remembered.

See Also

Personalized Menus (on page 63)

Schema Editor Naming Rules

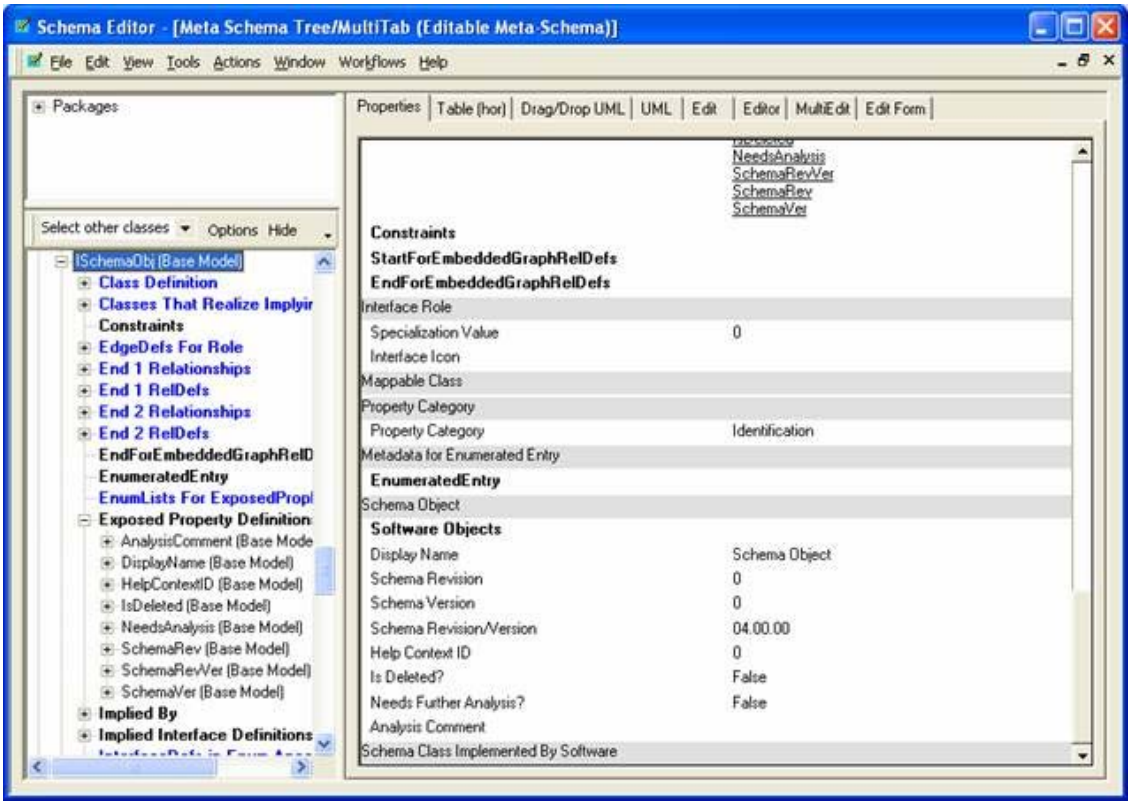
You can create name display rules from the **MetaSchema Extension** environment in the Schema Editor. Name display rules define how names of objects are displayed in the Schema Editor user interface. You can also create name display rules by double-clicking the **NameDisplayRule** node in any meta schema view in the Schema Editor.

Objects As Schema Versus Objects As Data

The Schema Editor can be used to view objects as data and to view objects (meta-schema and schema objects) as schema. When objects are viewed using the Schema Editor, those objects are viewed as data. However, under certain circumstances schema and meta-schema objects are presented as schema.

In the following example, a meta-schema object (ISchemaObj) displays as both data and schema. In the tree view, ISchemaObj displays as data. When displayed as data, properties of that object (as well as relationships (edges)) from that object are depicted. In the tree control, the relationships (edges) for ISchemaObj display.

In the **Properties** window, ISchemaObj is shown as a data object with both properties and relationships (edges) for ISchemaObj.



In the **Properties** windows, the text “Schema Object” displays with a dark background. Below it are a set of property names (“Display Name” through “Analysis Comment”). “Schema Object” is the DisplayName for ISchemaObj, “Display Name” is the DisplayName for DisplayName, “Analysis Comment” is the DisplayName for AnalysisComment and so on.

For this display (“Schema Object” through “Analysis Comment”), ISchemaObj display as schema. Note that the display of ISchemaObj as schema is different than the display of ISchemaObj as data (contrast the property names in the tree control (AnalysisComment through SchemaVer) with the corresponding names in the **Properties** control (for example, Analyis Comment).

When defining rules on how to display object names, you have the flexibility of defining whether the rule applies to display of objects as data objects or display as schema objects.

Applicable Name Display Rule

The meta-schema is seeded with two default name display rules. One rule applies when an object is viewed as data and one rule applies when an object is viewed as schema. If no other name display rules are defined, these two rules will be used to determine how to display object “names” throughout the Schema Editor.

You can define a default name display rule that overrides one of these meta-schema defined rules. In this case, the user-defined rule will supersede the meta-schema rule and will be applied to all objects. Note that two default rules, one for display as data and one for display as schema, would need to be defined to completely override the meta-schema defined rules.

You can define a name display rule that applies to a specific class definition. In this case, whenever an object of that particular class definition is displayed, it will be displayed using that specific rule.

Extending the MetaSchema with Name Display Rules

When determining the name display rule that is to be used, the following logic is used:

- Locate the class definition for the object being displayed.
- Get the applicable display rule for that class definition and whether the object is being displayed as data or as schema.
  - Locate an explicit rule for that class definition and display mode (“as data” versus “as schema”).
  - If no explicit rule is found, locate a user-defined default rule for that display mode.
  - If no user-defined rule is found, use the meta-schema defined rule for that display mode.

This process of locating the rule that applies is done by searching the container composition for the class definition for a match on either the explicit class definition or the user-supplied default rule. Since the container composition for the class definition is searched, name display rules that are defined as part of the schema will not be applied to the display of schema objects since the class definitions for the schema objects are part of the meta-schema and not part of the schema. Therefore, if name display rules are to be applied to the display of schema objects using the Schema Editor, these name display rules must be defined as meta-schema extensions and not as schema.

To define name display rules as meta schema, do the following:

1. Open the Schema Editor.
2. Click **File > New > Schema File** to create a new schema file (.xml extension).
3. Create the desired name display rules.
4. Exit and save the schema file.

5. Manually edit the .xml file and change the Scope property from Schema to MetaSchema.

After these steps are performed, start the Schema Editor, by selecting the **MetaSchema** tab and loading the XML file as the meta-schema extensions file.

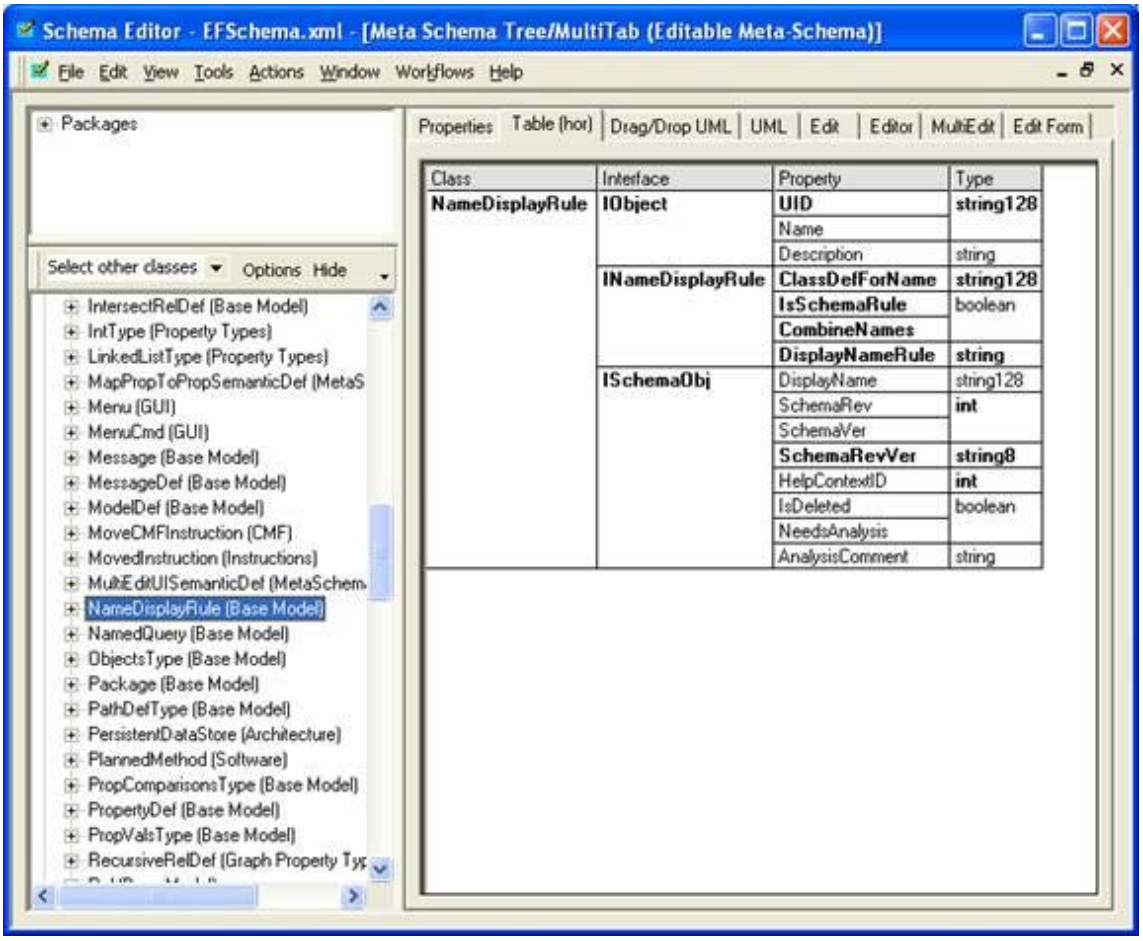
The Schema Editor is normally used to view and edit schema and map (tool) schema. For both the schema and the map schema, the class definitions for the displayed objects are part of the meta-schema and any name display rules that are defined must be done as meta-schema extensions to apply. Occasionally, the Schema Editor is used to view data files. For this case of viewing data files, the name display rules need to be defined as part of the schema rather than the meta-schema because the class definitions for the data objects are part of the schema not the metaschema.

The schema class definition PIDPipingComponent is an instance of ClassDef, which is a meta-schema class definition. Likewise, IPipingComponent is an instance of InterfaceDef (a meta-schema class definition) and NominalDiameter is an instance of PropertyDef (a meta-schema class definition). Therefore, to define a name display rule that impacts how the Schema Editor presents PIDPipingComponent, that rule must be added as a meta-schema extension to apply.

Valve V-100 is an instance of PIDPipingComponent (a schema class definition). Therefore, to define a name display rule that impacts how V-100 displays using the Schema Editor, that rule must be part of the schema (not a meta-schema extension).

New MetaSchema Objects for Name Display Rules

A new class definition NameDisplayRule has been added to the meta-schema to support the definition of name display rules.



In the example, NameDisplayRule includes IObject, INameDisplayRule and ISchemaObj. INameDisplayRule exposes the following property definitions.

- ClassDefForName (string128) – this is the explicit class definition for which the rule is being defined or “Default” (if the rule is to be applied for all cases where there is no explicit definition for the class definition).
- IsSchemaRule (boolean) – indicates whether the rule applies to displaying the object as schema (True) or as data (False).
- CombineNames (boolean) – indicates whether the displayed “name” for the object is made by combining the individual property values (delimited by “/”) or whether the first applicable property value is to be used (False).
- DisplayNameRule (string) – delimited string that identifies the property values to be used in the “name”.

DisplayNameRule Property Value

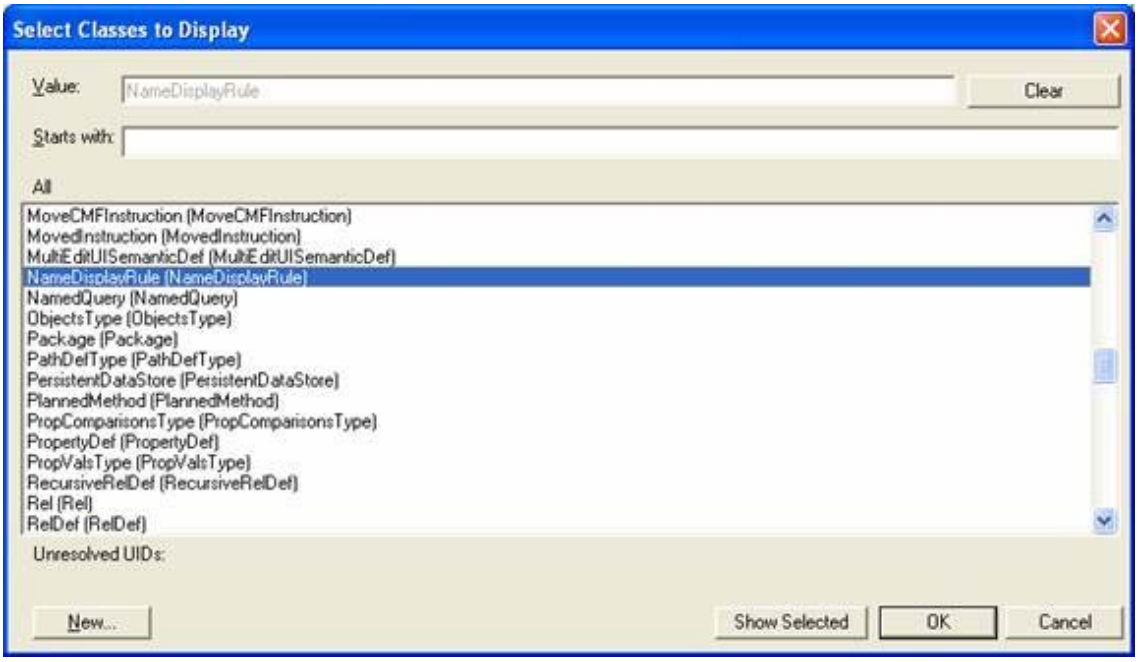
The DisplayNameRule property value is formed by delimiting the property values to be used for the name by commas. Therefore, if the name display rule involves three properties, the information for each of those properties will be separated by commas.

For each individual property, the following information is stored in the DisplayNameRule property value. Each piece of information is separated from the others using a tilde (~) delimited.

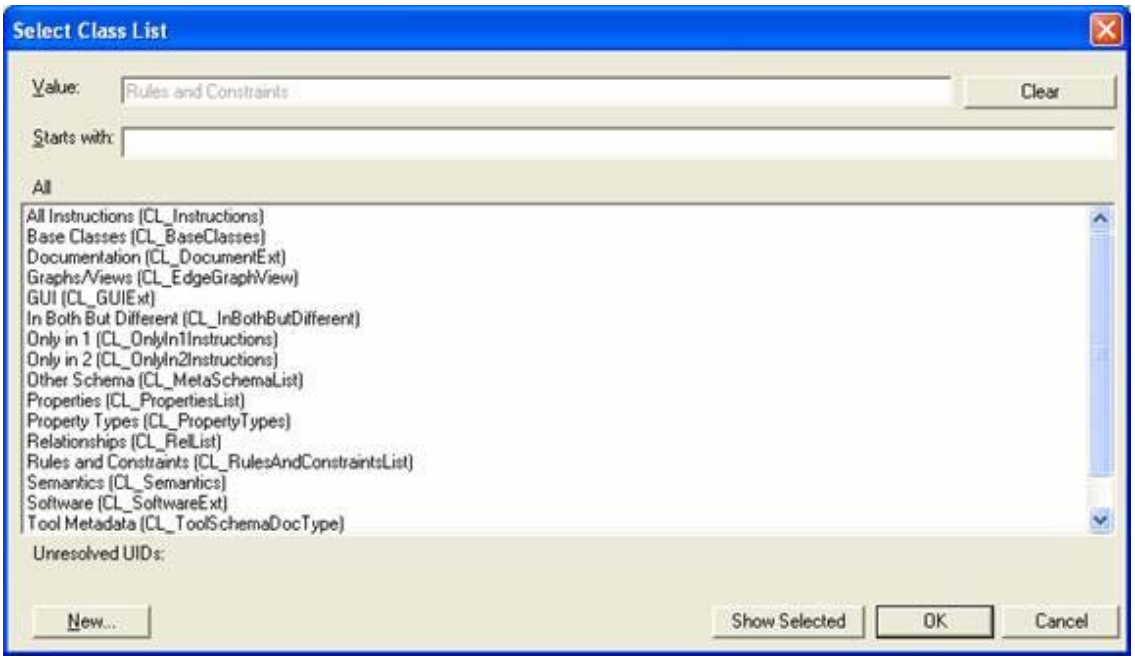
- Edge definition to traverse from the displayed object to get to the object with the property to be displayed. Normally this value is a null string.
- Interface definition containing the property definition.
- Property definition whose value is to be used for the “name”.
- Boolean flag (persisted as either “T” or “F”) indicating whether the property value is to be included in the name display even if that property value is a null string. This flag only applies if the rule indicates that the property values are to be combined to form the “name” (such as, CombineNames is True).

GUI for Viewing and Defining Name Display Rules

Since a name display rule is an instance of the NameDisplayRule class definition, one way to view name display rules is to select the NameDisplayRule class definition from the **Select Classes to Display** dialog box.



Alternatively, the **Rules and Constraints** entry from the class list selection dialog box can be used to view the name display rules.



The edit for NameDisplayRule objects is shown below.

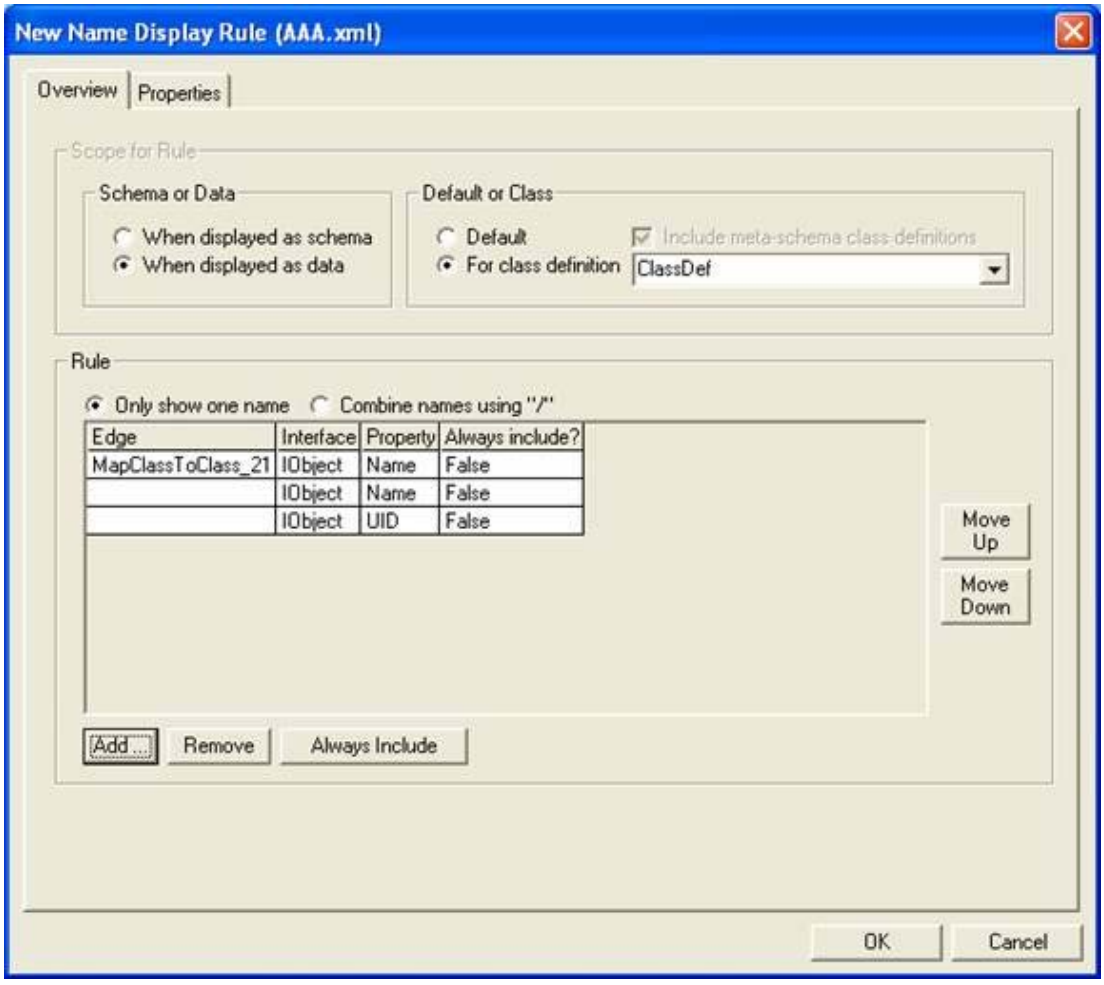
In the **New Name Display Rule** dialog box, the **Scope for Rule** window determines the scope of applicability from the rule. In this particular example, the scope has been defined to be for class definition ClassDef when displayed as data.

The name display rule for the specified scope is defined in the **Rule** area. The options at the top of this area indicate whether the first defined property value is to be used or whether the property values are to be combined to form the “name”.

The list of properties to be included in the “name” displays in the grid control. If only one property value is to be included in the “name”, then the first of these that apply for a particular object will be displayed as the “name”. For example, if there exists a MapClassToClass relationship for the ClassDef object (at least one tool schema has been opened and loaded) and Name is defined for that map class, then that value will be used as the “name”. If no MapClassToClass relationship exists or the map class does not have a Name value, then the rule identifies that the Name for the class definition object. Finally, if Name is not be defined for the class definition, then the UID is used.

- Select **Move Up** and **Move Down** to reorder the properties in the rule.
- Click **Remove** to remove a property from the rule.

- Click **Always Include** to toggle between always including the property in the “name” versus only including the property if it has a non-null value. This Boolean value only applies if the rule says that the property values are to be combined to form the displayed “name”. Note that the **Always Include** text will change to **Include If Defined** if the selected property has a True value for this flag.



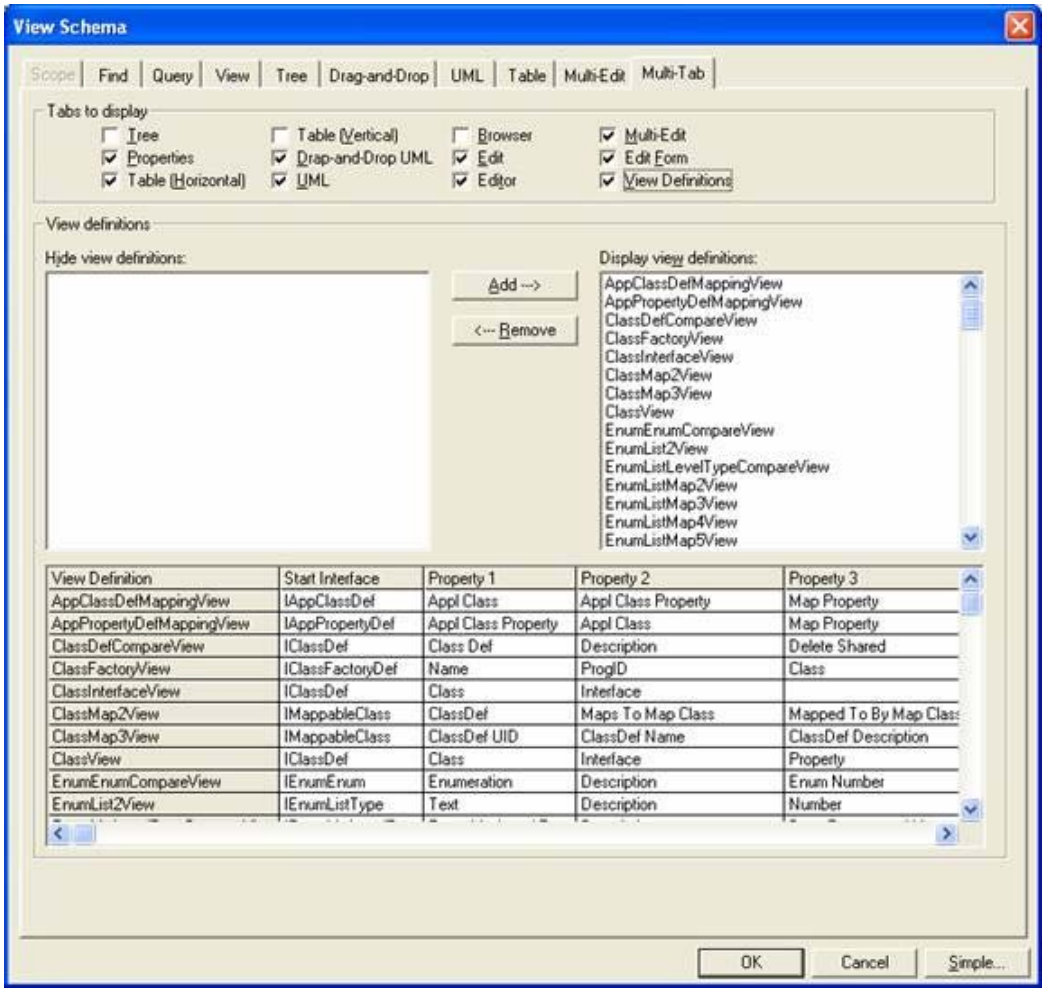
- Click **Add** to add a property to the rule. The **Select Property to Add to Rule** dialog box displays. Expanding the tree for this interface definition shows the properties for that role (as well as the edges for that role).



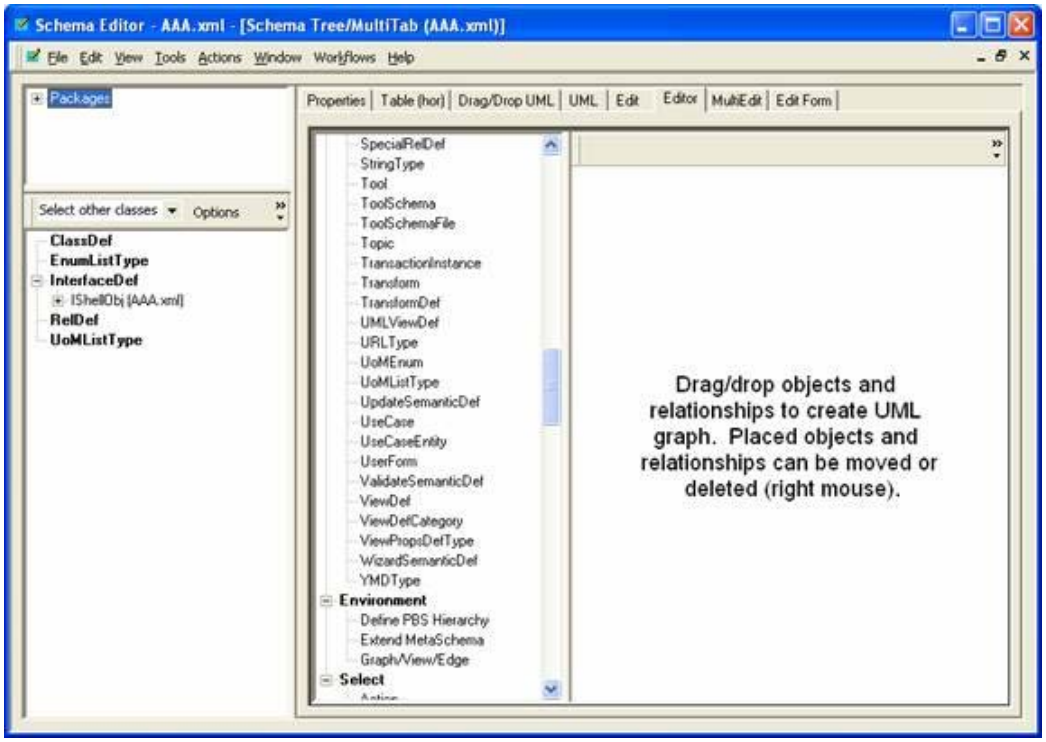
Define Naming Rules for Objects

1. Create a new schema file or open an existing schema file. The schema file should only contain meta-schema extension objects (no other schema objects).
2. View the schema file.

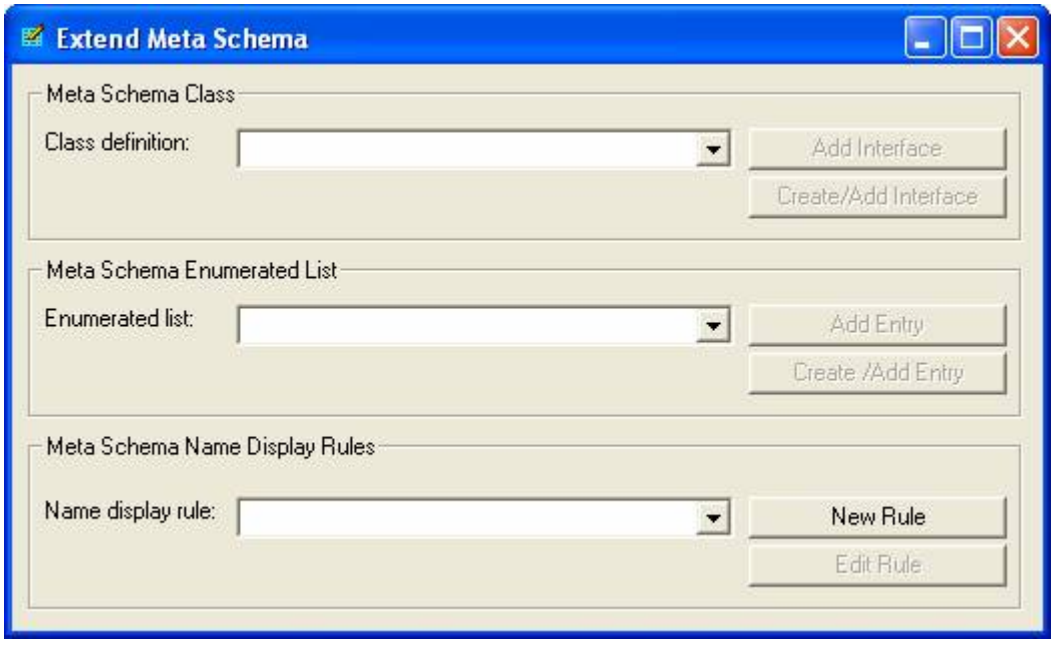
3. Select the **Editor** check box on the **View Schema** dialog box.



4. Click the **Editor** tab and scroll down to the **Environment** tree.

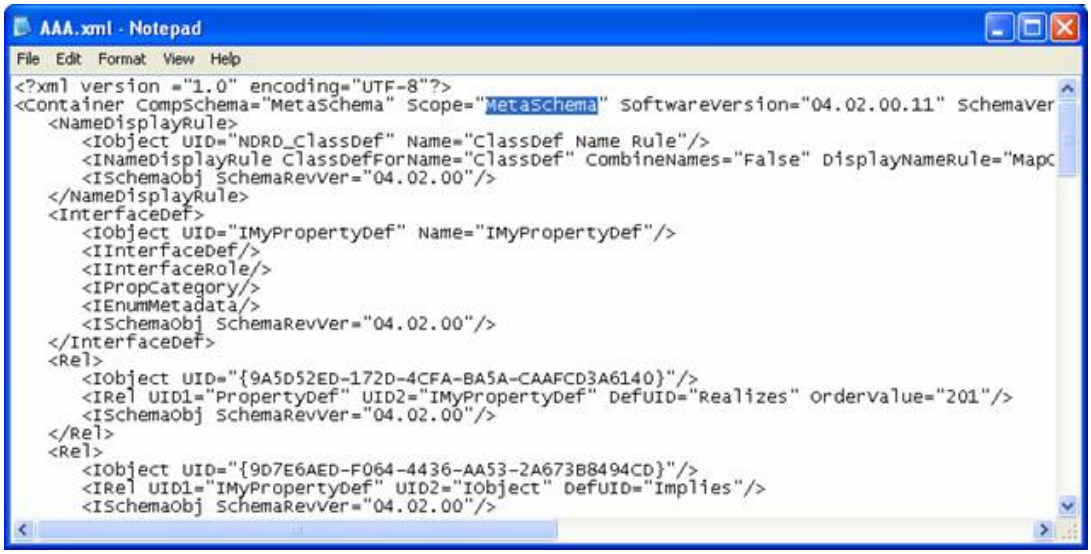


5. Select the **Extend MetaSchema** environment.



6. Use the **Extend Meta Schema** dialog box to create and edit name display rules. The rules control the "name" used by the Schema Editor to display meta-schema and schema objects.

7. If a meta-schema extensions file is opened and the session saved, the session file that is created will contain the meta-schema extensions file. The same is true if a configuration file is created. If this session file is opened, the meta-schema extensions file automatically load.
8. Click **New Rule** to create a new name display rule (instance of NameDisplayRule). The **Edit** dialog box displays.
9. Define the new name display rule. Existing name display rules can be edited by first selecting the name display rule from the combo box drop-down list and then selecting the **Edit Rule** button.
10. When the extensions to the meta-schema have been completed, the schema file needs to be converted to a meta-schema file. To convert the schema file to a meta-schema file, open the schema file in Notepad or another text editor.
11. Change the Scope entry on the first line from “Schema” to “MetaSchema”. If the need arises to further extend the meta-schema, then this same process can be done in reverse and the Scope changed back to “Schema”.



12. Click **File > Open > Meta Schema Extensions**.

SECTION 5

Opening and Using Files in the Schema Editor

In the Schema Editor, there are several different methods that you can use to open schema files and begin work. You can

- Check out the Schema configuration managed file (CMF) from the Desktop Client and launch the Schema Editor using a shortcut menu command. This uses a session file to save the selected CMF file, loaded tool schemas, and user interface settings. To modify the schema or tool mapping in an integrated environment, you should typically use this method of opening files and connecting to the Schema Editor. For more information, see *Configuration Managed Files (CMFs)* (on page 82).
- Open the Schema Editor and then open a session file that defines the loaded files and user interface settings. If you don't need to manage the SmartPlant schema using SmartPlant Foundation document management, you are still able to save your settings to a session file on exit from Schema Editor and open the session file to continue work. For more information, see *Session Files* (on page 92).
- Open the Schema Editor and then open a configuration file. If you are modifying the data model for stand-alone SmartPlant Foundation, a configuration file that references the necessary schemas and your schema extensions file is recommended for schema editing. For more information, see *File Configurations* (on page 93).
- Open the Schema Editor and use the **File > Open** commands to open schemas, tool schemas, and data files. For more information, see *Opening Files* (on page 96).

Configuration Managed Files (CMFs)

A configuration is a named collection of schema objects typically grouped by schema release version. Configuration managed files (CMFs) allow the Schema Editor to manage multiple schema configurations at one time to support forward and backward compatibility of schema versions.

Integration supports the SmartPlant authoring tools' use of different versions of the SmartPlant schema from the version used in SmartPlant Foundation. Integrated SmartPlant tool versions 2007 Service Pack 4 and later can be integrated with SmartPlant Foundation 2008 and higher. Integration software automatically handles required transformations of data for tools. When the authoring tools release new versions, these will also be compatible with SmartPlant Foundation 2008 and higher.

CMF files make this possible by storing identical objects across configurations only once and storing the differences in different configuration blocks. User customizations to the schema and metaschema information are also stored in their own configuration blocks.

CMF files allow you to

- Make changes that apply to only the version 2007 schema (for a 2007 tool).
- Make changes that apply to only the 2016 schema (for the latest versions of tools).

- Make changes that apply to both 2007 and 2016 schemas (for data exchanged between the 2007 tool and the newer tool).
- And so on for multiple versions of the schema and authoring tools in the future.

However, if you publish new properties that are only available in a newer version of the SmartPlant schema to an older schema version supported by an older tool, the new properties that you add in the newer version of the schema will not be supported.

CMF files have the following features:

- Every configuration in the CMF file is available for viewing. For more information, see *Viewable Configurations* (on page 84).
- Every configuration is available for editing. For more information, see *Editable Configurations* (on page 84).
- Multiple configurations can be worked on simultaneously. For more information, see *Multiple Editable Configurations* (on page 84).
- Single active configuration can be viewed but new objects that are created go into multiple configurations. For more information, see *Single Active Configuration with New Objects in Multiple Configurations* (on page 84).
- Pseudo files allow logical separation of objects within configurations and support a variety of options (for example, controlling whether the objects are loaded into memory). For more information, see *Pseudo Files* (on page 85).
- A variety of commands are available specifically for editing individual or multiple configurations. For more information, see *Special Configuration Commands* (on page 86).

CMF files can be opened in the Schema Editor, but their most common use is managing the schema configuration for integration in the SmartPlant Foundation Desktop Client. When you manage your schema configurations this way, the following work process is required:

- Check out the SmartPlant schema CMF file from the SmartPlant Foundation Desktop Client.
- Make sure your scope is set to the plant, right-click the new revision of the CMF file in SmartPlant Foundation, and click **Launch Schema Editor**. This provides the Schema Editor with the appropriate plant and SmartPlant Foundation site for connection.
- Select the schema version or versions that you want to view in the **Specify configurations to display** window of the **Active Configurations** dialog box.
- Click the **Advanced** button on the **Active Configurations** dialog box to view the **Set Configurations for new objects** window, where you can specify what version or versions of the schema changes should be written to. For example, to modify the 2016 schema, select the 2016 configuration for new objects.

  NOTE By default, all configurations are checked to allow forward and backward compatibility between schema versions when creating new objects or modifying existing objects.

See Also

Synchronize Tool Metadata with the Tool Map Schema (on page 328)

Viewable Configurations

You can view all of the available (non-obsolete) configurations. Any available configuration within the CMF file can be made the active configuration. You can view and navigate the objects, similar to a tool such as SourceSafe or Team Foundation Server.

Editable Configurations

You can view the object contents of a configuration and edit those contents. When you make a configuration active, you can view and edit all of the objects in that configuration. You can modify all non-obsolete configurations in the CMF file as desired.

Multiple Editable Configurations

A unique characteristic of CMF files is the ability to specify multiple configurations to be active at the same time. A particular change or addition can be made once and applied to multiple configurations, which is useful when working in an environment in which more than one configuration is actively used.

For example, an environment uses the 2007 version of SmartPlant P&ID in conjunction with the latest version of SmartPlant Instrumentation, and you want to add a property that is to be transferred between these two applications working against different configurations. In order to create the property, you make both the 2007 and 2016 configurations active at the same time and add the property. The property is then available in both configurations.

Single Active Configuration with New Objects in Multiple Configurations

While having multiple active configurations allows for new objects to apply to all of those configurations, the Schema Editor also supports having a single active configuration but having new objects apply to multiple configurations. One advantage of working against a single configuration but having new objects apply to multiple configurations is that all objects in the active configuration are viewable and editable. Working against multiple configurations only includes those objects that are identical across all configurations.

You can view and manipulate only those objects that are identical across all of the selected configurations when multiple configurations are active. The set of objects that are displayed and manipulated are exactly the same across the set of configurations. Only those objects that are identical are available when working against multiple configurations, which may cause dangling relationships (where the object at the other end is not identical for all of the configurations) or missing relationships (where the relationship only exists in some of the configurations or is different between the configurations). If multiple configurations are made active, it is unlikely that all of the objects for any of the configurations will be available.

Commands such as **Validate** do not work against multiple configurations because of the dangling relationships and missing objects and relationships. Working against a single configuration but having new objects added to multiple configurations allows all the commands to function.

A potential problem with adding new objects to multiple configurations while working against a single active configuration is that the new objects might cause validation problems for the other non-active configurations. For example, if a new object and a new relationship are created in

this environment and the existing object to which that new relationship is related does not exist within all of the configurations then validation of the other configurations generates dangling relationship messages.

However, working against a single configuration while adding new objects to multiple configurations is a useful environment for extending the SmartPlant schema.

Pseudo Files

Pseudo files allow for the objects within the configurations to be logically organized into files. Pseudo file functionality is also useful for defining behavioral characteristics for those objects.

The following set of characteristics can be defined against a pseudo file and apply to all the objects within that pseudo file:

Characteristics	**Features**
Path	Allows for the contents to be persisted to a file in another location.
SuppressOutput	Specifies that the contents of the pseudo-file are not to be included in a SaveAs file (XML).
SuppressLoad	Specifies that the contents of the pseudo-file are not to be loaded into memory.
ReadOnly	Specifies that the contents of the pseudo-file cannot be modified.
ReadOnlyFile	Indicates that the file for the pseudo-file is a read-only file. Similar to ReadOnly. Unlike ReadOnly, this value cannot be modified interactively and can only be changed by making the file writable (requires exiting and reopening the Schema Editor).
IsObsolete	Indicates the pseudo-file cannot be the active pseudo-file; therefore, new objects cannot be added to the pseudo-file.
WriteDisabled	Indicates that the file cannot be changed interactively. Similar to ReadOnly. Unlike ReadOnlyFile, WriteDisable has nothing to do with whether there is an external file for the pseudo-file and whether that file is itself read-only.
Level	Specifies the level for the pseudo-file. Level is used to define authoring objects (level 4) and is also used to define whether a pseudo-file can be missing (useful in development environment).

Characteristics	Features
Application	Specifies the application for the pseudo-file. If this application is not specified to be included as part of the **Set Active Configuration** command, then the objects in this pseudo-file are not be parsed into memory.
SuppressOutputGroup	Indicates the group to which the pseudo-file belongs. If this group is flagged to be suppressed at the CMF level, then the objects in this pseudo-file are not included in the file created using a **SaveAs** command.

NOTES

- Pseudo-file properties support a variety of workflows. For example, the properties that govern output suppression allow control over what is viewable versus what is included in a generated (**SaveAs**) file in a development environment.
- The pseudo-file properties support workflows for which inadvertent additions or modifications of the existing objects within the pseudo-file can be prevented.
- The SuppressLoad property allows for completely hiding those objects even from viewing.
- The IsObsolete and the control modification properties, such as ReadOnly, ReadOnlyFile, and WriteDisable, are used to control whether objects can be added to the pseudo-file (IsObsolete) or whether existing objects can be modified or deleted (ReadOnly, ReadOnlyFile, and WriteDisable).

See Also

Create a New Pseudo File in the Schema Editor (on page 193)
Merge a Previously Generated Schema into a Pseudo File (on page 195)
Convert an Existing Pseudo File into a Physical Pseudo File (on page 197)

Special Configuration Commands

The following commands are available specifically for CMF configurations:

Set Active Configurations – Defines the configuration or configurations that are to be active and into which new objects are added. Also defines the active pseudo-file for new objects, the set of applications that are to be included, and whether the authoring objects for those applications are to be loaded.

Set Level to Ignore Missing Files – Defines the level at which missing pseudo-files are acceptable.

Create Configuration – Creates a new primary configuration or to create a new pseudo-file configuration.

Edit Configurations – Displays information about all primary configurations and all pseudo-file configurations and allows the editing of this information.

Edit Pseudo-Files – Allows the setting of certain property values for the pseudo-file configurations.

Set Groups to be Suppressed – Defines the **SuppressOutputGroups** that are to be suppressed for a command that generates a new file (for example, **SaveAs**).

Merge File into Active Configuration – Merges the contents of an XML file into the active configurations.

Detect Duplicate UIDs – Locates objects across configurations that have duplicate UIDs.

Merge Identical Objects – Used to locate and merge objects that were independently created but identical across configurations.

Merge Configuration into Configurations – Adds objects that do not exist in the active configuration but do exist in the specified configuration to the active configuration.

Rename Configuration – Allows the renaming of a primary configuration or a pseudo-file configuration.

Move Between Pseudo Files – Moves objects from one pseudo-file to another pseudo-file.

Compare Configurations – Compares the active configuration with another configuration in the same CMF file.

Compare Configurations and Selectively Update Active Configuration – Compares the active configuration with another configuration in the same CMF file. The differences between the two configurations can be used to selectively update the active configuration.

NOTES

- Some of these commands are used to set properties for the configurations and are a fundamental part of those workflows that require the modification of one of these property values.
- The **Detect Duplicate UIDs** and **Merge Identical Objects** commands are used to address unintentional differences that are between objects in different configurations.
- The **Compare Configurations** and **Compare Configurations and Selectively Update Active Configuration** commands are useful in comparing two configurations and in fixing the active configuration based on the compared-with configuration.
- The **Merge Configuration into Configurations** command compares and automatically adds (no selectivity) objects from the specified configuration into the active configurations.
- The **Move Between Pseudo Files** command handles cases where objects are in the incorrect pseudo-file and need to be moved to the correct pseudo-file.
- The **Merge File into Active Configuration** command is useful for importing objects defined within an XML file into active configurations.

Allow Multiple Users to Edit the Same CMF File

If multiple users need to simultaneously edit the same schema CMF file, the following procedure should be followed:

- Check the CMF file out of SmartPlant Foundation. The CMF file that is checked out should be considered the master version.
- Copy the CMF file as many times as needed and distribute to other users/machines.
- When a user is ready to check in changes, follow the procedure described in Merging Individual Changes into the Master CMF File.
- When all users have merged in their changes, check the master CMF file into SmartPlant Foundation.

  NOTE Perform interim check in operations as often as you wish.

Merging Individual Changes into the Master CMF File

CAUTION The following procedure, if not executed correctly, can have undesirable results. Therefore, it is strongly urged that the individual doing the merges follow this procedure very carefully.

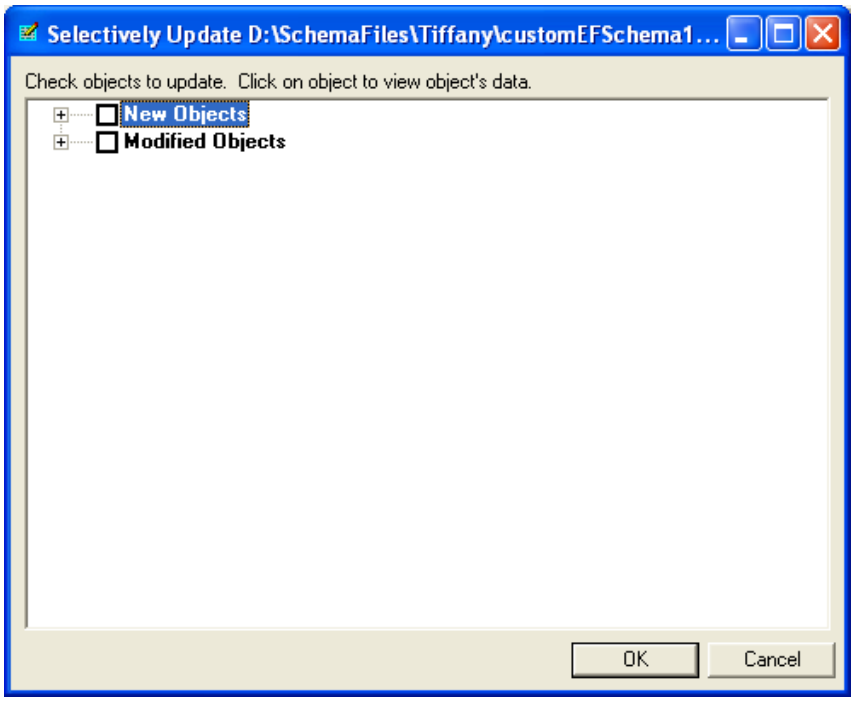
1. Before merging a CMF file into the master, each configuration that was modified in that CMF file should be validated and all problems (all errors, some of the warnings) detected during validation be resolved. Failure to validate the configurations in this CMF file will result in those validation problems being passed on to the master CMF file.
2. After all modified configurations have been successfully validated (such as, no problems detected during validation), the CMF file is ready to be merged.
3. Using the Schema Editor, open the master CMF file and set the active configuration to the first (next) configuration to be updated from the other CMF file
4. Click **File > Compare with File > Schema File > Schema Configuration in Another File > Selectively Update Active Configuration**.

   Or, click **Compare on the Overall Workflows** dialog box, then click **Schema Configuration** in **Another File > Selectively Update Active Configuration** from the shortcut menu.
5. In the **Open CMF File for Comparison** dialog box, select the other CMF file (such as, the one that is not the master) from which to update.
6. In the **Set Comparison Configuration** dialog box, select the configuration in the other CMF file that should be used.

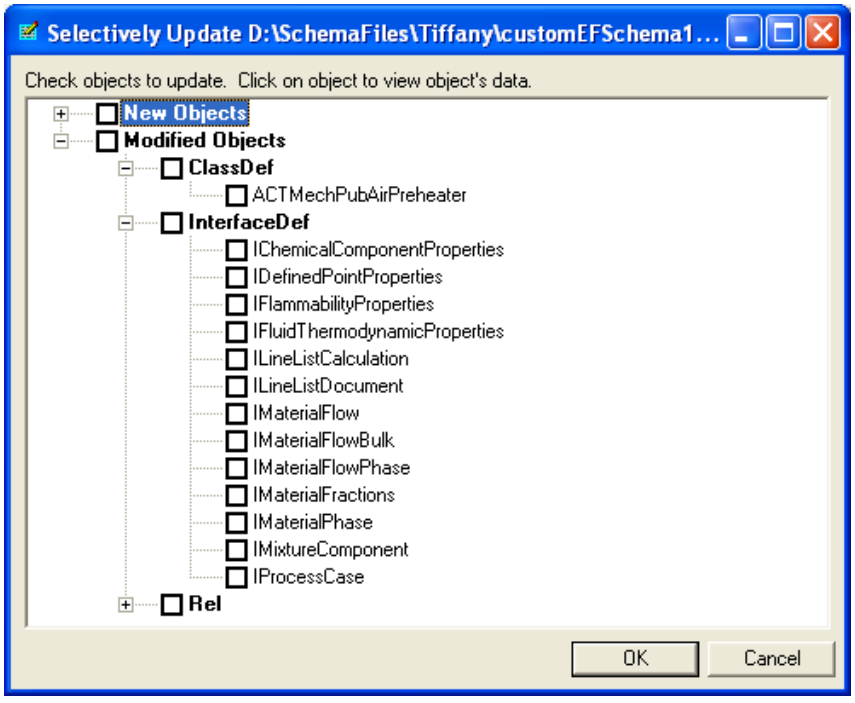
   CAUTION Be careful to pick the same configuration as the active configuration that was used for the master CMF file.
7. Select **OK** to begin the comparison

   After the comparison is performed, the differences are displayed in the **Selectively Update** dialog box. In the tree view, the differences are grouped as New Objects, Modified Objects, and Deleted Objects.

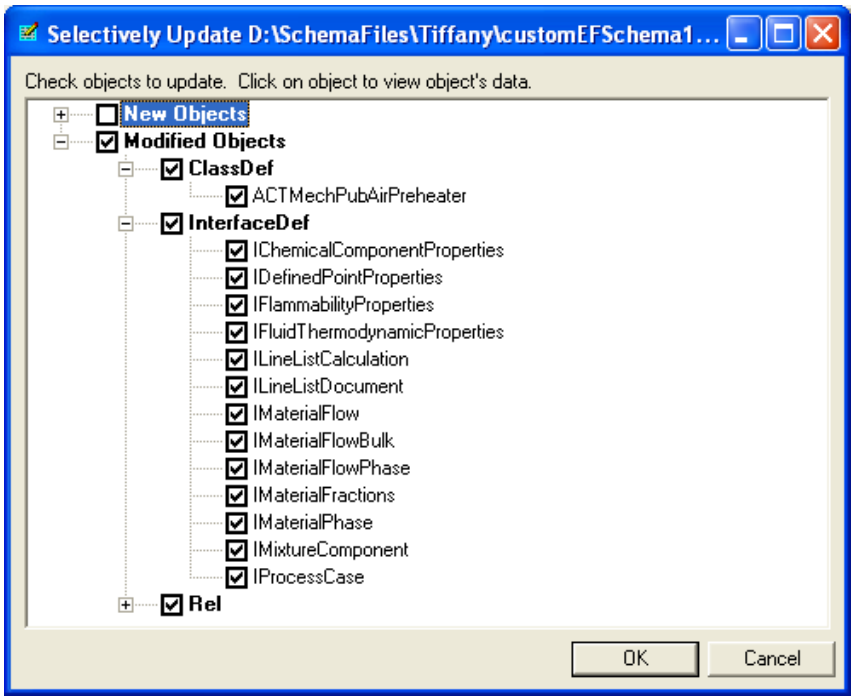
8. Check the object groups to update. By default, none of the nodes are checked and simply selecting **OK** before checking any of the boxes will result in no updates



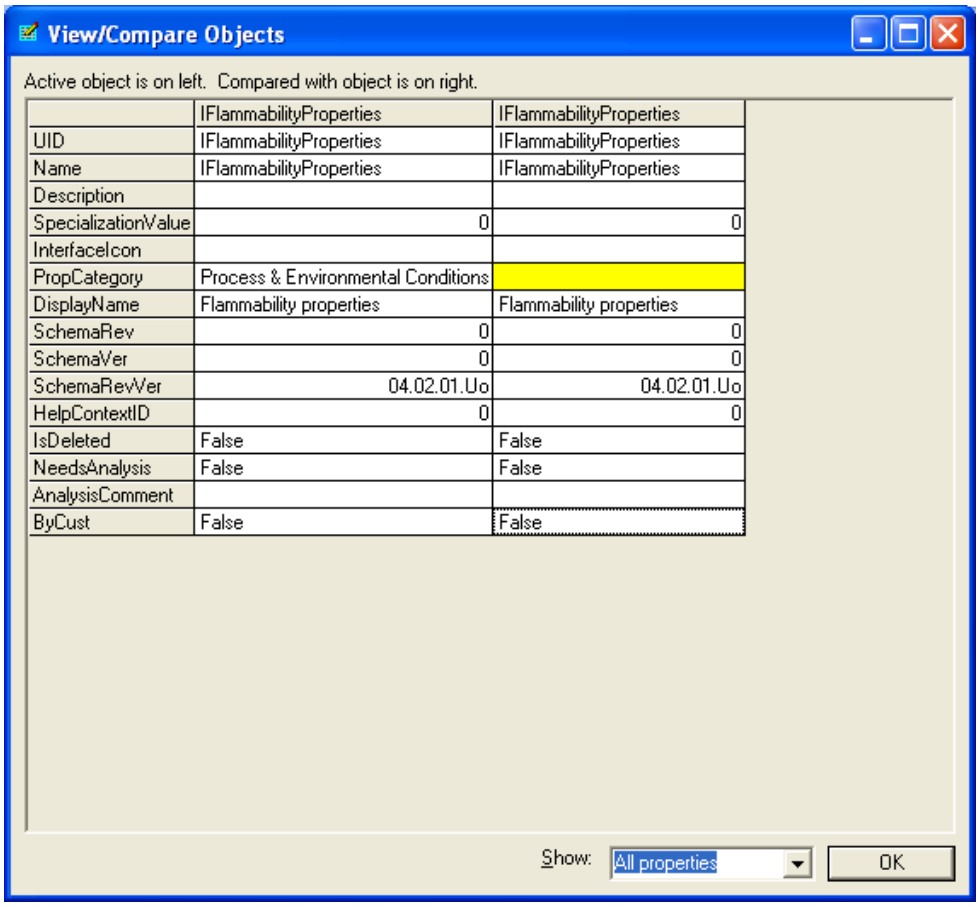
9. Expand the nodes to view more information about the objects that were added, modified and deleted.



Checking a node in the tree will check that parent node and all children nodes. For example, checking **Modified Objects** checks all the nodes for all the modified objects. Likewise, clearing a node will clear all the children nodes.



To view information about an object in the list, click the node for that object. For a new or deleted object, the edit dialog box for that object appears. For a modified object, the **View/Compare Objects** dialog box displays the differences in the object.



	IFlammabilityProperties	IFlammabilityProperties
UID	IFlammabilityProperties	IFlammabilityProperties
Name	IFlammabilityProperties	IFlammabilityProperties
Description		
SpecializationValue	0	0
InterfaceIcon		
PropCategory	Process & Environmental Conditions	
DisplayName	Flammability properties	Flammability properties
SchemaRev	0	0
SchemaVer	0	0
SchemaRevVer	04.02.01.Uo	04.02.01.Uo
HelpContextID	0	0
IsDeleted	False	False
NeedsAnalysis	False	False
AnalysisComment		
ByCust	False	False

NOTES

- In almost all cases, the **New Objects** node should be checked, indicating that all objects and relationships created within the other CMF file should be copied to the master CMF file
- For **Modified Objects**, the user should go through each object individually and click it to see the differences (form above). The user should then make a decision for that object whether the differences on the right should replace the object on the left. If the decision is yes, then that node should be checked. If the decision is no, then that node should be left unchecked. Unfortunately, there is no support for taking some of the changes.
- In general, none of the **Deleted Objects** should be checked. However, the user may want to go through the nodes for the individual deleted objects and decide that some of them actually should be deleted.

10. When all changes are identified, click **OK** on the **Selective Update** dialog box to update the active configuration in the master CMF file with the selected additions and changes.
11. When complete, exit the Schema Editor and save the changes to the master CMF file.

If more than one configuration is to be updated, the procedure described above should be repeated for the other configurations that are to be updated.

Session Files

Session files allow you to save registered tool information, file configuration information, tool map schema information for loaded map files, and displayed windows and dialog boxes that you want to restore when you reopen the Schema Editor. When you exit the Schema Editor, the software prompts you to save your session file so that you can return to your work easily. You can create, modify, and open session files in the Schema Editor using the **File > Session Files** commands. When you edit the CMF file from SmartPlant Foundation in version 2008 and above, the Schema Editor automatically saves a session file and uses it to determine the state of the Schema Editor when you launch it again from the Desktop Client.

When you create a session file manually in the Schema Editor or the Desktop Client creates a session file for you, the session file is saved in your My Documents folder on your computer. The file extension for session files is .SES.

See Also

Open a Session File (on page 92)
Create a Session File (on page 92)

Open a Session File

1. Click **File > Session Files > Open Session File**.
2. Browse to the folder where your session file is located, and click **Open**.
3. In the **Set Active Configurations** dialog box, select the schema version or versions that you want to look at, and click **OK**.

   TIP By default, the changes you make are applied to all versions of the schema managed by the CMF file, to support forward and backward compatibility between versions of the software. Click **Advanced** to view the **Set Configurations for new objects window**, where you can select a specific version to which to write changes.

See Also

Session Files (on page 92)

Create a Session File

1. In the Schema Editor, create the session that you want to save to a file.

   NOTE Settings that you might want to save in a session file include:

   - Registered tool information
   - Open file or file configuration information
   - Loaded tool map schemas
   - Displayed windows and dialog boxes

2. Click **File > Session Files > SaveAs Session File**.
3. Browse to the location where you want to store your session file.

   By default, the Schema Editor saves your session file to your **My Documents** folder.

4. Type a name for the session file.
5. Click **Save**.

See Also

Session Files (on page 92)

File Configurations

In the Schema Editor, you can use file configurations to save time by grouping the files that you want to open, the order in which you want to open them, dependencies among schema files, and whether schema files are editable. File configurations also determine which file is the active file in the Schema Editor when you open multiple files of the same type.

File configurations are especially important for extending the schema for stand-alone SmartPlant Foundation installations and for mapping for tools without metadata adapters. When you need to make modifications to the SmartPlant schema without managing the schema in the SmartPlant Foundation Desktop Client, we recommend that you save your changes in a schema extension file. After you create a schema extension file, you can create a file configuration that automatically opens the SmartPlant schema as read-only and then opens your schema extensions file so that you can make the required changes.

You can save file configurations and open saved configuration files with the **File > File Configurations** commands.

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)
Create a File Configuration (on page 93)
Modify a File Configuration (on page 94)
Open a File Configuration (on page 94)
Opening Files (on page 96)

Create a File Configuration

1. Open the file or files that you want to include in the file configuration. For more information, refer to one of the following related topics:
   - *Open a Schema* (on page 97)
   - *Open a Tool Map Schema* (on page 98)
   - *Open a Data File* (on page 99)

   TIP The order that you open files in the Schema Editor determines the order the files will be opened when you opened a saved configuration containing those files.
2. If you open more than one schema file, set the active schema file. For more information, see *Set the Active Schema File* (on page 98).
3. Click **File > File Configurations > Save As Configuration**.
4. Type the name of the configuration in the **File name** box, and then click **Save**.
5. To define dependencies for files contained in your configuration and whether files in the configuration are editable, click **File > File Configurations > Edit Configuration**. For more information, see *Modify a File Configuration* (on page 94).

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)
Modify a File Configuration (on page 94)
Open a File Configuration (on page 94)

Open a File Configuration

1. Click **File > File Configurations > Open Configuration**.
2. Browse to the configuration file that you want to open.
3. Click **Open**.

See Also

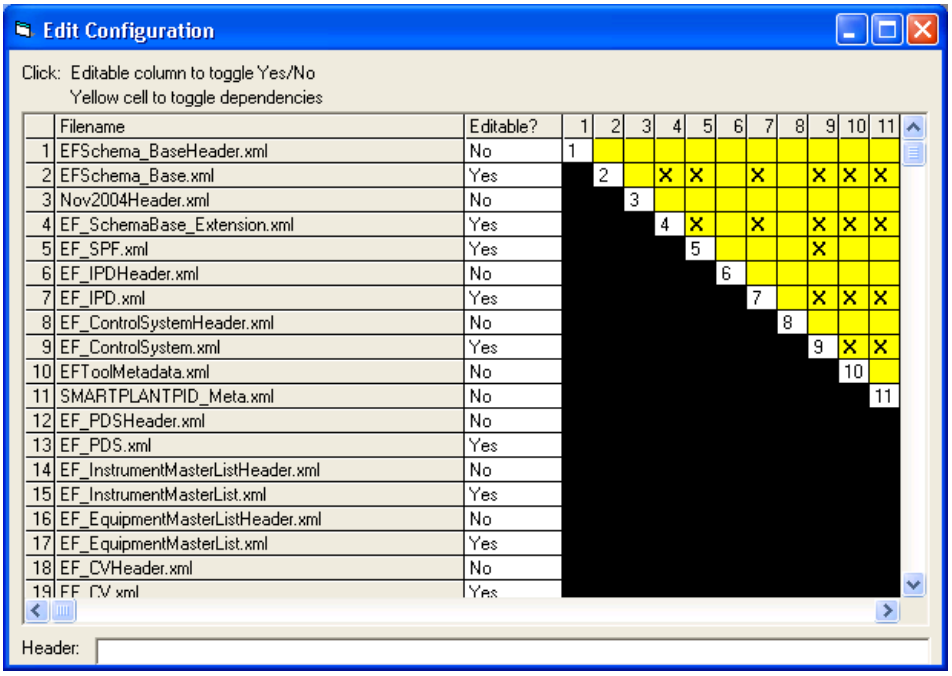
Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)
Create a File Configuration (on page 93)
Modify a File Configuration (on page 94)

Modify a File Configuration

1. Open the file configuration that you want to modify. For more information, see *Open a File Configuration* (on page 94).
2. Click **File > File Configurations > Edit Configuration**.
3. In the **Edit Configuration** window, click in the **Editable?** column for each schema file to set whether that file is editable when it is opened in the selected configuration.
4. To define dependencies for a selected schema file, click a yellow cell in the numbered column for the dependency.

   TIPS

   - An X in a numbered column in the row for the schema file indicates a dependency for that schema file.
   - The numbered columns across the top of the table correspond to the numbered rows along the left side of the table.



TIP You can also click the **Another Tool Schema** button in the **Workflows** dialog box to set the active schema.

5. Click **X** in the title bar of the **Edit Configuration** window to close the window and save your changes.

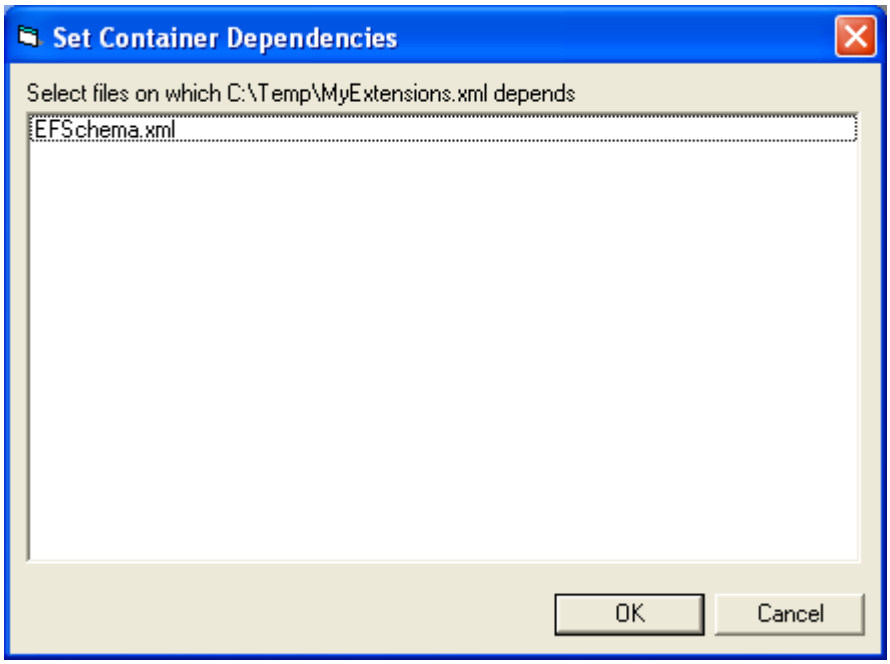
See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)
Create a File Configuration (on page 93)
Open a File Configuration (on page 94)

Set Dependencies for a Schema File

1. Open the schema that you want to define dependencies for and any files that it is dependent on.
2. Set the schema that you want to define dependencies for as the active schema.
3. Click **Tools > Another Schema > Set Dependencies for Active Schema File**.

4. In the **Set Container Dependencies** dialog box, click the schemas on which the active schema depends, and click **OK**.



NOTES

- After you define dependencies for a schema file, you can save the open files and dependencies to a file configuration so that you do not have to open the schema files and set dependencies every time you make modifications. To save the dependencies as part of a file configuration, click **File > File Configuration > Save As File Configuration**.
- You can define which files in the file configuration are editable after you save the file configuration. For more information, see *Modify a File Configuration* (on page 94).

See Also

Create a schema extension file (on page 207)
Extending the SmartPlant Schema (on page 146)
Create a File Configuration (on page 93)
Modify a File Configuration (on page 94)
Open a File Configuration (on page 94)

Opening Files

In the SmartPlant Schema Editor, you can also open the schemas, tool map schemas, and data files you want to view and edit. Generally, customers should use CMF files or CFGs to avoid opening files manually and make schema editing easier.

If you do decide to open files manually, you must open a schema file before you can start viewing schemas, tool map schemas, or data files. You can open a schema file directly using the commands on the **File** menu or on the buttons on the **Workflows** dialog box.

File Configurations

To open the files you need to work on in the Schema Editor, you can also create a file configuration. When you open a file configuration, the Schema Editor automatically opens the files that you selected. For more information about file configurations, see *File Configurations* (on page 93).

Active Files

When you open multiple schema files or tool map schema files, the file that you see in various views in the Schema Editor is called the active file. When you have multiple files open, you can change which file you are viewing by using the **Tools > Another Schema > Set Active Schema File** and **Tools > Another Tool Schema > Set Active Tool Schema File** commands. These commands are only available when you have multiple schema files or multiple tool map schema files open.

When you save a file configuration, the Schema Editor stores the active file as well.

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

Open a Schema

Before you can begin viewing the schema, data files, or a tool map schema, you must open the schema in the Schema Editor.

1. Click **File > Open > Schema File**.

   IMPORTANT

   - Typically, you should open and edit the SmartPlant schema using the delivered configuration file (EFSchema.cfg).
   - The SmartPlant schema configuration file references the SmartPlant schema file (read-only), other schema files (read-only), and your schema extensions file (editable). The delivered schema extensions file is named Extensions1.xml, and when you open EFSchema.cfg, your changes are saved in Extensions1.xml by default. However, you can specify additional extensions files to store your schema modifications. For more information, see *Modify a File Configuration* (on page 94).
   - The SmartPlant schema configuration file is delivered with the SmartPlant Server and Client components. By default, it is installed in the following location: ..\Program Files\Common Files\Intergraph\EFSchema\<software version>. However, if you use different SmartPlant schemas for each SmartPlant Foundation site, your schema file may be in a different location.
   - If multiple sites share the same SmartPlant schema file, then any changes that you make to the shared schema file will affect every site. Therefore, Intergraph strongly recommends that you make a copy of the delivered SmartPlant schema folder for each site. Store each customized folder within the site folder, for example, <drive>:\SmartPlant Foundation 2007 Server Files\Web_Sites\<site_name>\EFSchema\03.08. The folders containing the customized schema files must be network shared with read-only access.

   TIP If you already have a schema open in the Schema Editor, you can click **File > Open > Another Schema File** to open another schema.

2. Browse to the directory where the schema file is located.
3. Select the schema file that you want to open.
4. Click **Open**.

NOTE You can open XML schema files (.XML) and XML DOM schema files (.DOM) in the Schema Editor.

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

Set the Active Schema File

1. If multiple schemas are open in the Schema Editor, click **Tools > Another Schema > Set Active Schema** to set the active schema. The active schema is the one that you are editing and searching. Although you can see multiple schemas at once, you can only edit one at a time.
2. Select the schema that you want to be active in the **Set Active File** window, and click **OK.**

See Also

Set the Active Tool Schema File (on page 99)

Open a Tool Map Schema

Before you can open a tool map schema file, you must open the SmartPlant schema or a file configuration that contains the schema. For more information, see *Open a Schema* (on page 97).

1. Click **File > Open > Tool Schema File**.

   TIP If you already have a tool map schema open in the Schema Editor, click **File > Open > Another Tool Schema File** to open another tool map schema.
2. Browse to the directory where the file is located.
3. Select the file that you want to open.
4. Click **Open**.

NOTE You can open tool map schemas in XML (.XML) and Excel (.XLS) format in the Schema Editor.

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

Set the Active Tool Schema File

1. If multiple tool map schemas are open in the Schema Editor, click **Tools > Another Tool Schema > Set Active Schema** to set the active tool map schema. The active tool map schema is the file that you are editing and searching. Although you can see multiple tool map schemas at once, you can only edit one at a time.
2. Select the tool map schema that you want to be active in the **Set Active File** window.

See Also

Set the Active Schema File (on page 98)

Open a Data File

The data that can be viewed by the Schema Editor is the contents of the XML file that is created as a result of a publish operation. You can view the data organized by schema class with many of the schema views, such as the **Tree/Viewable UML** or **Tree/Properties** views. With these views, you can identify the class, interface, and property elements contained in the XML file.

Before you can open a data file, you must *open a schema* (on page 97) or a file configuration that contains a schema.

1. Click **File > Open > Data File**.
2. Browse to the directory where the file is located.
3. Select the file that you want to open.
4. Click **Open**.

NOTE You can open XML data files (.XML) in the Schema Editor.

See Also

Open a Data File Set (on page 99)
Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

Open a Data File Set

You can open multiple data files in order to view the differences and similarities between different versions of the files. You can also save a copy of the comparison report.

Before you open a data file set, you must open a schema or a file configuration that contains a schema.

Open the Data File Set

1. Click **File > Open > Data File Set**.
2. Click **Add Files from .txt File** to add a set of files that are listed in a plain text file.

   NOTE The text file must contain the full path to the file you want to add. Separate data file paths with a return.
3. Click **Add File** to add a single file to the Data file list.
4. Highlight a data file and click **Remove Selected Files** to remove it from the list.
5. Enter a name for the data file set in the **Name for file set** field.

6. Click **OK**.

View the Data File Set

1. Click **View > Data File Set**.
2. In the **View Data File Set** dialog box, click **OK**.

NOTE You can use the Compare with File command to compare the data file sets. For more information, see *Compare Files* (on page 464).

See Also

Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

Compare a Data File Set

You can open a second data file set to compare the differences and similarities, and save the comparison file separately.

Before you can use the **Compare with Data File Set** command, you must open a data file set.

Open the Compare Data File Set

1. Click **File > Open > Compare Data File Set**.
2. Click **Add Files from .txt File** to add a set of files that are listed in a plain text file.

   NOTE The text file must contain the full path to the file you want to add. Separate data file paths with a return.
3. Click **Add File** to add a single file to the Data file list.
4. Highlight a data file and click **Remove Selected Files** to remove it from the list.
5. Enter a name for the data file set in the **Name for file set** field.
6. Click **OK**.

View the Compare Data File Set

1. Click **View > Compare with Data Set**.
2. In the **View Data File Set** dialog box, click **OK**.

NOTE You can use the Compare with File command to compare the data file sets. For more information, see *Compare Files* (on page 464).

See Also

Open a Data File Set (on page 99)
Set the Active Schema File (on page 98)
Set the Active Tool Schema File (on page 99)

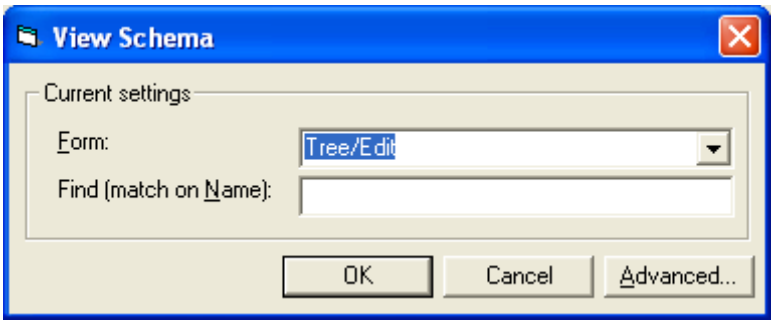
SECTION 6

Navigating Files and Data in the Schema Editor

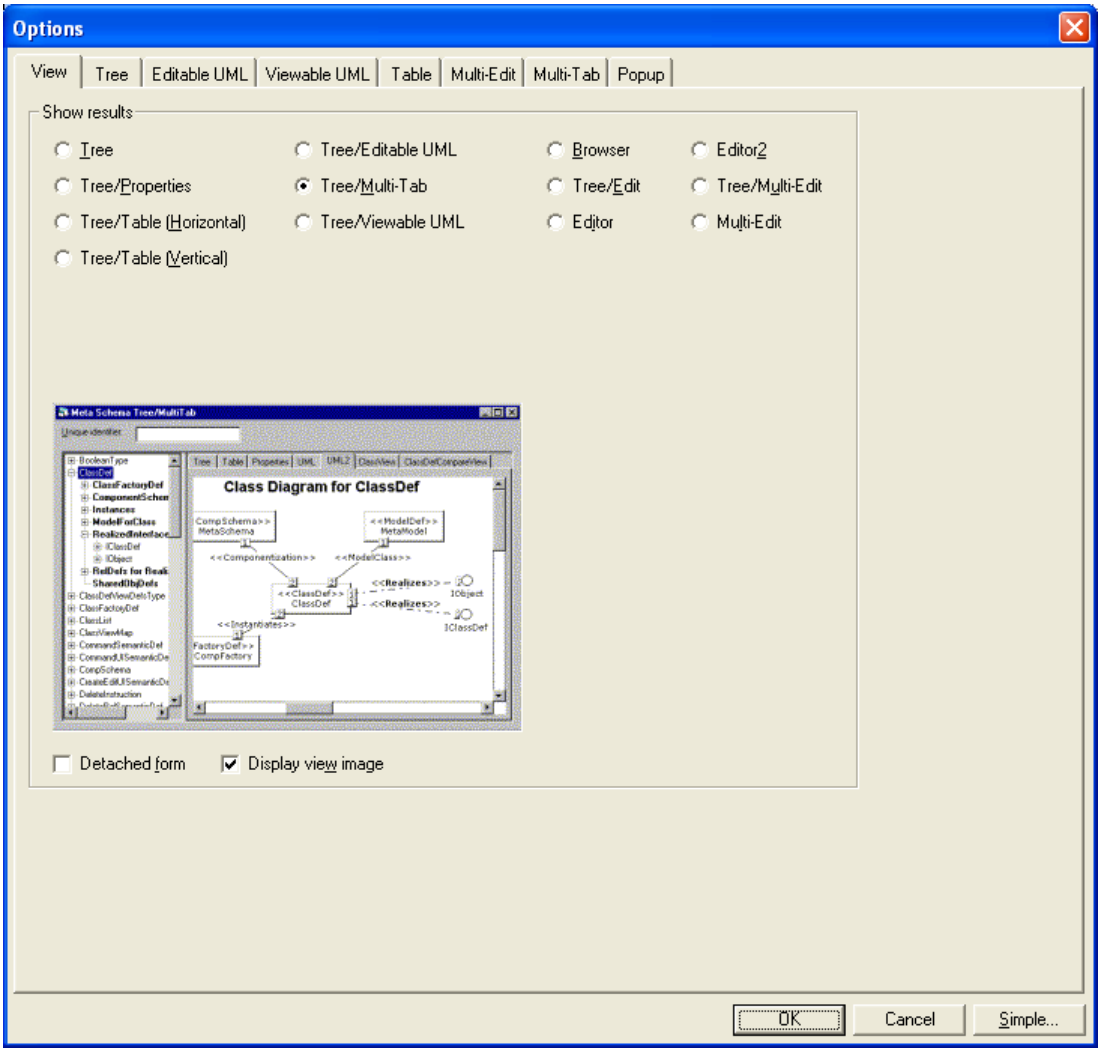
In the Schema Editor, you can view any schema, tool map schema, or data file in .XML format. When you open a file, it is loaded into memory. To actually see the schema file, use the commands on the **View** menu to display the **View** dialog box.

View Dialog Box

The **View** dialog box allows you to select the view that you want to use to see your schema, tool map schema, or data file. The **View** dialog box has two modes: simple and advanced. In **Simple** mode, you can select the view you want from a list without setting view options.



In **Advanced** mode, click **Options** to display the **View** tab. The **Options** dialog box has multiple tabs that allow you to select a view, set options for views, set the scope for your view, and so on.



TIP You can switch between modes in the **View** dialog box by clicking the **Advanced** button to switch to **Advanced** mode and the **Simple** button to switch to **Simple** mode.

Schema Editor Views

The many different views in the Schema Editor provide different types of information that help you understand how the schema, tool map schema, and data files work. For example, when you view the schema in any of the tree views, except the schema tree/UML view and the schema hierarchy views, you can typically see the schema class definitions. As you expand nodes in the tree views, you can navigate the relationships in the schema and view interface definitions, property definitions, and even instances of the class if you also have a data file open in the Schema Editor.

The Schema Editor also provides graphical views. These views allow you to see both static and dynamic UML diagrams containing the classes, relationships, interfaces, and properties defined in the file.

Viewing Multiple Objects

The following controls support multi-object viewing:

- Tree
- Table (hor) and Table (vert) and the other view-based tabs (at right end of multi-tab control)
- Editable UML
- MultiEdit

Typically, you use the **Tree** view to select and see multiple objects. To select multiple objects for display, use standard keyboard shortcuts, such as **Ctrl**,**Shift**, or **Ctrl-A**. Click an object that you select and it displays in the Schema Editor window. If you are viewing the schema using the **Editable UML** view, drag the objects to the Schema Editor window.

NOTE Objects selected from the tree control must have the same source node. Selecting objects from different source nodes will clear the objects previously selected from a different source node.

View Navigation

After you select the view that you want, you can navigate in the view in different ways. In tree views, you can click + beside a node to expand it. In table views, you can select an object in the associated tree view to display data in the table. In UML views, you can drag and drop objects to create new UML views and modify existing UML views.

See Also

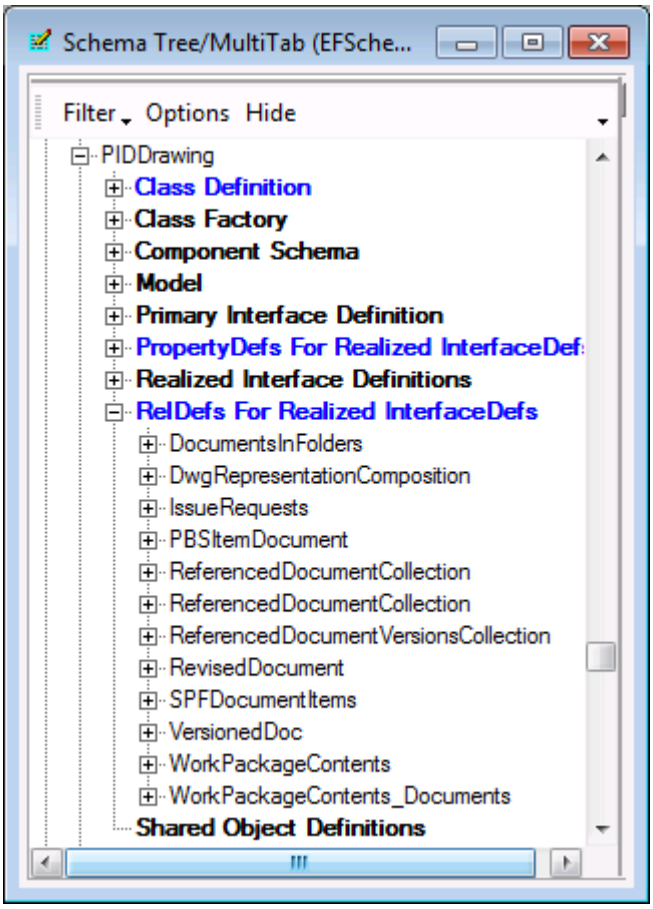
Tree/Table View: An Example (on page 105)

Tree View: An Example

The simplest way to view the schema in the Schema Editor is the **Tree** view. The **Tree** view displays the classes in the top-level nodes. When you expand a class, you can see the relationships defined for the class.

The **Tree** view is useful for general navigation through relationships, but not for looking at property values. You can navigate all the relationships defined in the schema using the Schema tree view.

For example, in the **Schema Tree**, you can trace the PIDDrawing class all the way to the relationship definitions for the interfaces it realizes.



However, the schema tree is not a true hierarchy. The Implies relationships in the schema allow you to navigate through the schema almost endlessly.

TIP Bold text in the tree view represents relationship edges. Relationship edges define paths that go from one point to another point in the data without traversing the entire tree. Blue relationships are relationships across which only limited operations can be performed. For example, adding or deleting relationships is not supported as shortcut menu commands across blue relationship edges whereas they are supported across black relationships.

Blue edges either involve more than one relationship traversal or they involve traversal across special relationships. A special relationship is actually not a relationship at all, but is instead something that the software determines. For example, the Class Definition for an object is not a relationship between the object and its class definition (ClassDef) but is instead something that the software determines through other information.

To view a file in the standard tree view, do the following:

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree**.

TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.

4. Click **OK**.
5. To traverse the relationships in the **Tree** view, expand the nodes.

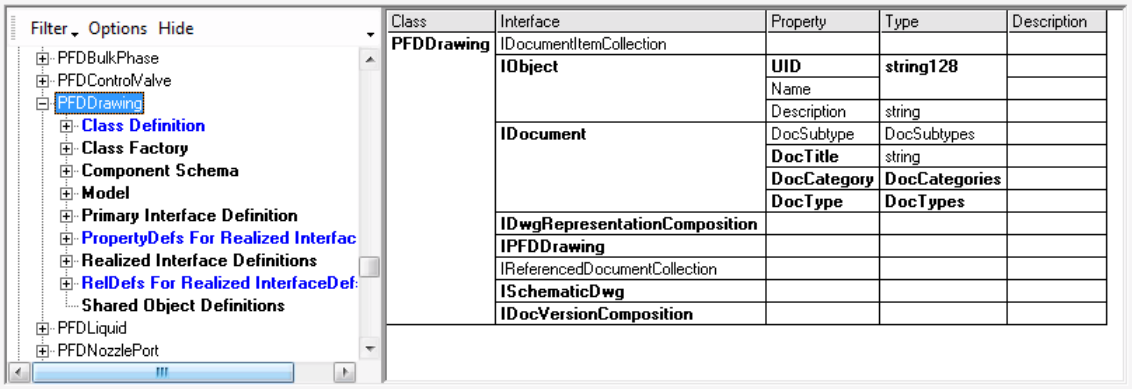
See Also

Navigating Files and Data in the Schema Editor (on page 101)

Tree/Table View: An Example

The **Tree/Table** view is a combination of the standard tree view with a table that provides a different view of the information based on what you select in the tree view. The table views are predefined by class to show the most useful information for each type of object.

For example, if you select the PFDDrawing class in the tree view, you can view all the interfaces realized by the class and all the properties and the corresponding property types exposed by those interfaces in the table view.



Class	Interface	Property	Type	Description
PFDDrawing	IDocumentItemCollection			
	IObject	**UID**	**string128**	
		Name		
		Description	string	
	IDocument	DocSubtype	DocSubtypes	
		DocTitle	string	
		DocCategory	**DocCategories**	
		DocType	**DocTypes**	
	IDwgRepresentationComposition			
	IPFDDrawing			
	IReferencedDocumentCollection			
	ISchematicDwg			
	IDocVersionComposition			

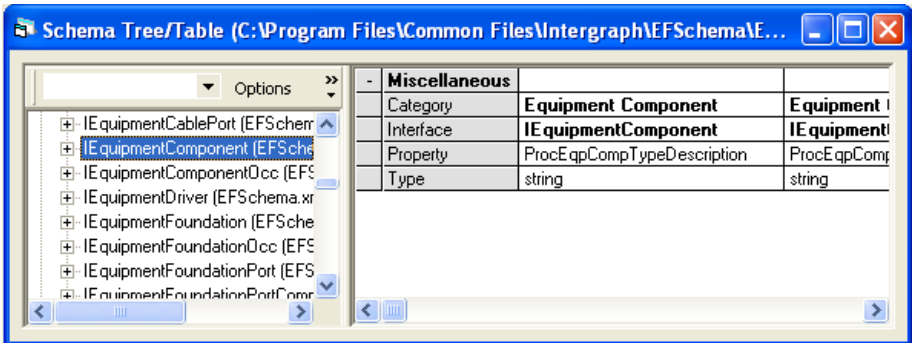
In the **Table** view, bold text is used in class definitions and interface definitions to identify required entries. Bold interfaces in the **Table** view indicate that the interface is required for the associated class. For example, in the previous example, bold text shows that IDocument is a required interface for PFDDrawing.

Properties that appear in bold text in the **Table** view are required for the associated interface. For example, DocCategory, DocTitle, and DocType are required properties for the IDocument interface, while DocSubtype is not required.

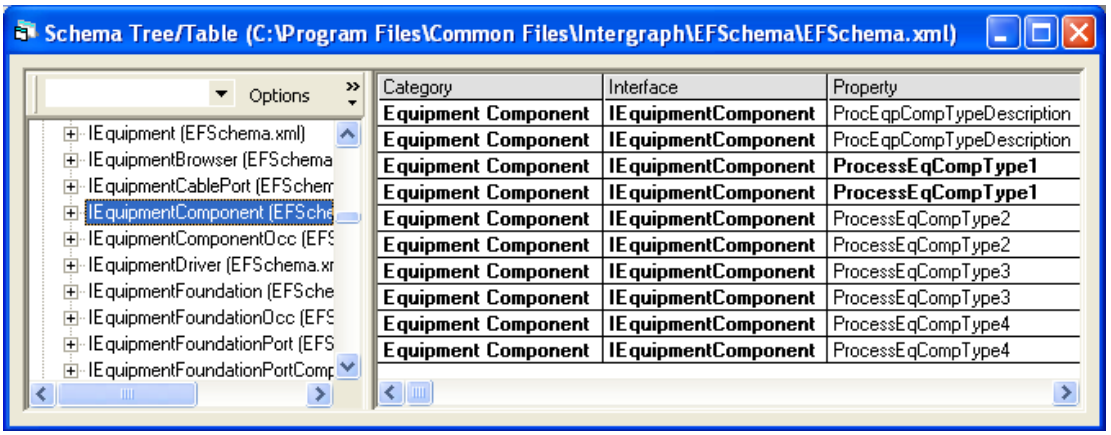
Bold property types in the **Table** view indicate that the value for the property cannot be null. For example, for the IObject interface, the UID property is required, and its string value cannot be null.

Bold text in the **Tree** view represents relationship edges. Relationship edges define paths that go from one point to another point in the data without traversing the entire tree.

There are two types of tables that you can choose from: vertical and horizontal. The vertical table displays row headers along the left side of the table and data in vertical columns.



The horizontal table displays column headers across the top of the table, with the data displayed in horizontal rows.



To view a file in the **Tree/Table** view, do the following:

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree/Table Horizontal** to display a horizontal table or **Tree/Table Vertical** for a vertical table.

   TIPS

   - If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
   - If you want to change how data is displayed in the **Table** view, make changes on the **Table Options** tab.
4. Click **OK**.
5. To traverse the relationships in the **Tree** view, expand the nodes.
6. To see the predefined **Table** view for an object, click the object in the **Tree** view.

See Also

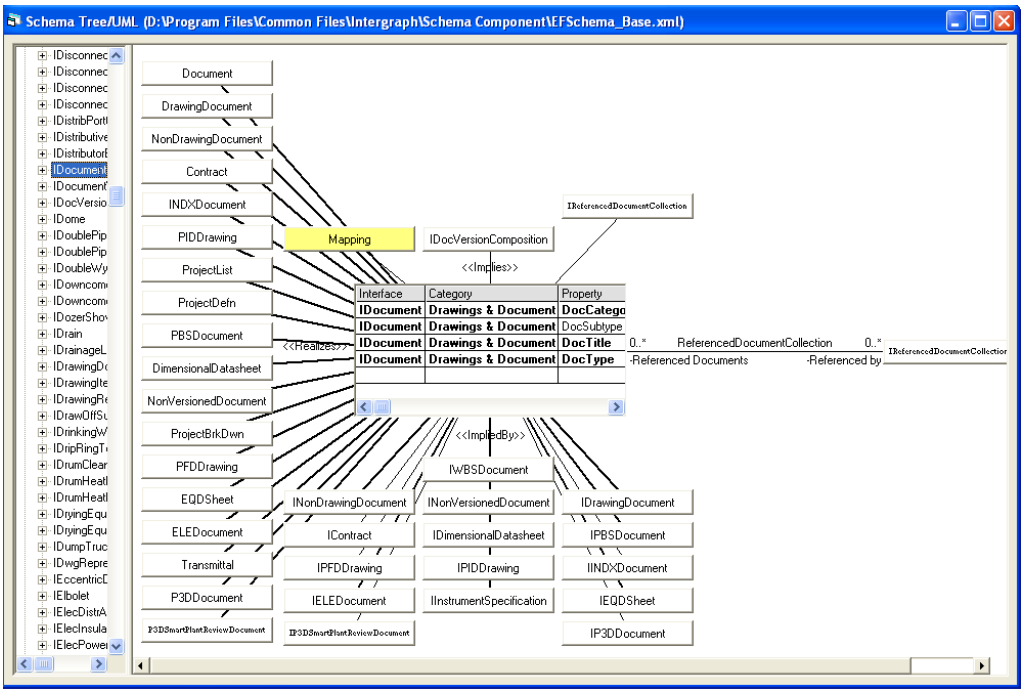
Navigating Files and Data in the Schema Editor (on page 101)

Tree/Viewable UML View: An Example

The **Tree/UML** view is a combination of the standard tree view with a Unified Modeling Language (UML) view. The **UML** view provides a graphical representation of the relationships and relationship definitions involving the object selected in the tree view. The selected object appears in the center of the **UML** view. In the **UML** view, you can click any object to center the UML around the selected object to view its relationships and relationship definitions.

The **UML** view is useful for viewing relationships graphically. It is also useful for visualizing relationship definitions that are not displayed elsewhere.

In the following graphic, the IDocument interface is selected in the **Tree** view, and all its relationships and the properties it exposes appear in the **UML** view. Each type of relationship, such as <<Realizes>> or <<Implies>>, is shown inside angle brackets between the selected object and other classes and interfaces in the **UML** view.



Relationship definitions include the objects on both ends of the relationship and the roles specified for both ends of the relationship. For example, in the graphic a ReferencedDocumentCollection relationship exists between IDocument and IReferencedDocumentCollection. The first end of the relationship has a role of Referenced Documents and a cardinality of zero-to-many, which means any number of objects can be involved on this end of the relationship. On the second end of the relationship, the role is Referenced by, and the cardinality is also zero to many.

TIP Bold text in the tree view represents relationship edges. Relationship edges define paths that go from one point to another point in the data without traversing the entire tree.

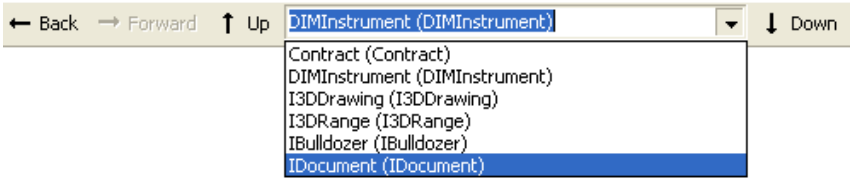
To view a file in the **Tree/UML** view, do the following:

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.

2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree/UML**.

   TIPS

   - If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
   - If you want to change how data is displayed in the tree view, make changes on the **Tree Options** tab on the **Options** dialog box.
   - If you want to change how data is displayed in the **UML** view, make changes on the **Viewable UML** tab on the **Options** dialog box.
4. Click **OK**.
5. To view the UML for an object, click the object in the **Tree** view.
6. To center the UML view on another object in the current view, click the object in the **UML** view.
7. To go back to the last **UML** view, click **Back** on the toolbar.
8. If you browse back to a previous view, click **Next** to see the next UML view.
9. To select a particular **UML** view that you have recently viewed, click the name of the view that you want to see in the view list.



10. To move through the view list, use the **Up** and **Down** buttons on the toolbar.

See Also

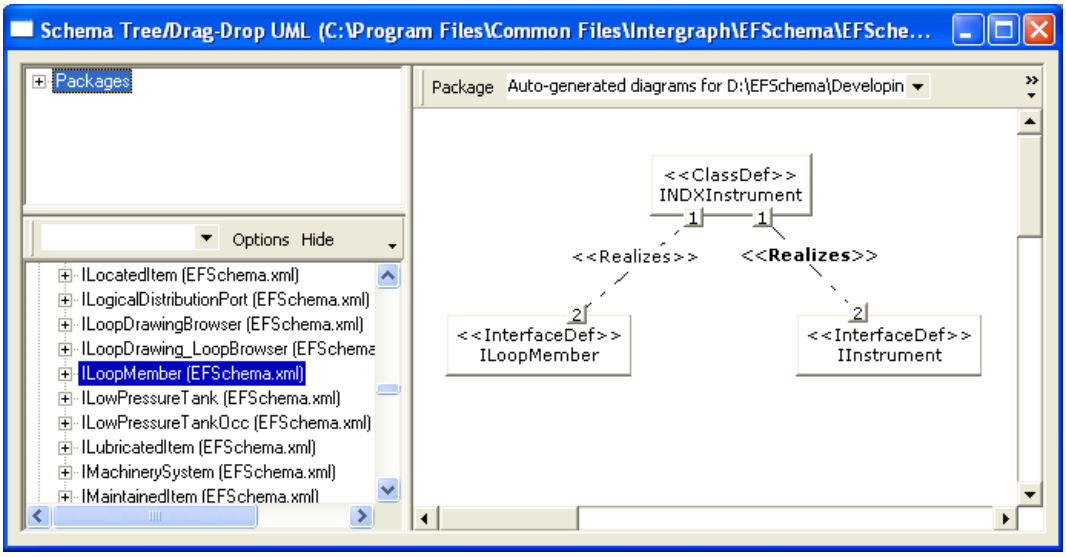
Navigating Files and Data in the Schema Editor (on page 101)

Tree/Editable UML View: An Example

The **Tree/Editable UML** view displays a tree view on the left of the drag-and-drop UML pane. You can select items from the tree view to display in the **UML** view by dragging the objects from the **Tree** view to the **UML** view. As you drag items to the **UML** view and drop them, the relationships among those items are automatically displayed in the **UML** view.

When you drag and drop an object to the UML view, you can choose whether you want to display the default UML view for that object or create your own. If you create your own UML view, you must give the UML view a name and description and define which existing package to which you want to add the new UML view.

For example, dragging the INDXInstrument, ILoopMember, and IInstrument classes and interfaces from the **Tree** view to the UML view without displaying the default UML views for each object results in the UML diagram shown in this graphic:



In the resulting **UML** view, you can see the realizes and implies relationships between the selected objects and the relationship ends for each one.

You can also use the **Tree/Editable UML** view to select an existing UML view that you want to display using the **Packages** tree. Packages are used to organize UML diagrams. Clicking a UML diagram in the **Packages** tree or dragging and dropping the UML diagram from the **Packages** tree to the **UML** view displays that UML diagram. You can also create your own UML views, name them, and store them as part of a package.

After you create your own **UML** view by selecting the objects that you want to display in the UML view or select a UML view from the **Packages** tree, you can customize the display even further by doing one of the following:

- Moving objects and labels
- Changing the display color for objects
- Removing objects, relationships, and labels
- Changing the display of interfaces
- Aligning objects

To move an object in the **UML** view, drag the object to a new location. You can move classes, interfaces, and other objects in the **UML** view. As you move objects in the **UML** view, they remain attached to each other by their relationships.

To use commands for modifying the UML display for a particular object, right-click the object in the **UML** view, and then click the appropriate command on the shortcut menu. To align objects in the **UML** view, right-click the background of the **UML** view, and then click **Align** on the shortcut menu.

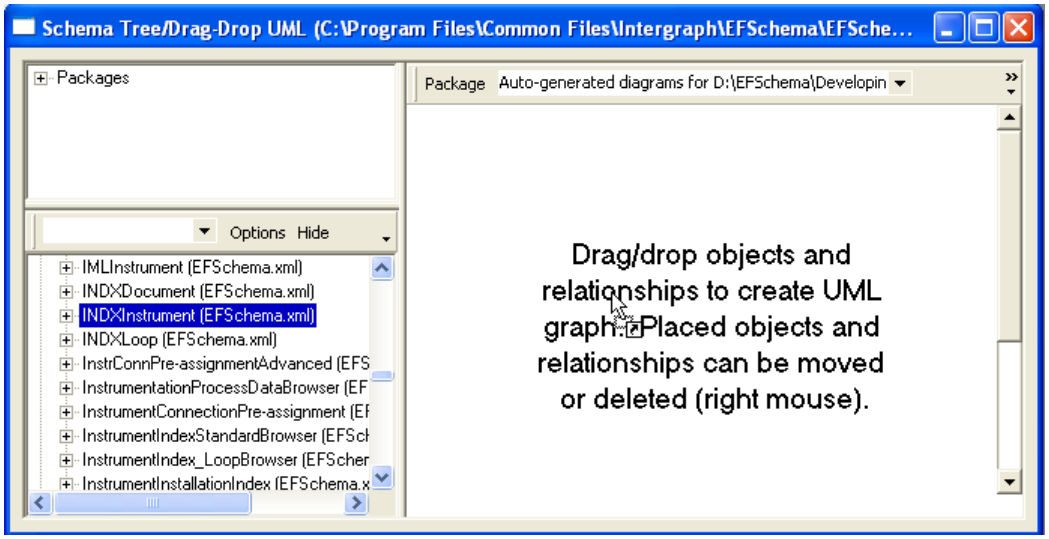
After you create a UML view, you can clear the view and start again. To create a new UML view, right-click the background of the **UML** view, and then click **New UML view definition** on the shortcut menu.

To view a file using the **Tree/Editable** view, do the following:

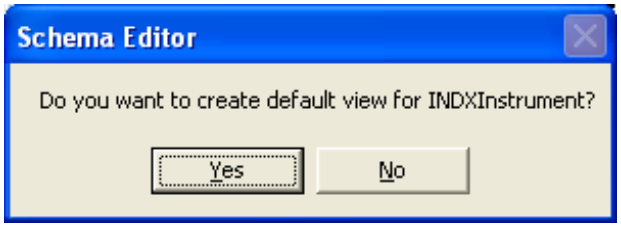
1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree/Editable UML**.

   TIPS

   - If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
   - If you want to change how data is displayed in the **Tree** view, make changes on the **Tree Options** tab on the **Options** dialog box.
   - If you want to change how data is displayed in the **UML** view, make changes on the **Drag/Drop Options** tab on the **Options** dialog box.
4. Click **OK**.
5. Navigate to an object in the **Tree** view at the bottom of the **Tree/Editable UML** window.
6. Drag the object to the **UML** view.



7. In the **Schema Editor** message box, click **No** to create a customized UML view.



   TIP To display the default UML view for the selected object type, click **Yes** in the **Schema Editor** dialog box.
8. Beside the **Package** box, click ... to select the existing package in which you want to store the UML view.

TIP If you want to create a new package to store your UML view, click **Create** to define the new package.

9. Define a display name and other properties for the UML view on the **Properties** tab of the **New UML View Definition** dialog box.
10. Click **OK** to save the UML view definition.
11. Move the object around in the **UML** view as needed.
12. Right-click the object to change the display of the object.
13. Add additional objects to the **UML** view by browsing in the **Tree** view and dragging the objects to the UML view.

NOTE You can view existing UML views by browsing to the view in the **Packages** tree and then selecting the UML diagram that you want to see in the **UML** view.

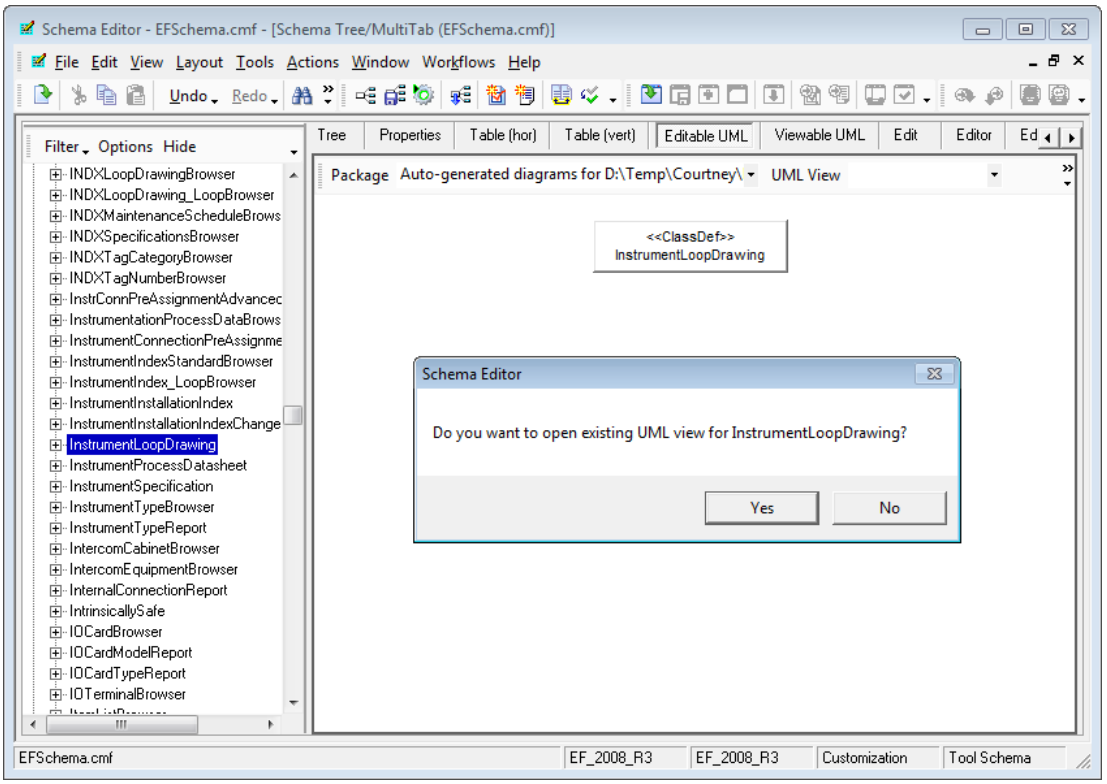
See Also

Navigating Files and Data in the Schema Editor (on page 101)

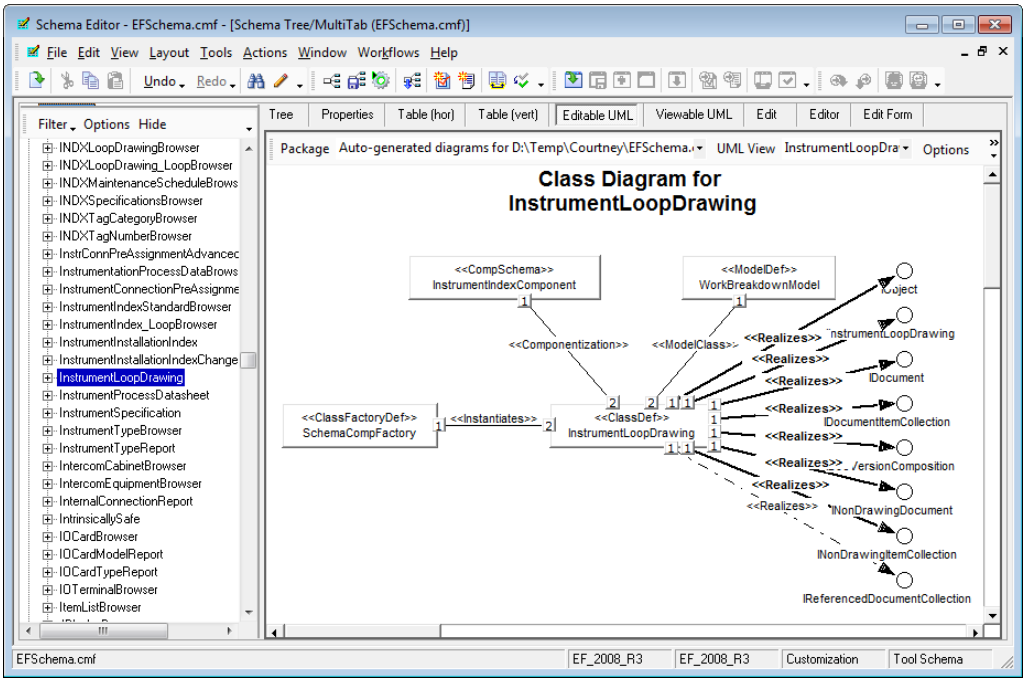
Create a UML Diagram

UML diagrams are created within the Schema Editor using the **Editable UML** view. If you want to create a UML diagram based on new objects, use the **Editor** tab on the **Multi-Tab** view. If you want to use existing objects to create a new UML diagram, use the **Editable UML** view on the **Multi-Tab** view.

Using the **Editable UML** view to create a UML diagram, drag the object from the **Tree** view to **Editable UML** window. A message appears asking you if you want to create a default UML diagram. Often, the default UML diagram produced contains information saves steps. Click **Yes** to produce a default UML diagram.



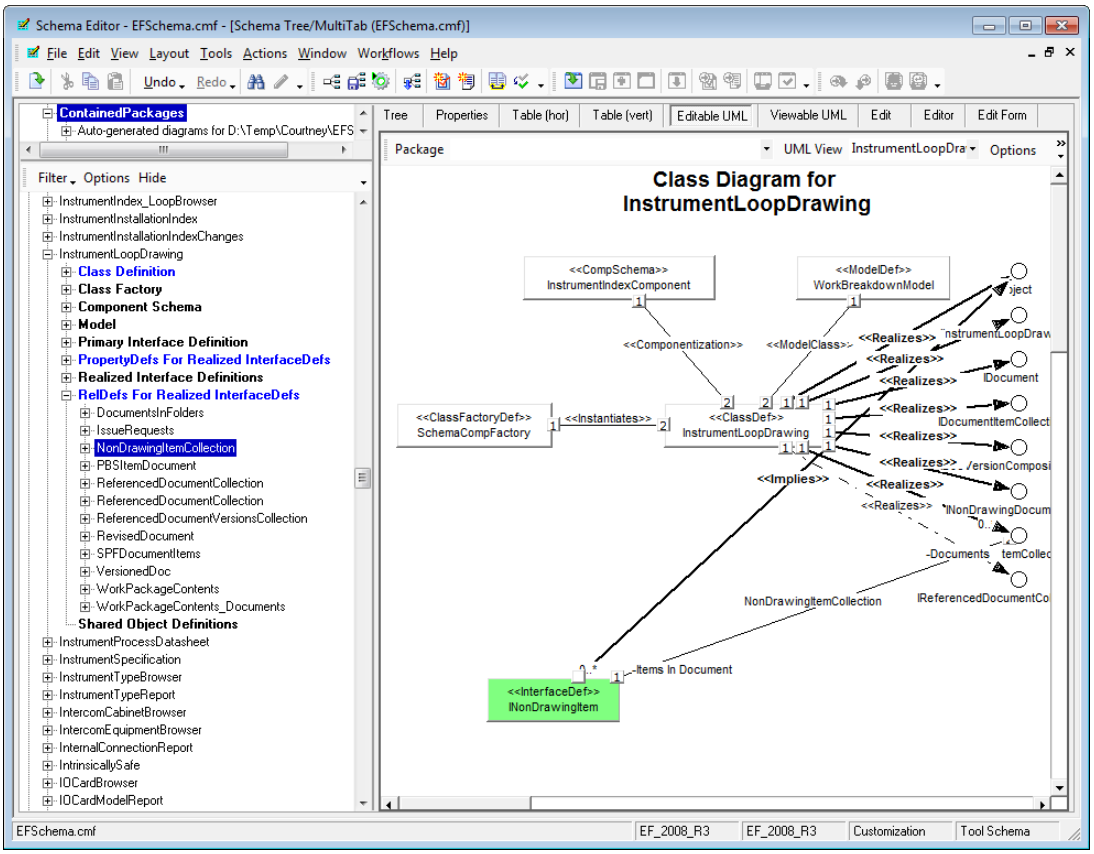
The default UML diagram appears.



Objects in the UML diagram can be repositioned by dragging them to the desired location. To remove displayed objects from the UML diagram, right-click on the selected object and then click **Erase.**

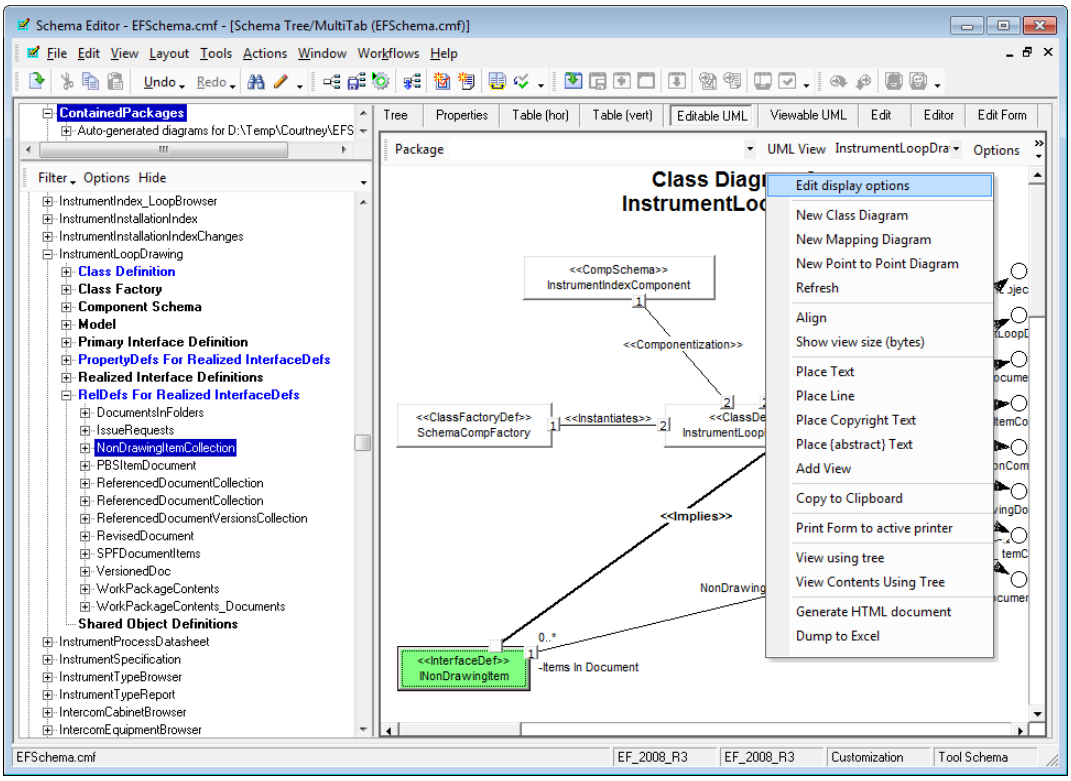
IMPORTANT Do not click **Delete** as this will delete the object and not just erase it from the diagram.

Additional objects can be added to the UML diagram by simply dragging the objects from the **Tree** view and dropping them onto the diagram at the desired location. For example, expanding the **InstrumentLoopDrawing** node in the **Tree** view and dragging **NonDrawingItemCollection** onto the UML diagram results in the following display.



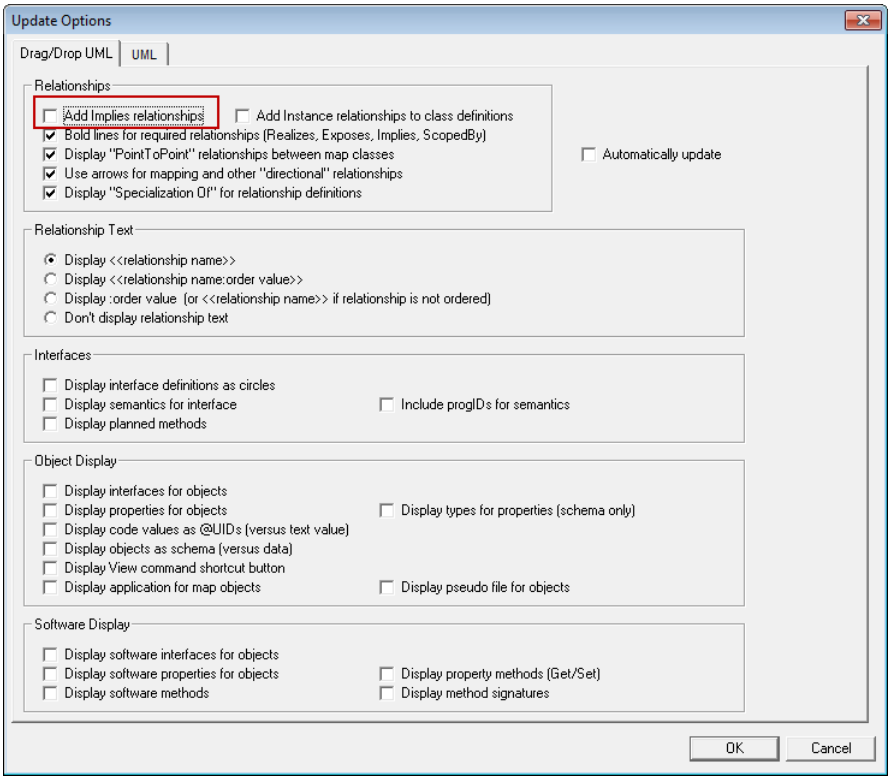
The **INonDrawingItemCollection** interface definition was added to the diagram in order to show the **NonDrawingItemCollection** relationship definition. A line for the **Implies** relationship from **INonDrawingItemCollection** to **IObject** was added to the diagram.

The automatic addition of Implies relationships can become annoying so the user may want to turn this option off by right-clicking in a whitespace area of the UML diagram, and then selecting **Edit display options** on the shortcut menu.

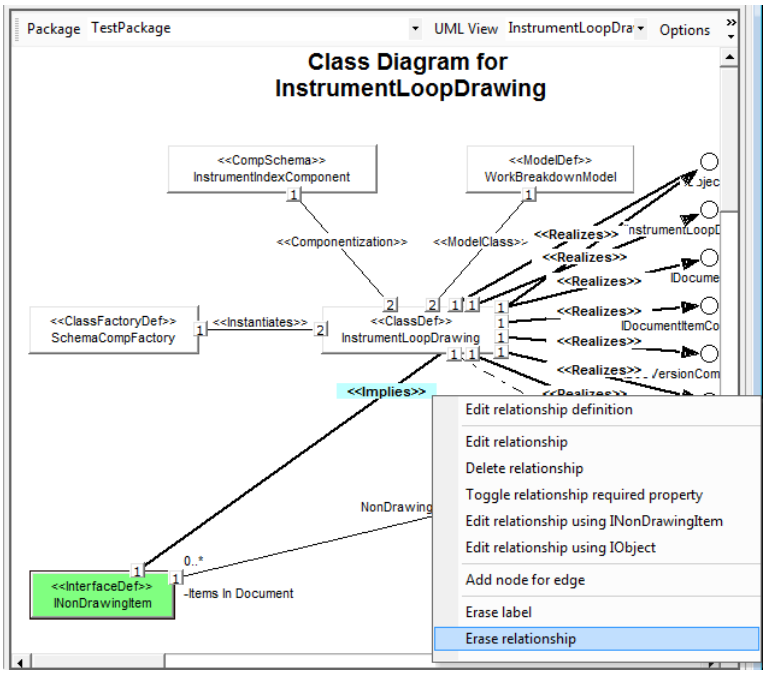


On the **Update Options** dialog box, remove the check from the **Add Implies relationships** check box. Turning this option off means that subsequent interface definitions added to the UML diagram will not trigger the addition of the Implies relationships for that interface definition. However, any existing Implies relationships on the UML diagram will not be affected.

NOTE Changing the option on this dialog box only changes the option for this view. Click **Tools > Options** to change it for subsequent **Editable UML** views.

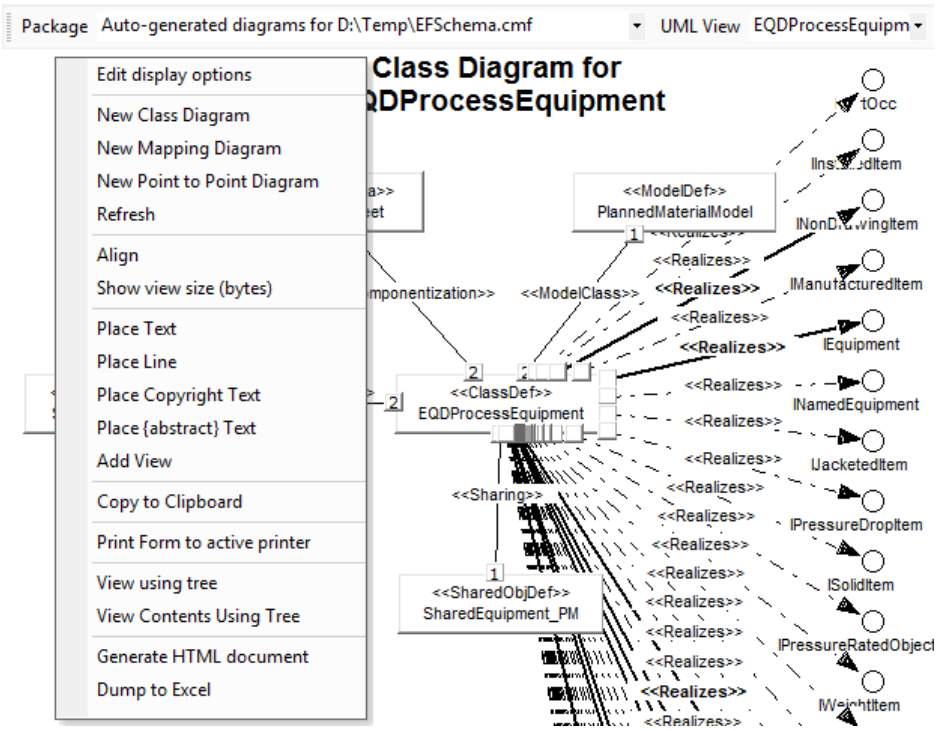


To remove the **Implies** relationship between **INonDrawingItemCollection** and **IObject**, right-click on the **<<Implies>>** text and click **Erase relationship**. Do not click **Delete relationship**. Or, right-click on the point (labeled 1 or 2) at either end of the relationship and then click **Erase relationship**.



Align Objects in Editable UML Views

1. Right-click anywhere in the background of the **Editable UML** view.



2. On the shortcut menu, click **Align**.

Remove Objects from Editable UML Views

1. In the **Editable UML** view, right-click the object that you want to erase.
2. On the shortcut menu, click **Erase**.

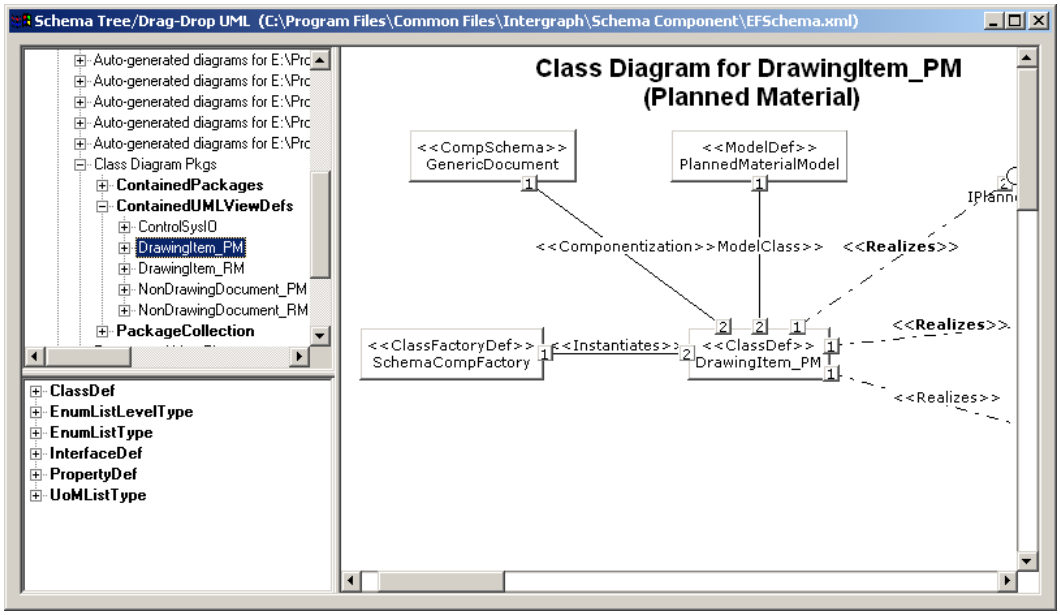
   The object and all its relationships are automatically removed from the UML view.

Clear Editable UML Views

1. Right-click anywhere in the background of the **Editable UML** view.
2. To clear the current UML view and start over, click one of the following on the shortcut menu:
   - **New Class Diagram**
   - **New Mapping Diagram**
   - **New Point-to-Point Diagram**

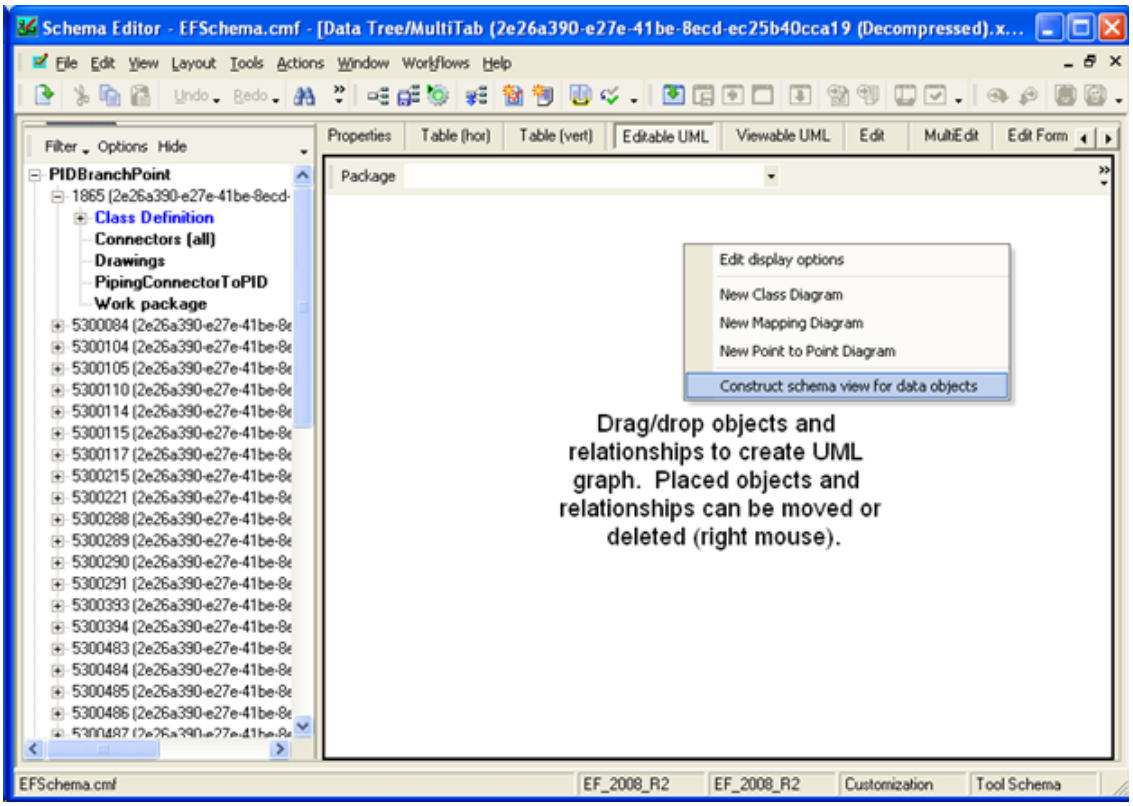
Open Saved UML Views

1. Click **View > Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree/ EditableUML**.
4. In the **Packages** tree, browse to the UML view that you want to view.
5. Select the UML view definition that you want to see in the UML view.



Auto-Generate UML Diagrams Based on Published Data

Tool map schema files do not provide complete information about what is published by an application. The **Construct Schema View for Data Objects** command addresses the lack of information by examining all the objects in a tool data file and generating a UML diagram for each class definition for which there is at least one instance in the data file. The **Construct Schema View for Data Objects** command is available on the shortcut menu in the **Editable UML** view.



Tool map schema files add to the logic built into the adapters for integrated applications. The tool map schema provides a way to extend and customize what is published. Tool map schemas are not intended to provide a detailed description of mapping for an application; much of what is published by that application can be hard-coded into the adapter. For example, the SmartPlant P&ID map file contains no information about the relationships that are published by the software. The SmartPlant P&ID application publishes a significant number of relationships. The logic for these published relationships is embedded in the SmartPlant P&ID adapter.

Each UML displays the following:

- A class definition displays all the interfaces and properties that exist for any of its instances within the data file. Interfaces and properties for a class definition that are not found in the data file are not included in the UML diagram.
- Relationship definition for which there are relationships in the data file that apply to instances of the class definition are displayed in the UML diagram. The interface definitions at both ends of each relationship definition are displayed.

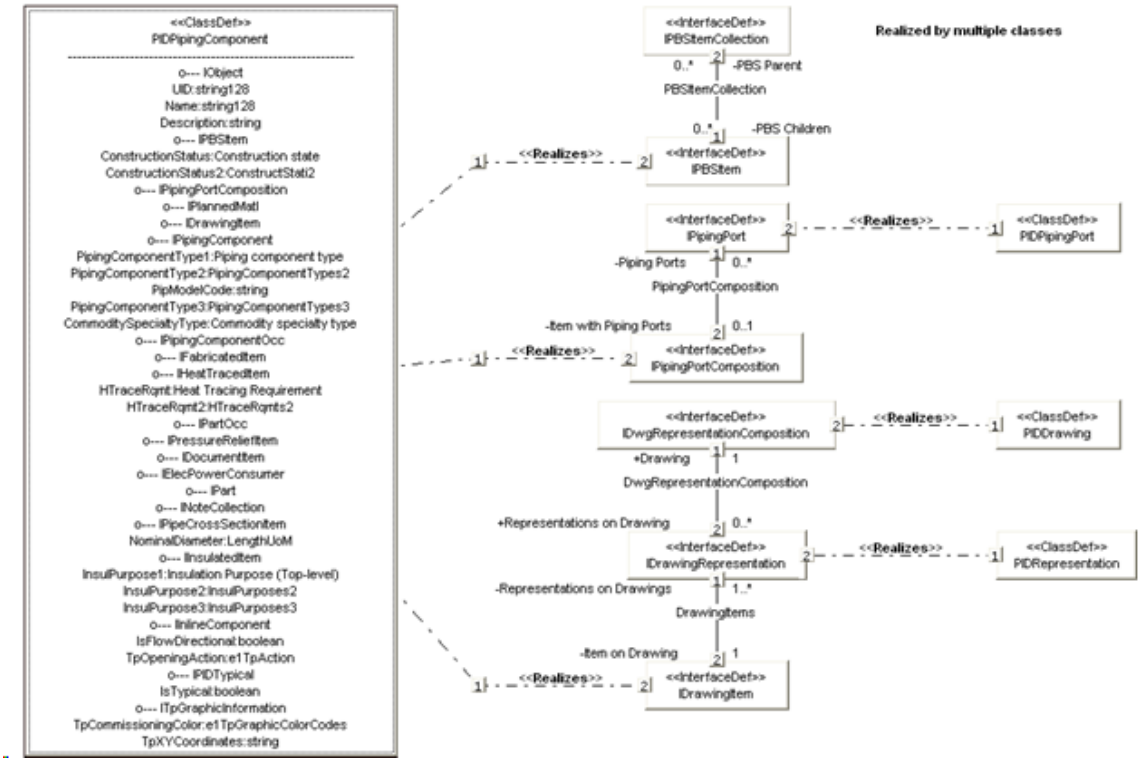
NOTE If there are other relationships in the data file that apply to either of these interface definitions, then they are also added to the UML diagram regardless of whether they apply to the class definition for the UML diagram.

- Interface definitions added as the result of displaying relationship definitions are connected to the class definition (through Realizes relationships) for the UML diagram as applicable.

  NOTES

  - For each interface definition added as the result of displaying relationship definitions that is not realized by the class definition, one or more class definitions that do realize that interface definition are added to the diagram.
  - If more than two class definitions realize the same interface definition, then the text "Realized by multiple class definitions" is added to the diagram.

A typical UML diagram generated using this command is shown below.



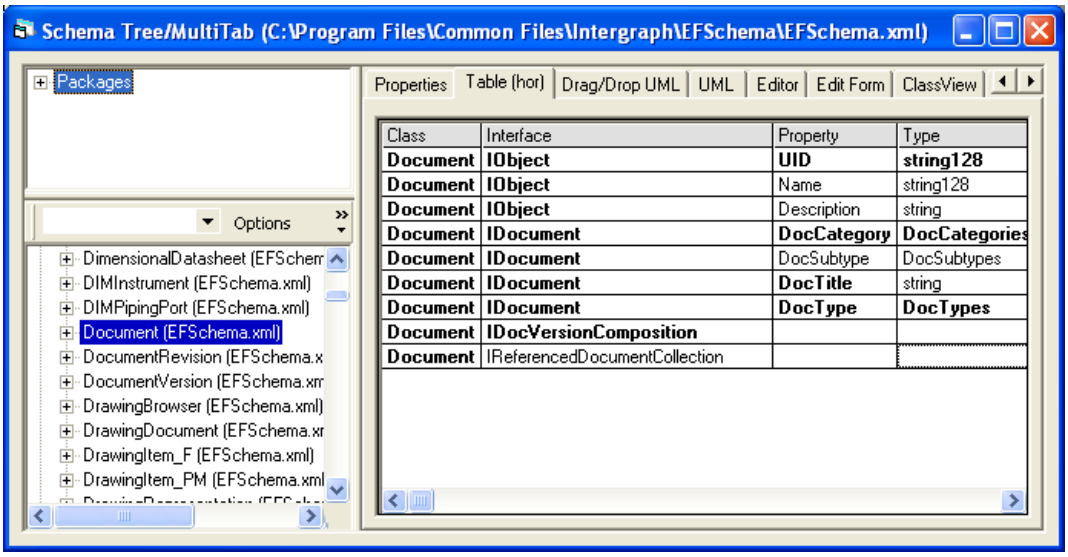
To view the published relationships in the **Editable UML** view, do the following:

1. In the Desktop Client, check out the SmartPlant Schema CMF file.
2. Open the CMF file and the tool map schema file using the Schema Editor.
3. Open a data file for the appropriate integrated application.
4. Generate the UML diagram in the **Editable UML** view using the **Construct Schema View for Data Objects** command.

Tree/MultiTab View: An Example

The **Tree/MultiTab** view is a combination of the standard **Tree** view and several other standard views, including another **Tree** view, a **Table** view, a **Properties** view, a standard **UML** view, a **Drag/Drop UML** view, and other views depending on the selected object.

When you open the **Tree/MultiTab** view, the tabs are blank. After you select an object in the tree on the left side of the window, the tabs update with information on the selected object. Additional views that apply to the selected object in the **Tree** view are also added to the right side of the default tabs when you select an object in the main **Tree** view.



For example, if you select the Document class in the main **Tree** view, the following occurs on some of the tabs:

- The **Properties** tab displays the properties for the Document object in a view that is similar to SmartPlant Foundation.
- The interfaces and properties for the Document class appear horizontally on the **Table (hor)** tab.
- The **Editable UML** tab displays the UML view for the selected object. You can add other objects to the view by dragging them from the **Tree** view and dropping them on this tab.
- The **Viewable UML** tab displays the default UML view for the Document class. This is the same UML view displayed in the Schema **Tree/UML** view.
- The **Edit Form** tab allows you to edit properties for the object you select in the tree view.

To view a file in the **Tree/Multi-tab** view, do the following:

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **MultiTab**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.

4. Click **OK**.
5. Select an object in the **Tree** view to see details for the object on the tabs.
6. Move from tab to tab to see different views of the object.

See Also

Navigating Files and Data in the Schema Editor (on page 101)

View Options

In the Schema Editor, you can set options for the behavior and appearance of tree views, table views, drag-and-drop views, and UML views in the **Options** dialog box.

See Also

Set View Options (on page 123)

Set View Options

1. Click **Tools > Options**.
2. On the **View** tab, select the type of view for which you want to define options.

   TIP The other tabs in the dialog box change depending on the type of view you select. For example, if you select a **Tree** view, the **Tree Options** tab is available.
3. On the **Tree** tab, set options for the display of information in tree views.
4. On the **Editable UML** tab, set options for the behavior and display of information in drag-and-drop UML views.
5. On the **Viewable UML Options** tab, set colors and other display options for information in UML views.
6. On the **Table** tab, set options for the display of information in table views.
7. On the **MultiEdit** tab, set options for editor views with multiple tabs.
8. On the **MultiTab** tab, set options for multiple tab views.
9. On the **Popup** tab, set shortcut menu options that you want to display.
10. Click **OK**.

SECTION 7

Finding Objects in the Schema Editor

In the Schema Editor, you can locate particular objects in the schema, meta schema, tool map schemas, or data files by defining search criteria for the object that you want to find, including the name and description of the object and its UID.

To find an object in the Schema Editor, you can click **Edit > Find** to define search criteria and to limit the scope of your search to the schema, meta schema, tool map schemas, data files, or a combination of these. Using the **Find** dialog box is especially useful if you know you know part of the UID, name, or description of the object that you want to find.

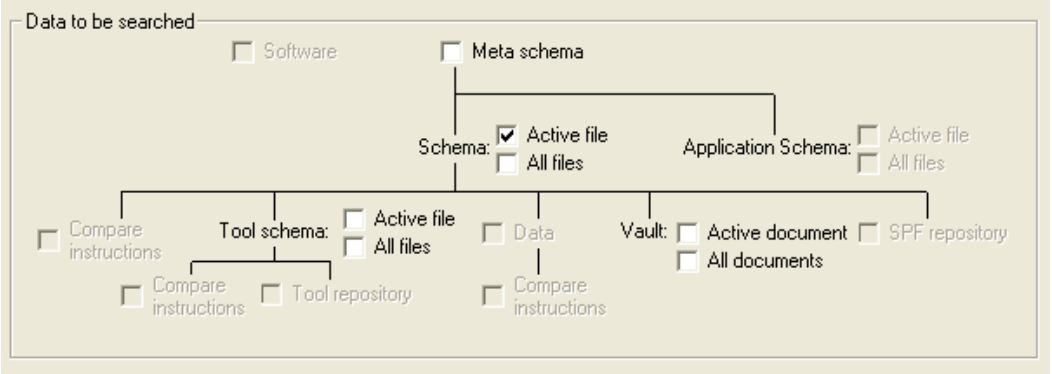
See Also

Find Objects Using Search Criteria (on page 124)

Find Objects Using Search Criteria

1. Click **Edit > Find**.
2. On the **Scope** tab, select the appropriate check boxes to identify what data you want to include in your search.

   For example, if you want to search the meta schema and active schema for a particular object, select the **Meta schema** check box and the **Active file** check box beside **Schema**.



3. On the **Find** tab, enter the criteria that you want to use in your search. The criteria that you can search by are the properties of IObject.
4. If you want the search to be case sensitive, select the **Case sensitive** check box.
5. Click **Options** on the **View Schema** dialog box.
6. On the **View** tab, click the type of view in which you want the software to display your search results.

   TIP Depending on the type of view you select, you can also define view options on the **Tree**, **Table**, **Viewable UML**, and **Editable UML** tabs.

7. Click **OK**.

NOTE If you select specific class definitions for searching, the Schema Editor saves the selected class definitions for your next search.

See Also

Finding Objects in the Schema Editor (on page 124)

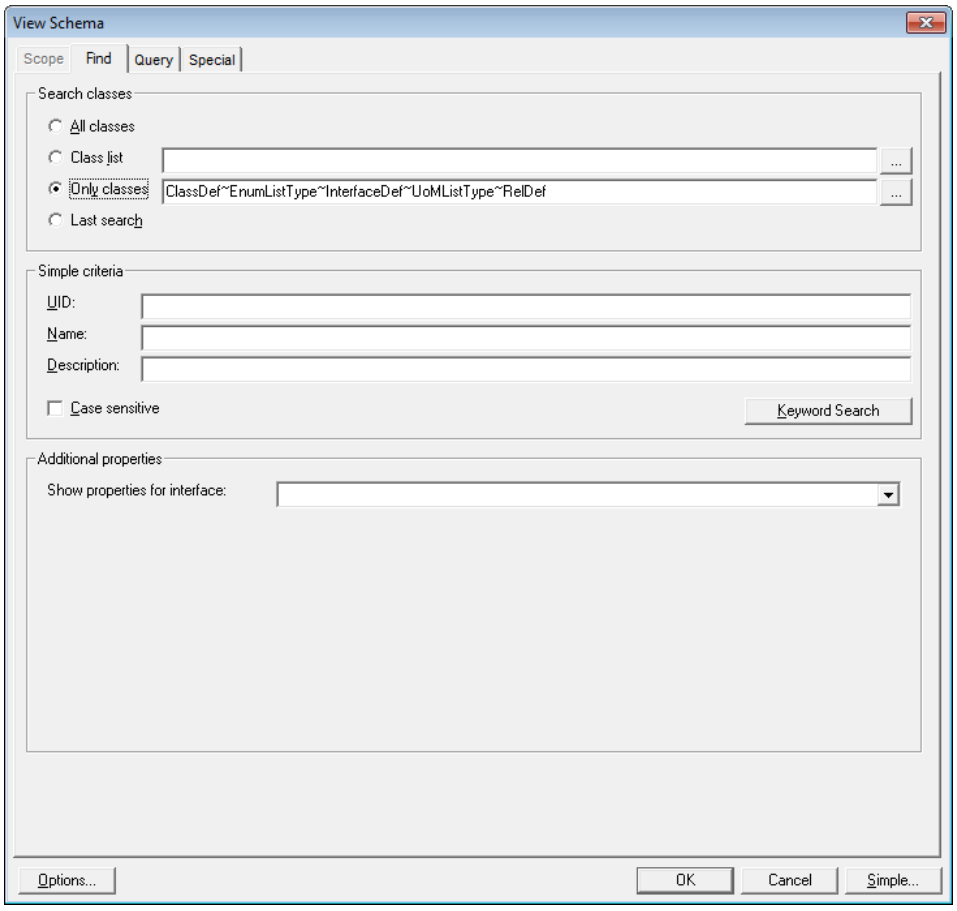
Find Objects by Class

Once the desired information set or sets have been defined, the next step in the query process is to define the classes that are to be searched. For the **View** commands, there is a default set of classes that will be searched. For the **Find** command, the default is to search all classes. If the default set of classes is the desired set, you can begin searching without changing the classes.

If the default set of classes to search is not the desired set of classes to search, then you can change the set of classes by doing one of the following:

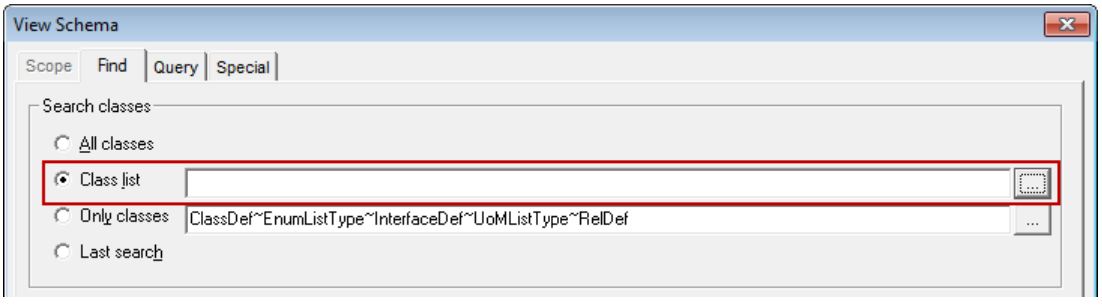
- Select **All classes** on the **Find** tab. This option expands the search to include all classes for the selected information set or sets.
- Select a predefined set of classes from one of the class lists.
- Explicitly specify the classes to search.
- Select **Last search**. This option defines the query to only search the classes used in the last search. The set of classes to be searched are shown to the right of **Last search**.

- Select a named query or specify an interface query from the **Query** tab to query based on that interface rather than on the selected list of classes.

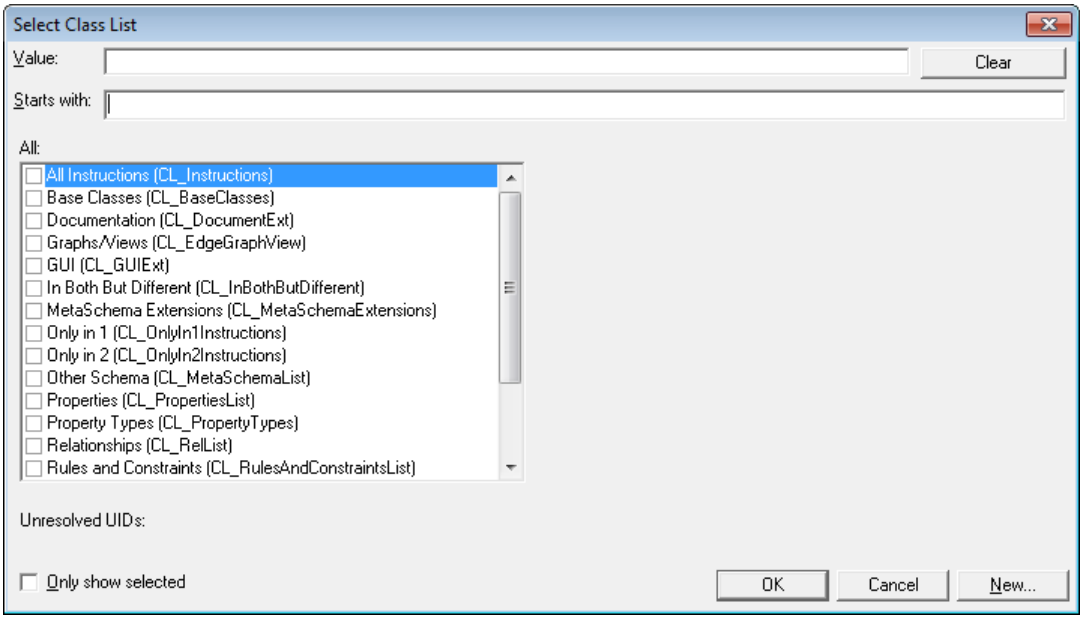


Selecting a Predefined Class List

A number of predefined class lists are automatically loaded into the meta-schema during initialization of the Schema Editor. These lists are defined to identify sets of related classes that contain information that the user might want to search and display. The predefined class lists are displayed by selecting … **Browse** next to the **Class list** box on the **Find** tab.

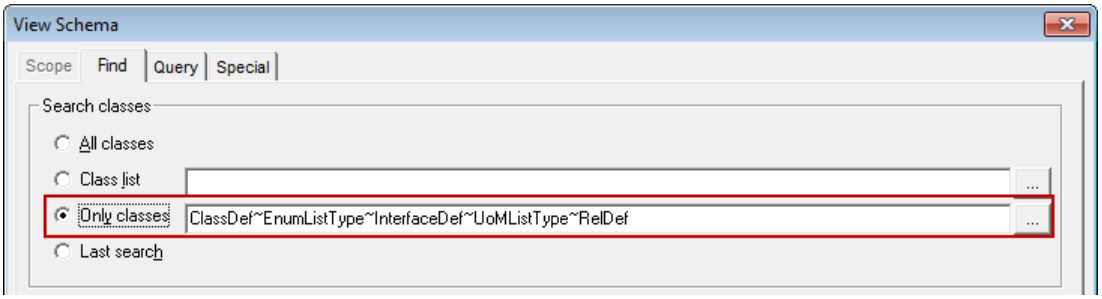


Check the class lists on which to search on the **Select Class List** dialog box.

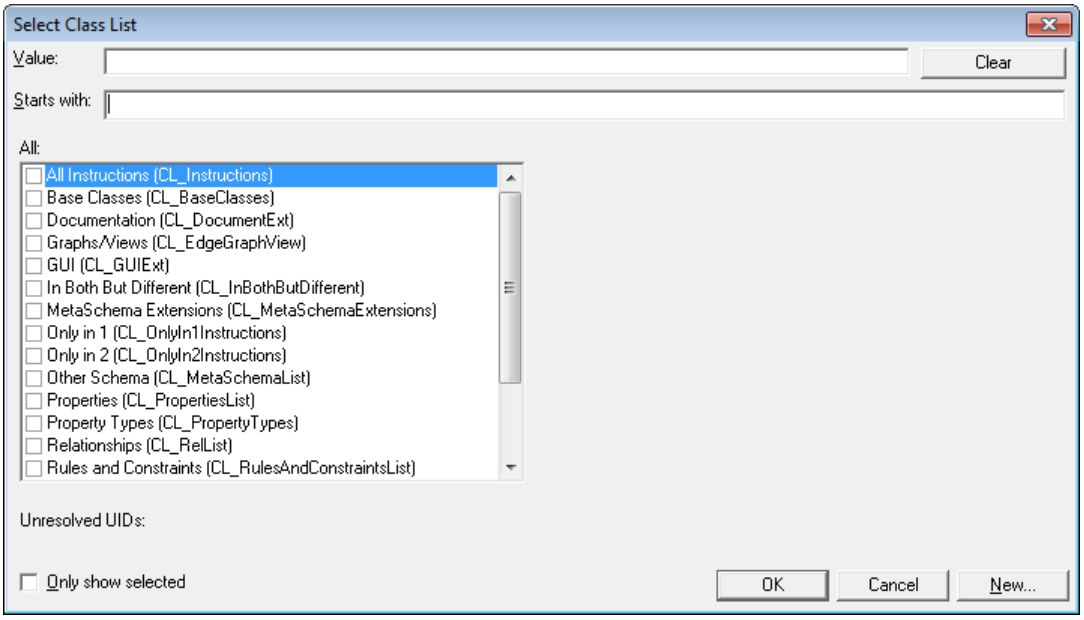


Explicitly Specify the Classes to Search

For the ultimate control of what to search, you can specify which classes are to be searched by select ... **Browse** next to **Only classes.**



In the **Select Class List** dialog box, the user can choose exactly which classes to search, as seen in the following image.



Find Objects Using Simple Criteria

Once the set or sets of information to search and the classes to search have been identified, the criteria to use for the search can be specified. If you do not want to use the default searches, you can use a simple search by doing one of the following:

- Specify criteria for any combination of **UID**, **Name**, and **Description.**
- Select an interface and then specify criteria for any combination of the properties that appear on the **Find** tab.
- Perform a **Keyword Search** by doing a simple web-style query or an advanced web-style query.
- Select a predefined named query from the **Query** tab.
- Select an interface to query from the **Query** tab and then specify a more complicated query criteria.

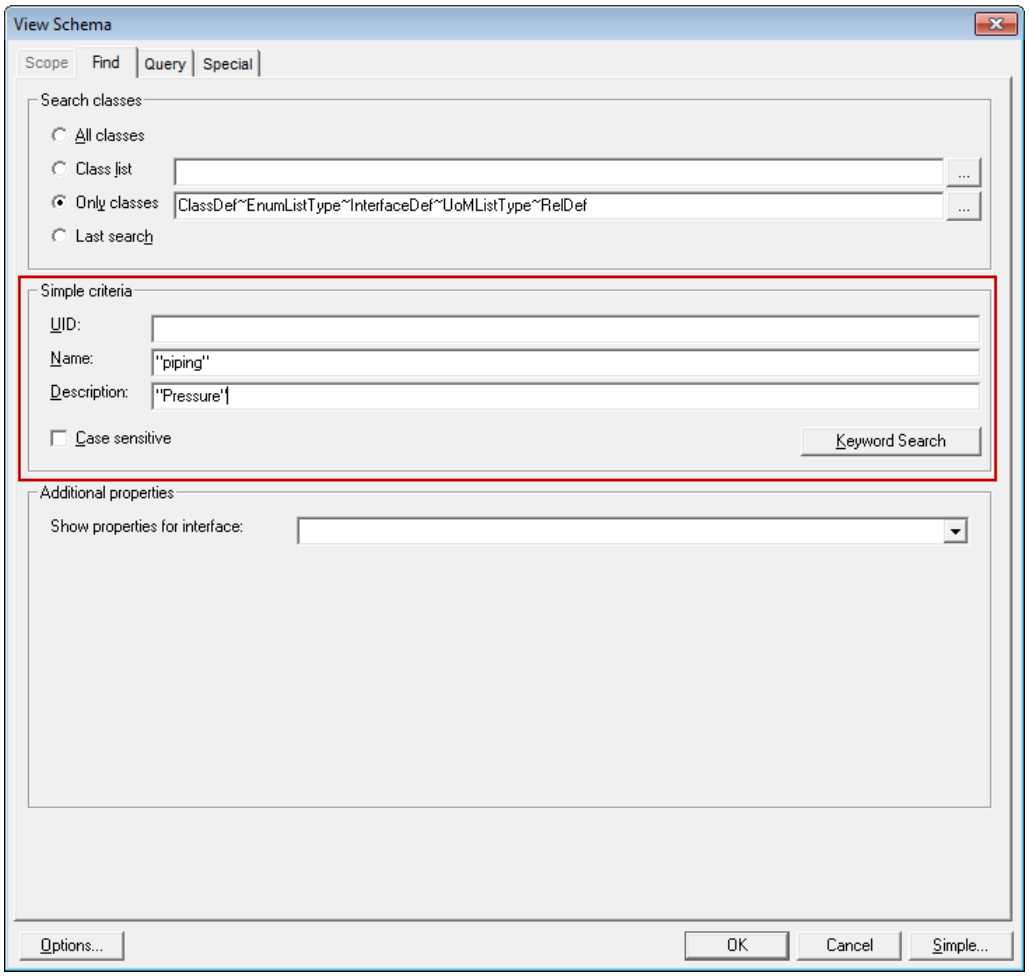
Simple Criteria Using Property Text Fields

On the **Find** tab, the simplest way to specify query criteria is to type the desired criteria in the **UID**, **Name**, or **Description** boxes or select an interface from the **Show properties for interface** list.

If criteria are specified for more than one property, then the criteria for the properties use the **AND** operater. For example, the following entries would result in a query to find all objects that contain “Piping” in their name and “pressure” in their description.

NOTES

- Using astericks in front of and at the end of a search indicates that anything may precede or follow these letters in the name and description searches.
- Criteria specified using the text fields is treated as Like criteria.



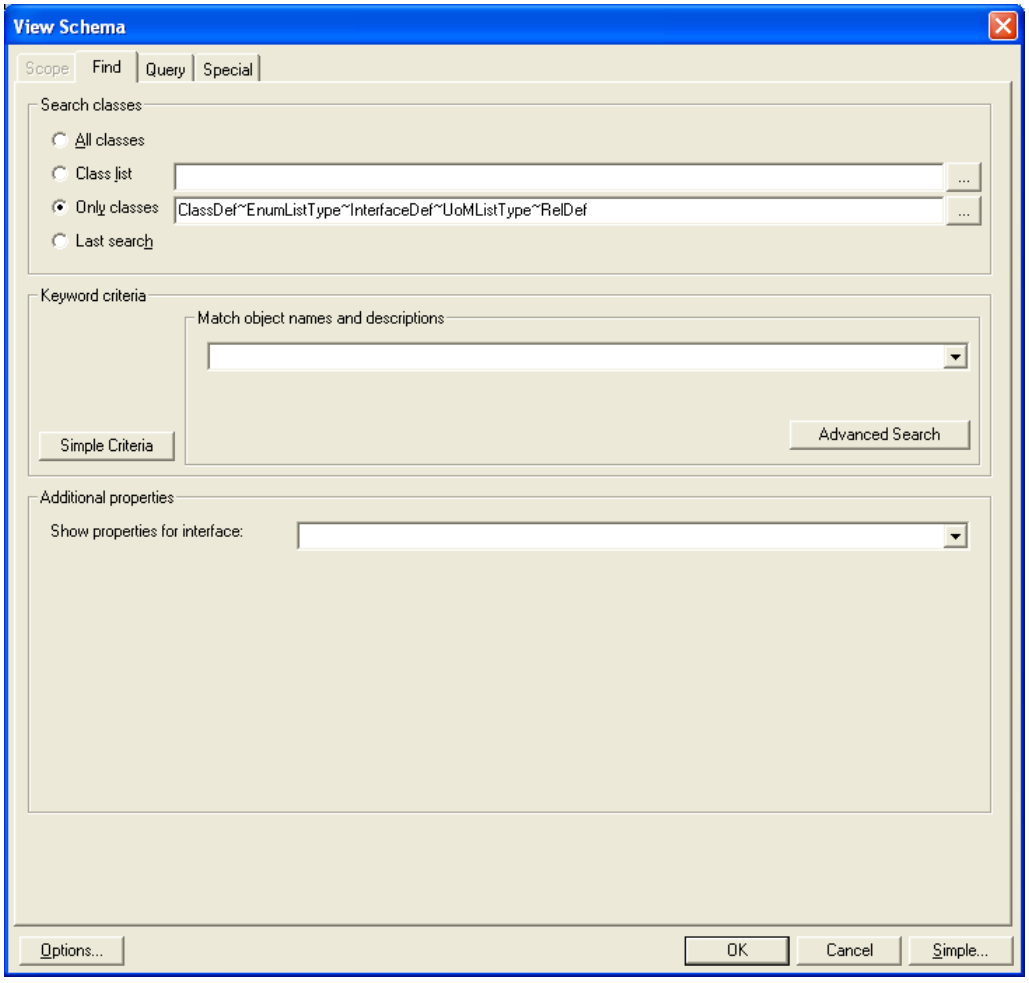
Keyword Search

The Schema Editor supports keyword searching of the Name and Description property values using a Web-like syntax.

Simple Keyword Search

1. Click **Keyword Search** on the **Find** tab to open a single keyword list box.

2. Type the criteria on which to search or select an existing keyword search from the list.



Any keywords entered into this field are used to search the Name and Description property values for all objects of the selected classes for the specified information set or sets. Entering a keyword into this field is the equivalent of entering the same text into the Name and Description fields with the following differences:

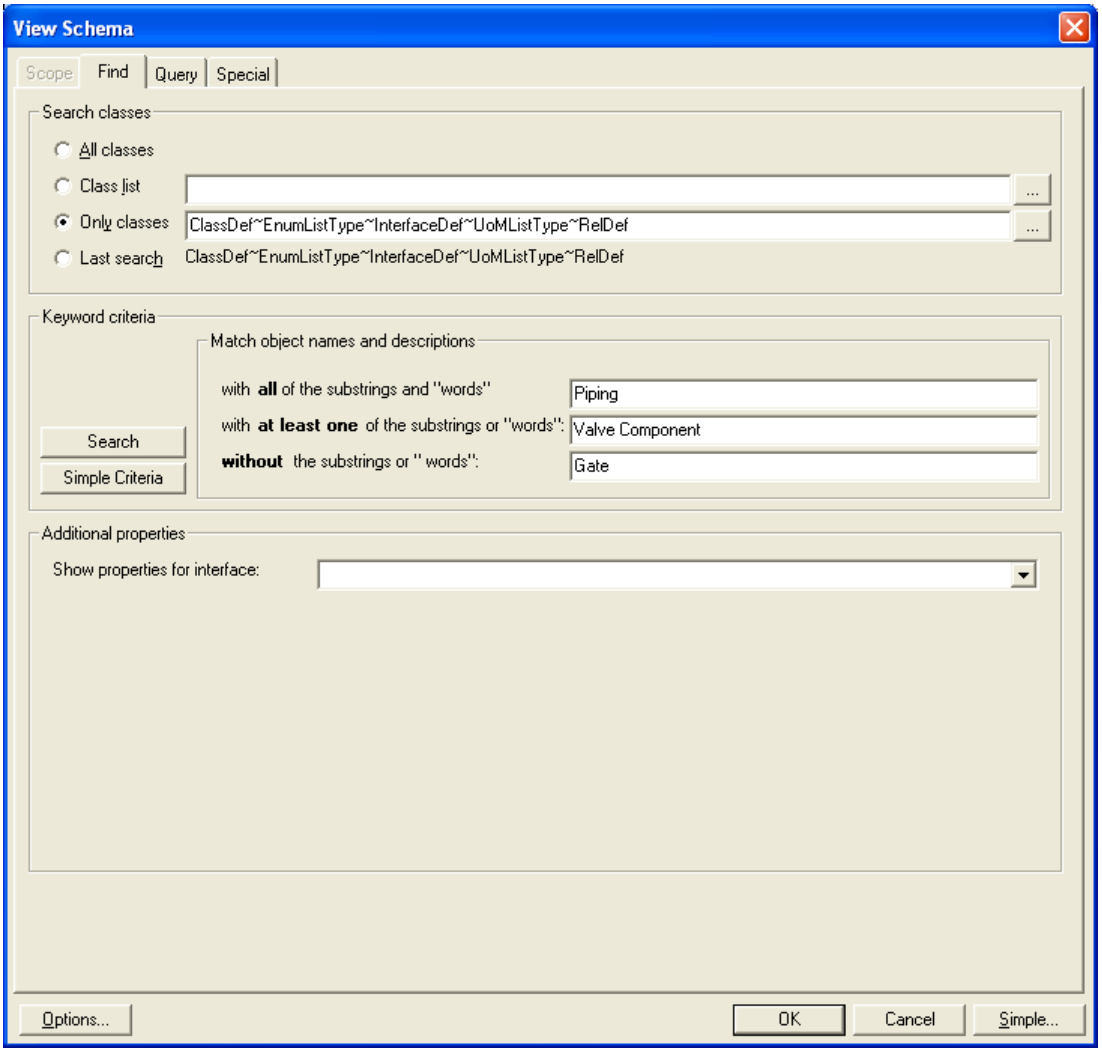
- Keyword searches check both the Name and Description property values.
- Wildcard asterisks surrounding the keywords are not required and, in fact, should not be included as they are treated exactly as any other character.
- Multiple keywords can be entered whereas only one keyword could be specified using the UID, Name or Description test fields.
- The **ALL**, **ANY**, and **NOT** operators can be used to qualify the search to be performed.
- Multiple words in sequence can be specified by enclosing those words inside quotation marks.
- Keyword searches are case insensitive.

- Keyword searches check both the Name and Description property values for each object whereas criteria specified using the UID, Name and Description text fields only searches the specified property value.
- When using the keyword search, it is recommended that you do not use wildcards, such as asterisks. Any asterisks included in the keyword search criteria will be treated like any other character. For example, specifying ***Piping*** as a keyword would only match an object that explicitly contained ***Piping*** (including the asterisks) in its Name or Description property value.
- You can do a search on multiple keywords. For example, type **Piping Component** and the search matches objects containing both **Piping** and **Component** in their Name or Description property values. By default a match on the keyword criteria will only apply if the Name or Description property values contain all the specified keywords (the criteria uses the **AND** operator). However, the user can change that behavior by using the **ANY** and **NOT** operators.
- Keywords using the **ANY** operator are treated like criteria using the **OR** operator. For example, any object that matches any of the keywords after **ANY** will be considered to satisfy the criteria. Therefore, specifying a keyword criteria of **Any Piping Component** attempts to match objects containing either Piping or Component in their Name or Description property values.
- Keywords using the **NOT** operator in the criteria identify keywords that are not to be found in the Name or Description property values. For example, a keyword criteria of **Piping Not Component** attempts to match objects that contain **Piping** in their Name or Description values but do not include **Component** in that property value.
- The **ALL** operator is also a special keyword. Its use defines that an object must match all of the specified keywords in the criteria. There is an implied **ALL** at the start of any keyword criteria and, therefore, normally **ALL** is not specified. For example, the keyword criteria **Piping Component** and **All Piping Component** are equivalent.
- **ALL, ANY**, and **NOT** operators can all be specified as part of the keyword criteria. For example, the keyword criteria **Piping Any Valve Component Not Gate** specifies that a matching object must include the **Piping** keyword, must include either **Valve** or **Component** and must not include **Gate**.
- Since **ALL, ANY**, and **NOT** are special keywords, if the keyword criteria should include one of these special keywords, they must be enclosed within quotation marks (“”). The keyword search criteria **“All” Piping** attempts to match objects that contain both the **All** and **Piping** keywords in their Name or Description property values.
- If a search matching multiple words in sequence is desired, then those words should be enclosed inside quotation marks (“”). For example, the keyword search **“Piping Component”** only matches objects that contain **Piping Component** in their Name or Description property values.
- Keyword searches, unlike criteria specified using UID, Name, and Description, is case insensitive. Therefore, while a Name criteria of ***Piping*** only matches an object with a Name value that contains **Piping** with a capital **P**, a keyword criteria of **Piping** will match a Name or Description value that contains a lowercase **p** (such as piping).

Advanced Keyword Search

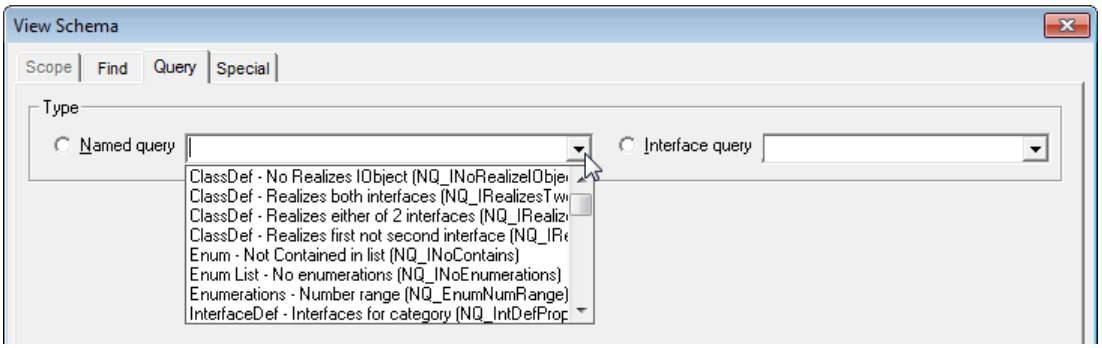
Advanced keyword searching is also available using the **ALL, ANY**, and **NOT** operators. Click Advanced and separate search boxes appear for these specific search operators, one for

specifying the **ALL** criteria, one for specifying the **ANY** criteria, and one for specifying the **NOT** criteria. For example, searching on **Piping Any Valve Component Not Gate** on the **Advanced** group would appear as follows.

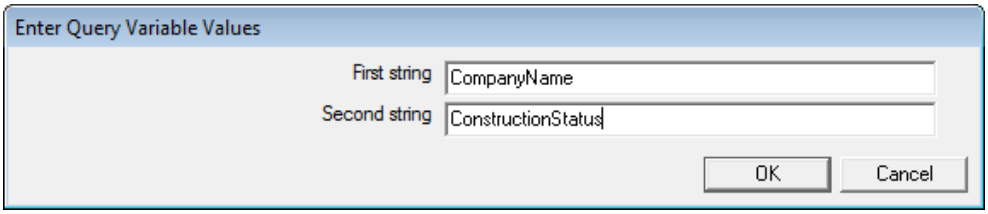


Find Objects Using Predefined Named Query

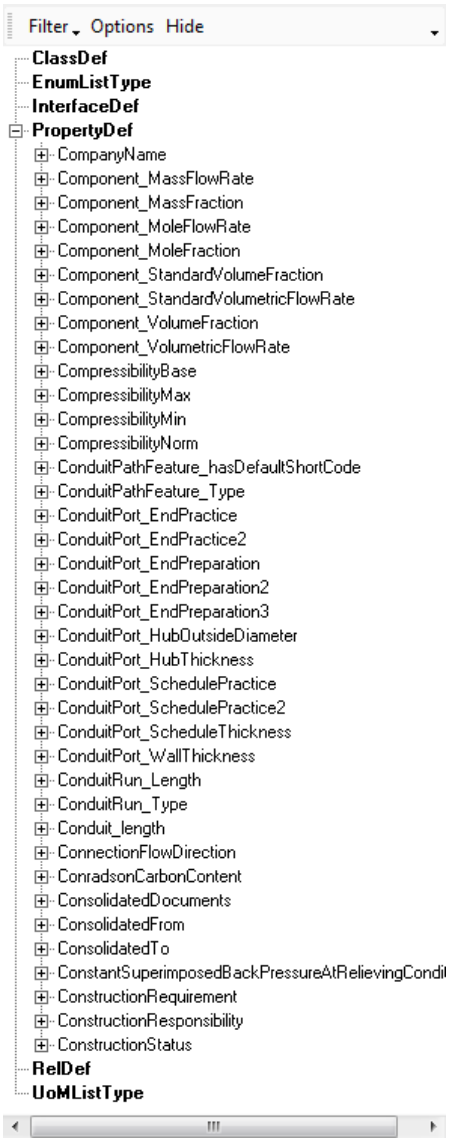
A number of useful queries are predefined and can be selected from the **Named query** list on the **Query** tab of the **View Schema** dialog box.



Many of the named queries come with a requirement for the user to specify a value for one or more of the individual criteria within the named query. For example, when you query on **PropertyDef – Between two strings**, the **Enter Query Variable Values** dialog box appears. For this type of Named query, you must type the strings that bound the matching property names information on which to query; for example, specifying that only property names between **CompanyName** and **ConstructionStatus** are to be matched.



The properties appear in the **Tree** view.



Find Objects by Interface

On the **View Schema** dialog box, you can query on a specific interface using the Interface query list on the **Query** tab. When you select Interface query, options appear that allow you to perform the following types of querieis:

- **Simple** - Specify property-based criteria to query using the AND operator.
- **Full** - Specify complex criteria using the constraint language.
- **Filter** - Define the criteria using a spreadsheet user interface.

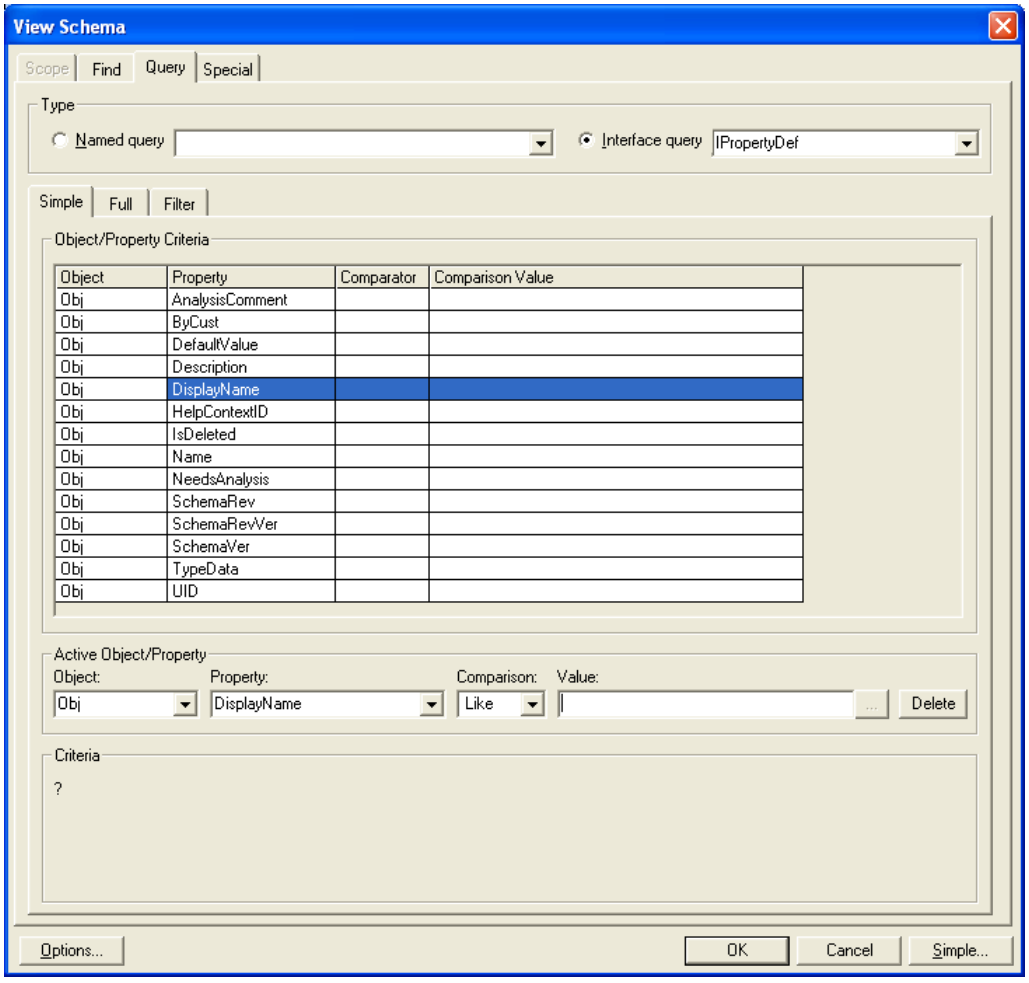
Simple Interface-Based Query

Use the **Simple** tab for a relatively simple way to define fairly complex query criteria. When you select an interface definition and then select the **Simple** tab, a table appears containing a row for each property exposed by the specified interface or an interface directly or indirectly implied by that interface. You can define criteria to be applied to that specific property during a query.

When you select a row in the table, additional query options appear under **Active Object/Property** group at the bottom of the **Simple** tab. These options are populated based on the selected row. You can then change the comparison type and provide the value for the comparison.

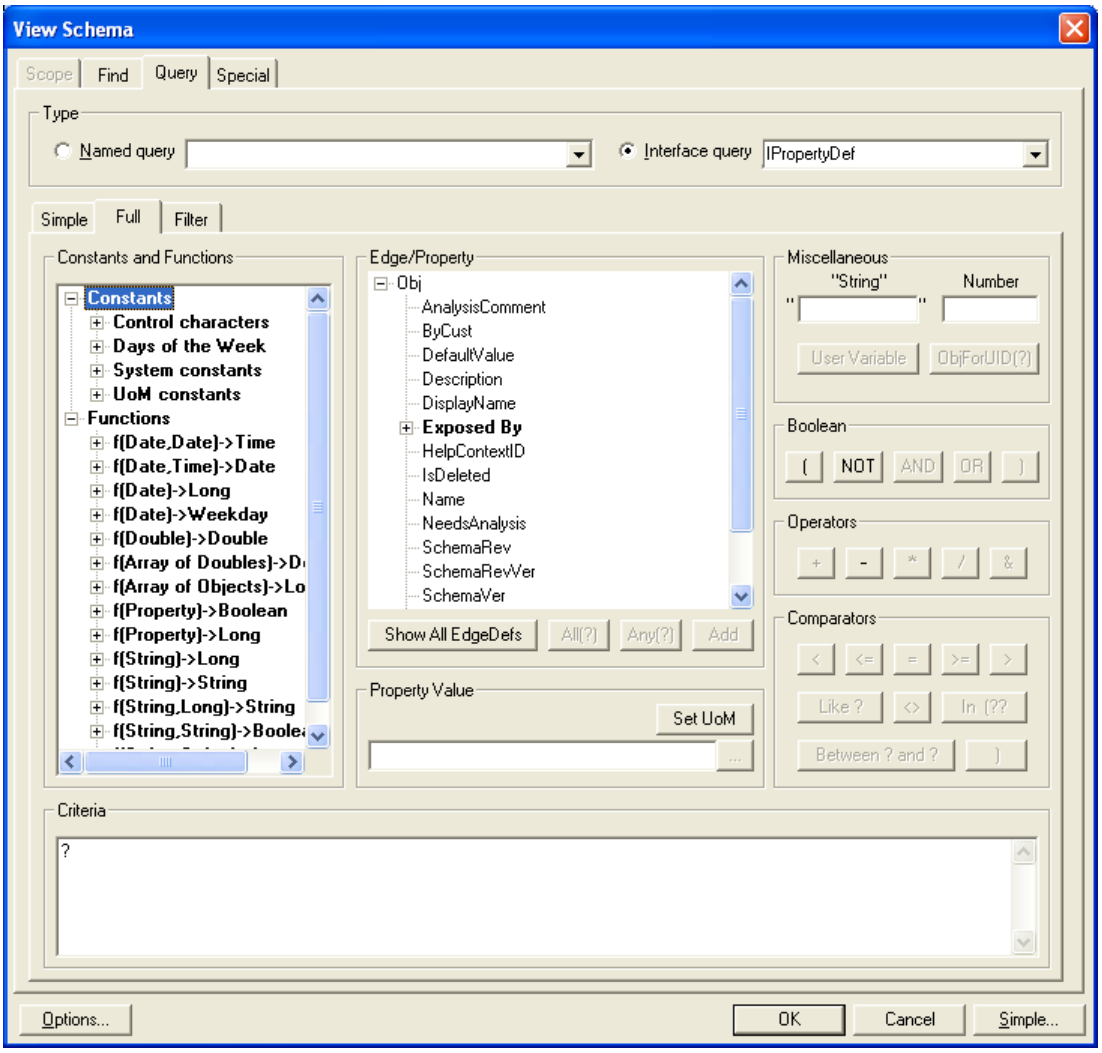
1. Select an object row in the **Object/Property Criteria** table.
2. Under **Active Object/Property**, select a comparison operator under the **Comparison** list.
3. Click ... **Browse** to provide a **Value**.

Your criteria appear in the table and in the **Criteria** group.



Full Interface-Based Query

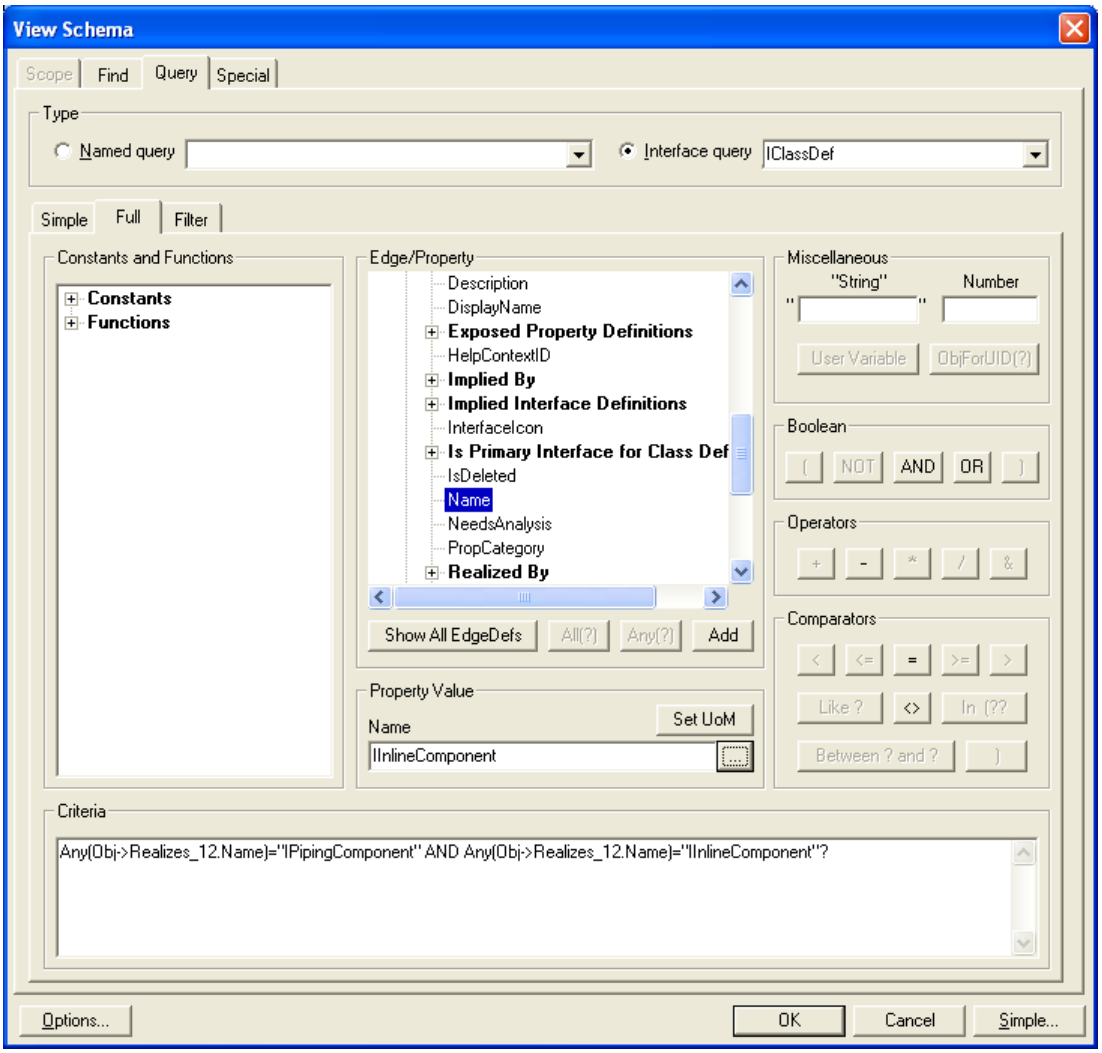
You can define queries using the constraint language on the **Full** tab. The constraint language is a robust language geared towards the definition of constraints and queries and, as such, should be powerful enough to handle the most complex desired queries. For example, if a query is to traverse one or more relationships as part of the criteria, the only way (with a few exceptions) to define that query is using the **Full** query.



Query for Existence Using the Constraint Language

The following examples explains how to query for the existence of a relationship to a specific object is covered.

The **ANY** operator along with the specified equivalent comparator and value should be used to query for the existence of a particular related object across a to-many relationship edge. For example, to query for class definitions that realize both IPipingComponent and IInlineComponent, the criteria in the image below should be specified.



Query for Non-Existence Using the Constraint Language

Querying for non-existent objects or relationships between objects is done using either the **Count()** function or the **ALL** operator in conjunction with a negative test depending on whether the query is for the non-existence of a relationship or the non-existence of a relationship to a specific object.

If the query is to locate objects that are missing a specific relationship, then do the following:

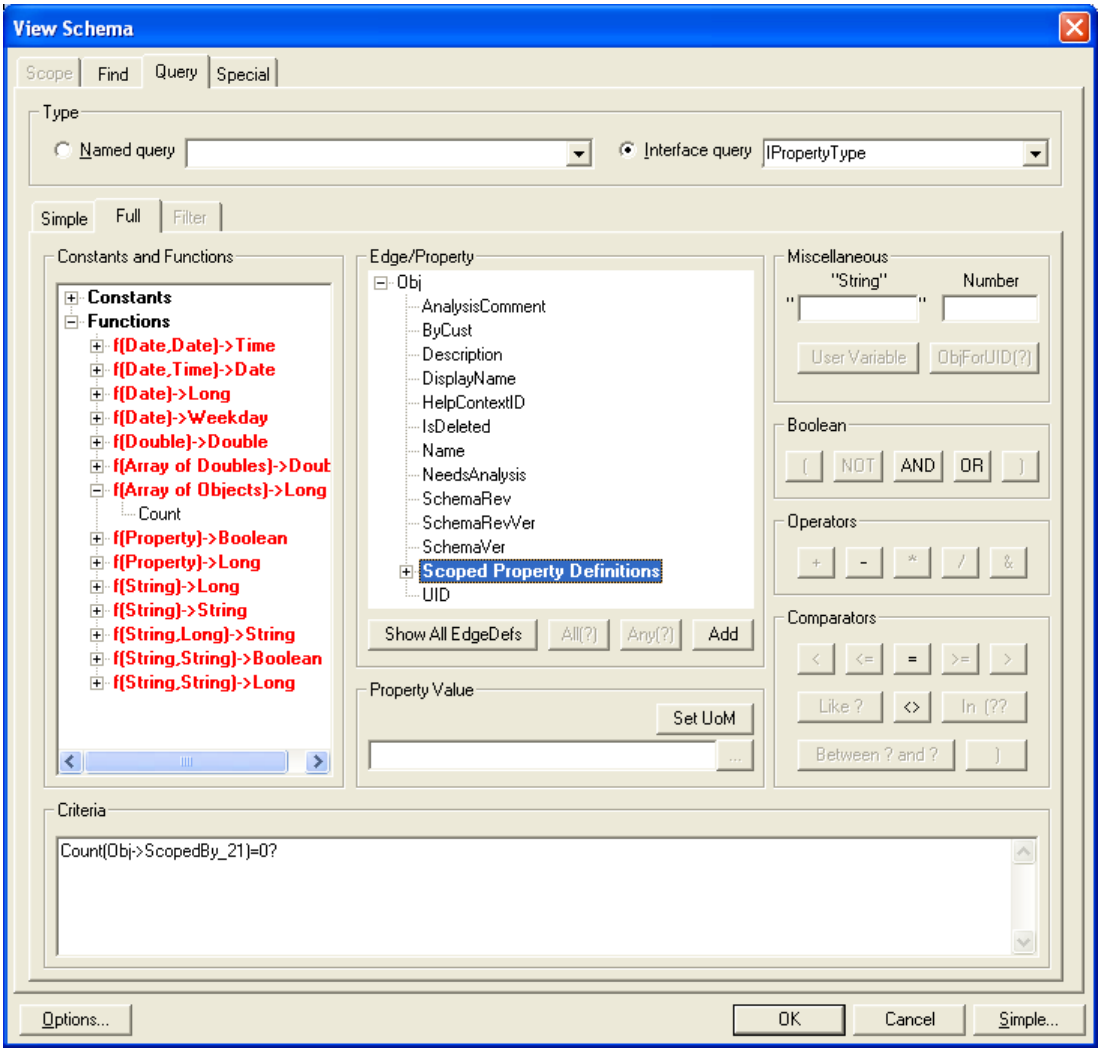
1. Select the **Count** function from the **Functions** list in the **Constraints and Functions** tree.
2. Select the specific relationship edge from the **Edge/Property** tree and provide a value of **zero (0)**. For example, to find all property types for which there are no property definitions that are scoped by that property type, do the following:

a. Select **IPropertyType** from the **Interface query** list.

b. Select Count from the **Constraints and Functions** tree.

c. Select the **Scoped Property Definitions** from the **Edge/Property** tree.

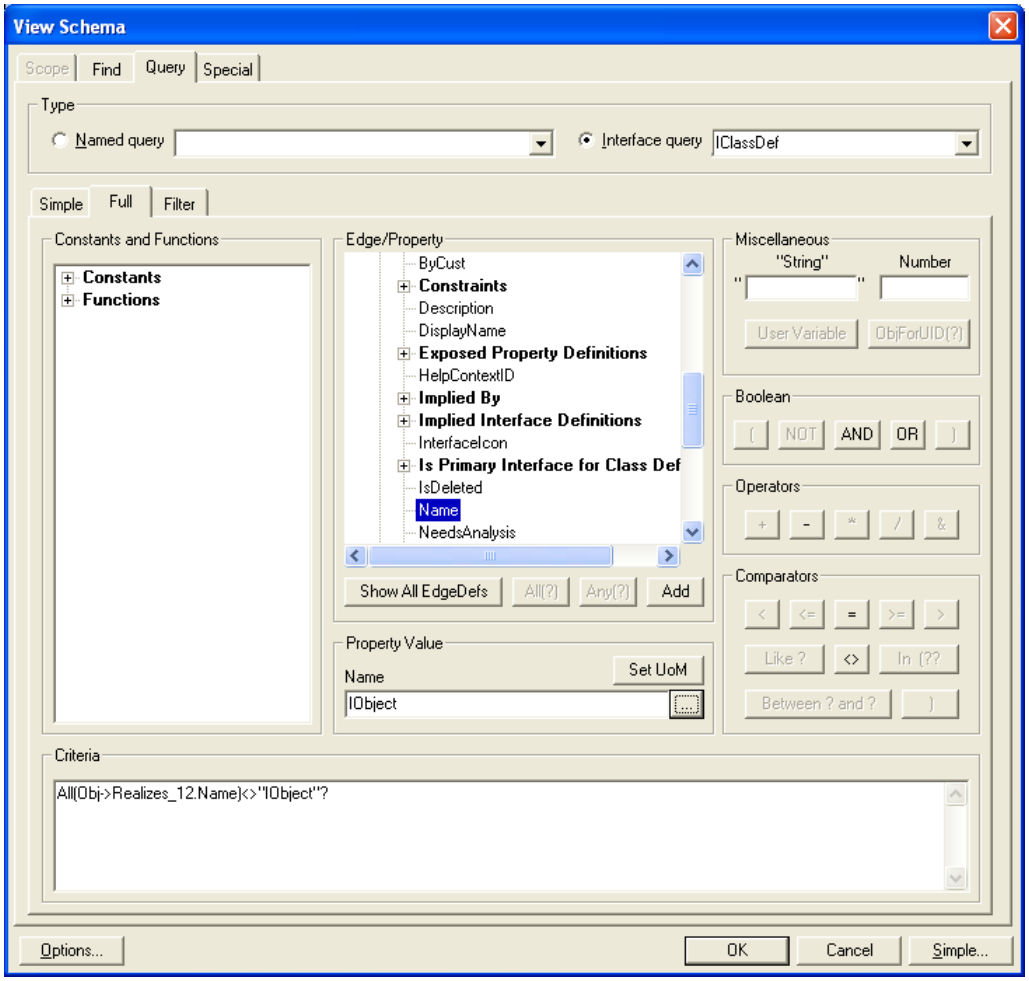
d. Click **=** in the **Comparator** group and provide a value of **zero (0)**.



If the query is to locate objects that are missing a relationship to a specific objec, use the **ALL** operator in conjunction with a negative comparator. For example, to locate all class definitions that don't realize IObject, do the following:

1. Select **IClassDef** from the **Interface query** list.
2. Select the **Name** property across the Realized Interface Definitions in the **Edge/Property** tree, and then click **All(?)**.
3. Click **<>** in the **Comparator** group to indicate that all realized relationship definitions should not be IObject.

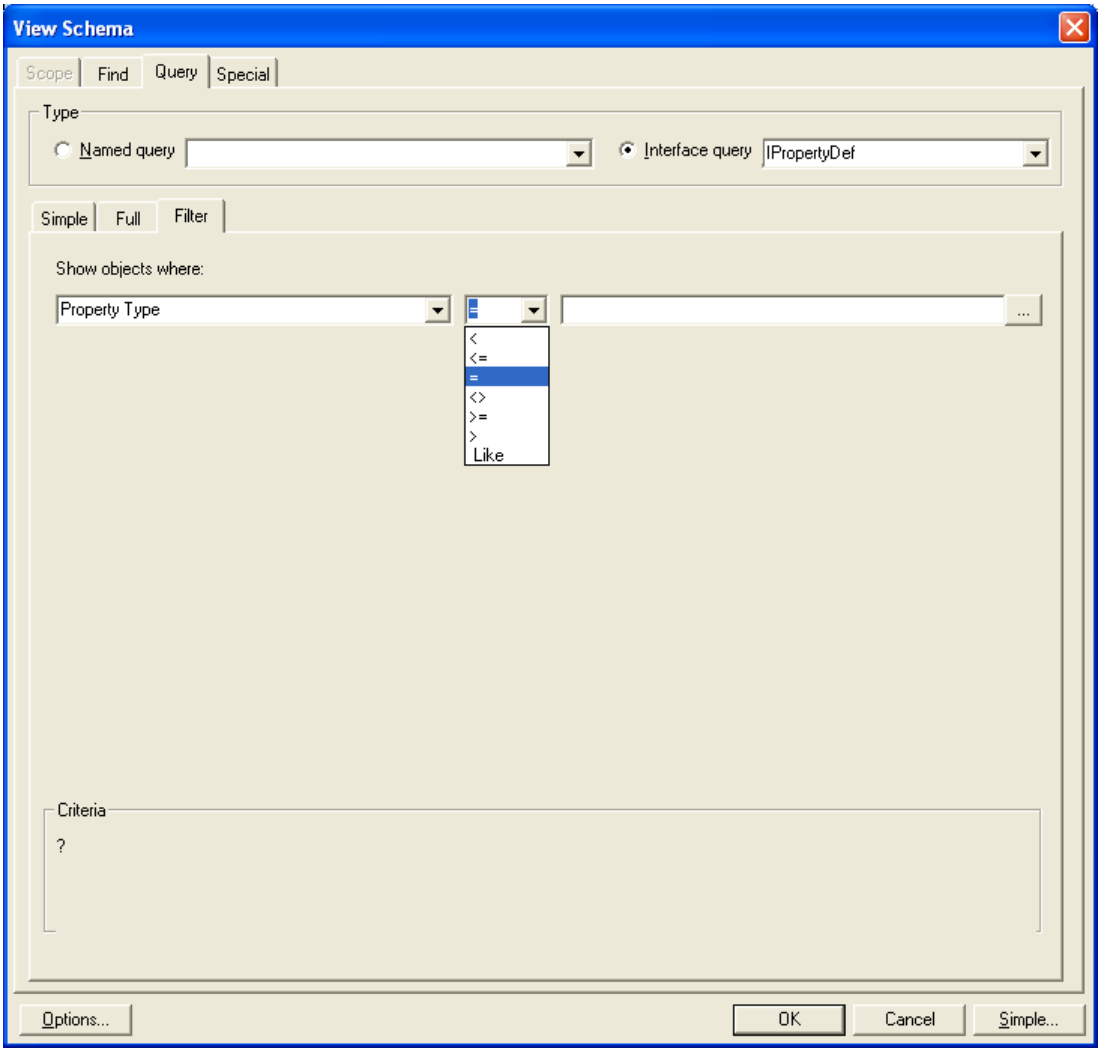
4. Select **IObject** as the realized interface definition in the **Property Value** box.



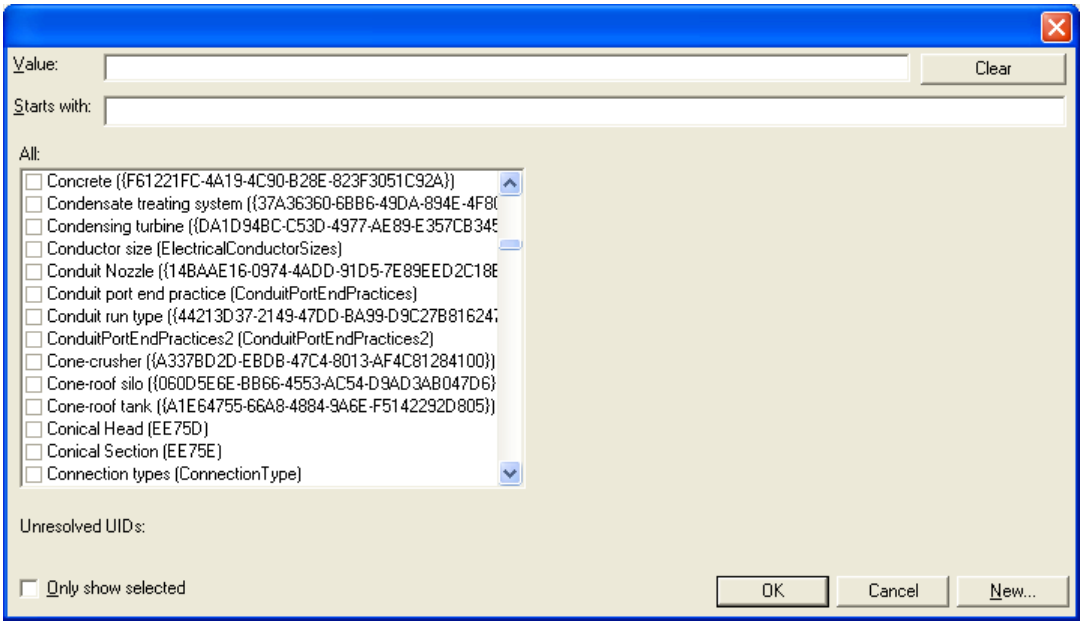
Filtered Interface-Based Query

If a view definition has been defined for the selected interface, then the **Filter** tab is available and uses a spreadsheet-like interface to define the query criteria.

If a view definition has been defined for the selected interface, then the set of view properties for that view definition will be available from the first list view under **Show objects where**. Select the comparator from the middle list view.

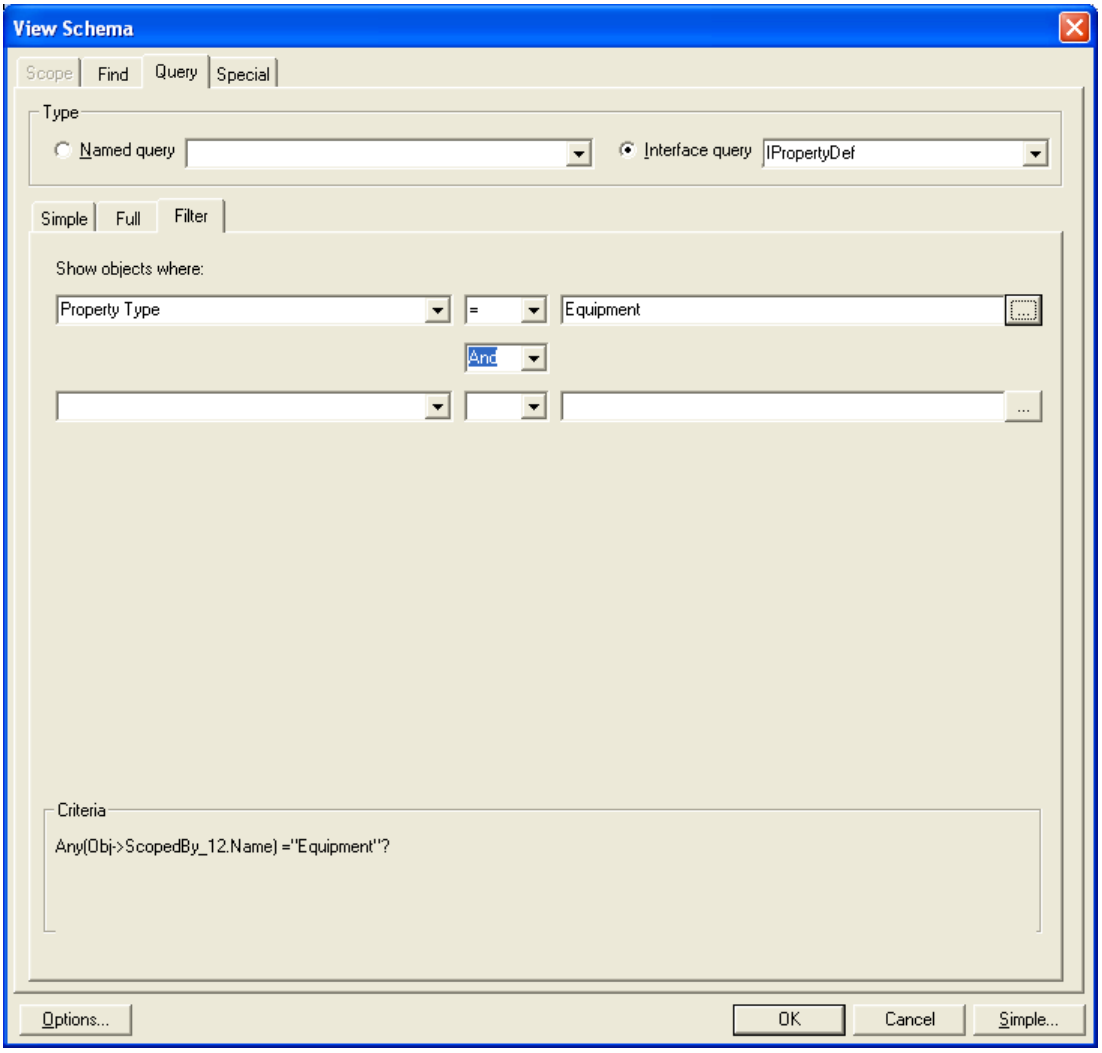


Enter a value in the text box to specify the value for the comparison. If the selected property is enumerated or has other information available, then ... **Browse** is available and you can select the appropriate value from the dialog box that appears.



Since view properties can be defined across edges, the criteria specified on the **Filter** tab may involve the traversal of relationships as part of the criteria. This is the only way, other than using the constraint language on the **Full** tab, of specifying criteria that involves relationship traversal.

Once you provide a value for the specified property and comparator, another set of options appear that can be used to specify additional criteria. In addition, an intermediate operator list appears that allows you to specify whether the criteria is to use the **AND** or **OR** operator.



SECTION 8

SmartPlant Schema Documentation

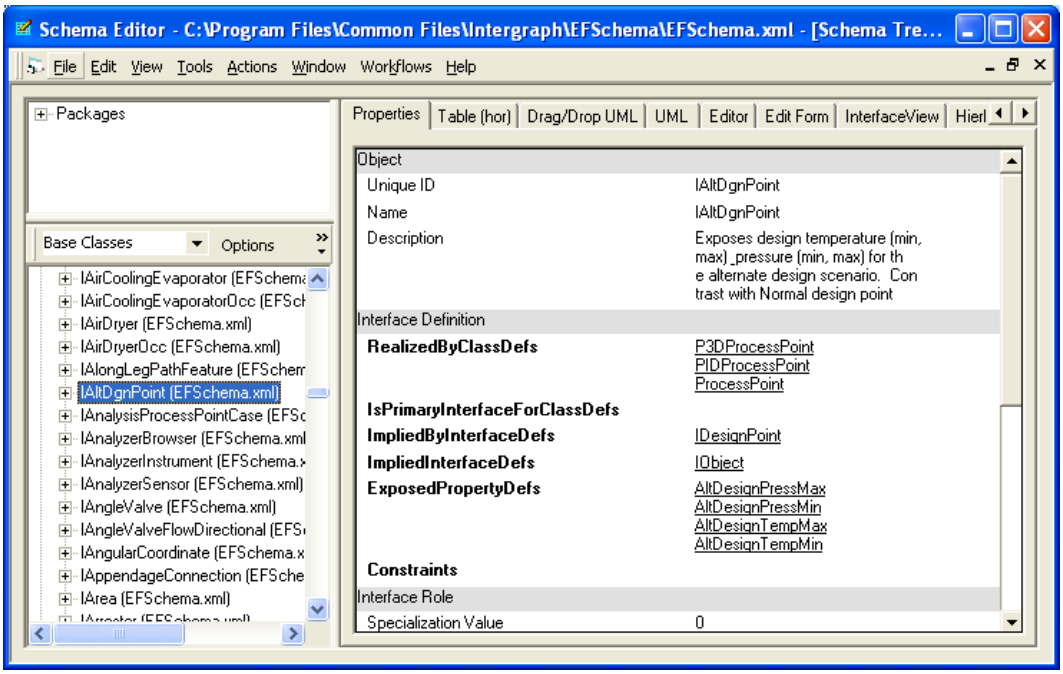
Before you modify the SmartPlant schema in any way, you must understand the objects that the SmartPlant schema contains and the rules that apply to those objects. However, the Schema Editor documentation does not contain topics that cover the objects in the SmartPlant schema because the schema is constantly undergoing changes.

The SmartPlant schema contains internal documentation about many of the objects contained in the schema. The **Description** property contains brief documentation for some of the class definitions, interface definitions, relationship definitions, property definitions, and units of measure in the SmartPlant schema.

Not all objects in the SmartPlant schema have documentation defined for them. For example, concrete objects, such as pumps, are not typically documented within the SmartPlant schema because most users know what pumps are. Some abstract objects, like streams, have a description if a description was available when the schema was modified.

You can use the **Tree/Multi-Tab** view in the Schema Editor to view the existing documentation for objects in the SmartPlant schema. The **Description** field on the **Properties** tab in the **Tree/UML** view allows you to view information about an object selected in the tree view.

For example, the following graphic displays the description for the IAltDgnPoint interface in the **Tree/Multi-Tab** view.



See Also

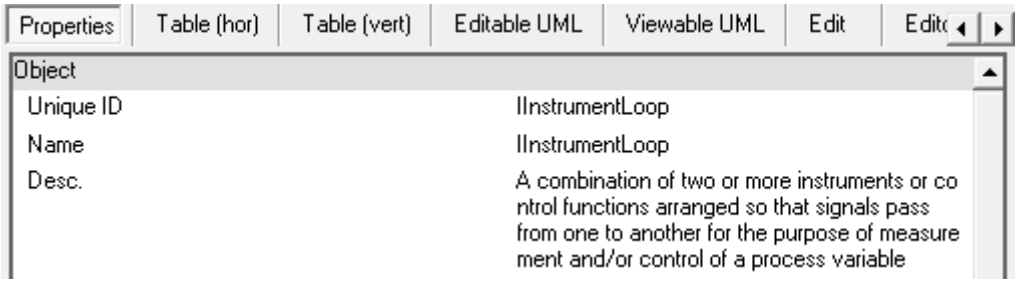
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
View SmartPlant Schema Documentation (on page 145)

View SmartPlant Schema Documentation

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Tree/MultiTab**, and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
4. To view the description for an object in the schema, expand the appropriate node for that object in the bottom tree view in the **Schema Tree/MultiTab** window.
5. Click the object for which you want to see a description.

   TIP If a description for the object exists in the SmartPlant schema, the description appears on the **Properties** tab under **Object**.



   Not all objects in the SmartPlant schema have descriptions defined for them.

NOTE You can use the drop-down list above the **Tree** view to change the **Tree** view contents. For example, if you want to only view property definitions in the **Tree** view, select **Properties** in the list. If you want to see class definitions, interface definitions, relationship definitions, enumerated list types, and UoM list types, select **Base Classes** in the list.

See Also

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
SmartPlant Schema Documentation (on page 144)

SECTION 9

Extending the SmartPlant Schema

The delivered SmartPlant schema contains many objects and relationships. However, you may need to add new objects and relationships to meet the needs of your company.

To modify or extend the SmartPlant schema, you can use the Schema Editor.

IMPORTANT Even though you can edit any .xml file in a text editor like Notepad, Intergraph recommends that you use the Schema Editor for all SmartPlant schema modifications. If you do view the SmartPlant schema in a text editor, be sure that you do not modify the schema version number. If the schema version number in the .xml file is different from the version of the delivered schema instructions file, schema upgrade may have errors or may fail.

See Also

Create a schema extension file (on page 207)
Set Dependencies for a Schema File (on page 95)

Naming Conventions in the SmartPlant Schema

Before you begin modifying the SmartPlant schema, you need to understand the naming conventions used for objects in the schema. The following naming conventions are used in the schema:

- Interface definitions are prefixed with "I". For example, IPump is an interface definition that expresses the role of a "pump."
- Many class definitions are prefixed with information about the component schema that they are associated with. For example, EQDStrainer has a Componentization relationship with the EquipDataSheet component schema.
- Properties may be prefixed with the name of the interface definition that exposes them. For example, Instrument_FluidType is exposed by IInstrument. This is called *denormalization* of an attribute.
- Some relationship definitions are named according to their patterns, such as collection or composition. For example, PBSItemCollection is a collection, and PipingPortComposition is a composition. For more information about patterns, see *Patterns and Abstract Roles* (on page 147).
- Primary interface definitions for concrete objects generally end in "Occ," which is "the occurrence of an object." For example, the primary interface definition for PIDInstrument is IInstrumentOcc. For more information about occurrences, see *Patterns and Abstract Roles* (on page 147).
- Because class definitions, interface definitions, and property definitions can be implemented from within a programming language, these objects must not include spaces nor the following special characters in their names: !@#$%^&*()+=- []\{}:;'"?<>./.

Patterns and Abstract Roles

A common theme of data modeling is to discover similar concepts and develop a pattern that may then be applied whenever another similar concept occurs. The SmartPlant schema has many underlying patterns that can be identified and used by schema developers.

Abstraction Patterns

The abstraction patterns are based on abstracting one or both ends of a relationship. This abstraction eliminates direct dependencies between interfaces and promotes a flexible object model.

Collection Pattern

A collection pattern is used for many-to-many relationships in which a collection interface contains collection members as shown below.



If abstraction is done on the left side of this relationship, the interface on that side is *I<member type>Collection* where <member type> identifies the type of members in the collection, such as IModelItemCollection for a collection of ModelItems. The relationship is named the same as the interface with the leading *I* removed.

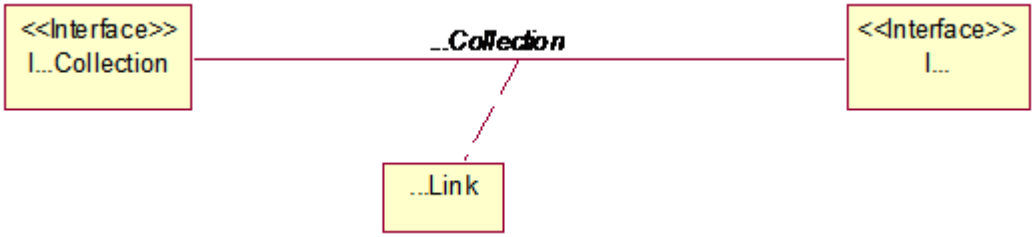


If abstraction is done on the right side, then the interface on that side is named *I<collection type>Member* where <collection type> identifies the type of collection, such as IInstrumentLoopMember. The relationship is named based on the member interface.

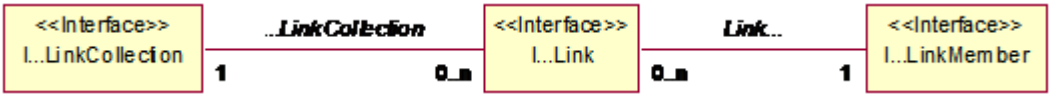


LinkCollection Pattern

If a collection pattern has a link class, such as the following example:

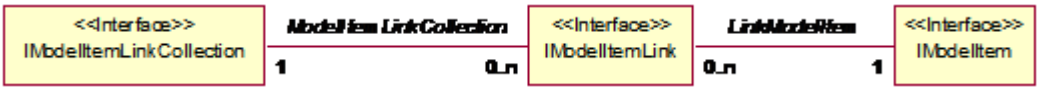


then the pattern is as shown below:

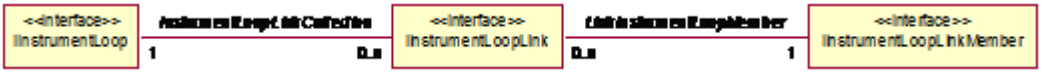


In this pattern, which is typical for many-to-many relationships with link attributes, the cardinalities of both relationship definitions are one-to-many with the many side existing on the intermediate link class. The various interfaces involved are abstracted similarly to that for the collection pattern with the addition of the word *Link* before *Collection* and before *Member*. The intermediate interface is named based on the type of members, such as IModelItemLink for a collection of ModelItems.

If the abstraction is on the left side of a collection pattern, the interface name on the left is *I<member type>LinkCollection* and the left relationship is same name without the leading *I*. In this case, the *I...Link* interface is *I<member type>Link* and the relationship on the right is *Link<member type>*.



If the abstraction is on the right side, the same rules apply as for the collection pattern. The interface on the right becomes *I<collection type>Member* and the relationship and link interface are named logically following the same rules.



Assembly Pattern

An assembly relationship is a special type of collection relationship in which a particular collection member can belong to, at most, a single collection instance.



If abstraction is on the left side, then the relationship name and interface on the left are named based on the <member type>.



If the abstraction is on the right side, then the relationship name and the interface are named based on the type of assembly.



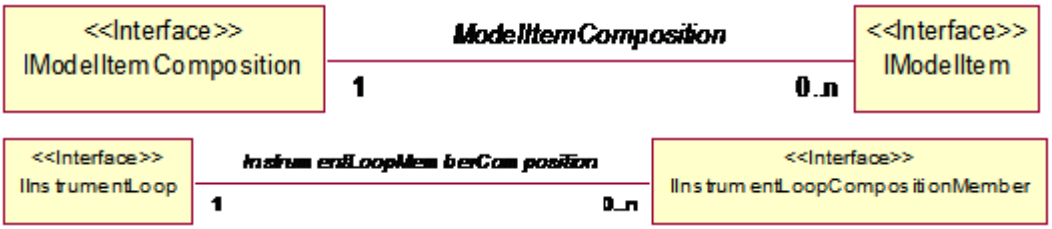
Composition Pattern

The composition relationship is a specialization of the assembly relationship in which the cardinality on the left side is one. In this relationship, the object or interface on the right side

cannot exist without a relationship to an object on the left side. Specifically, if the object on the left-hand side is deleted, the objects on the right-hand side must also be deleted.



The naming of the interfaces and relationship for this pattern is the same as for the collection and assembly patterns.

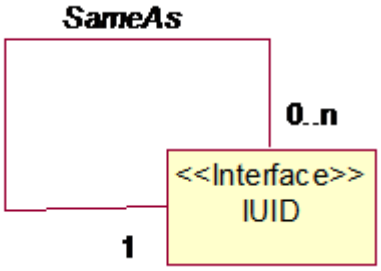


SameAs Pattern

Two approaches can be used to indicate that an object is the same object as an object previously loaded into SmartPlant Foundation:

- Publish the object with the same UID.
- Publish the object with a SameAs relationship to the object for which it is the same object.

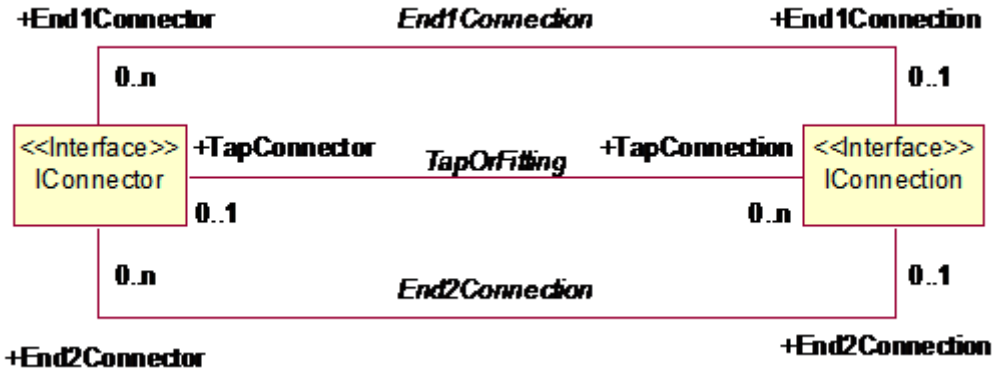
The SameAs pattern is used to indicate this latter case.



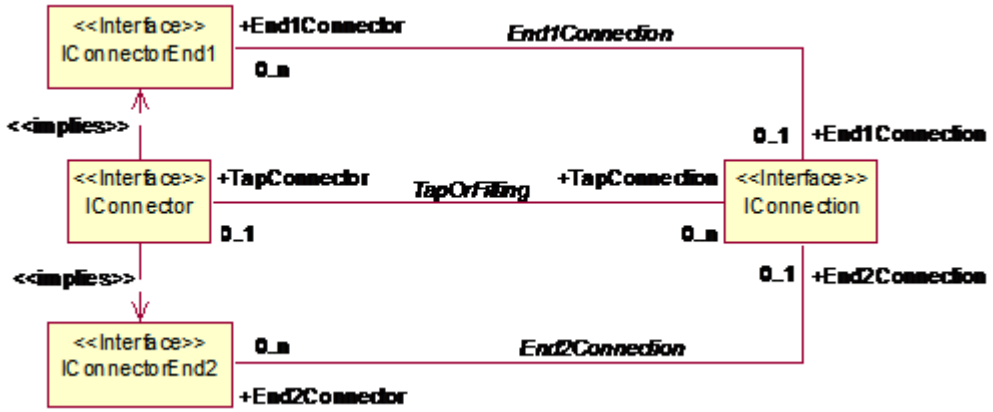
The SameAs pattern is not a pattern since there is just the one case involving the IUID interface. However, all relationship definitions in the SmartPlant schema are classified as being an instance of a particular pattern, and the SameAs relationship definition is not an exception.

Connectivity Pattern

The connectivity pattern can have one connection at each end and zero or more TapsOrFittings in between. Each connection may be an end point for more than one connector, or it can be a tap for one connector and an end point for another connector. Alternatively, a connection can be a fitting for the connector.



To add information about a connection, add an interface on the to-many side of the relationship definition. For example, if property information is to be defined for each of the end connections, interfaces are added for each end (such as IConnector End1 and IConnector End2) of the connector and related to the connection as shown in the following diagram:



End connections denote the end point of one end of a connector. A TapOrFitting is any other point along the connector that does not terminate the connector. For example, a valve in a P&ID pipe run that does not change any of the pipe run properties, and does not break the pipe run into two pipe runs, will have two fitting points tied to the one piping run connector.

For SmartPlant applications, instances of the connectivity pattern typically refer to connectors as connectors at the physical level, runs at the logical level, and systems at the more abstract level.

Occupancy Pattern

The occupancy relationship is a one-to-one relationship in which a particular object occupies the space created by another object.



With the occupancy relationship, the abstraction is always on the right side and this interface is always named the same as the interface on the left side with an occupant suffix. The relationship is named based on the interface on the left with the I prefix removed and an occupancy suffix added.



Interface Patterns

Patterns are used to express the *typical material* and *part occurrence* hierarchies.

Typical Material vs. Part Occurrence Interface Diagram Pattern

As the word *typical* implies, it refers to parts that are in a catalog that have typical properties. Below is a sample plant design workflow to develop the idea of *typical material*.

- An early-stage plant design simulator (like AspenTech Zyqad) establishes the need for moving a fluid from one area to another at a rate of 100 GPM. The ability to move fluid is a *required facility*.
- Further work is done by a separate tool (SmartPlant Instrumentation) that determines a safety factor of 25% is required for the plant. The safety factor is a *required material*.
- The P&ID designer plans for two pumps to provide redundancy, which is a *planned material*. One conceptual pump is now two pumps that must be purchased.
- Pick a pump from the manufacturers' catalog that is listed with a flowrate of 75 GPM (typical material). Use the catalog part number EY-75VCP as the part name.
- Create a purchase order for two pumps from a company that carries the part (a *planned* pump). The pair of pumps can pump 150 GPM total.
- When the pumps are delivered, they are a newer model pump, each of which is rated at 80 GPM. Each pump gets an inventory tag (an *actual* pump), and the pair fulfills all of the design requirements.

Different Ways to Think About a Pump (Polymorphism)

As you can see, some properties about a pump apply to:

- Early-stage pump design requirements (flowrate is required).
- Required pump (tag number).
- Catalog pump (manufacturer's catalog part number and weight).

- Actual pump (serial number).

In order to support these different views of different property sets for the pumps, the SmartPlant schema supports the concepts of part, tag, and asset.

- The manufacturer's catalog number does not have any meaning to the actual pump that is installed in the plant.
- In a similar manner, the serial number of the pump is not listed in the catalog, but just the characteristics.

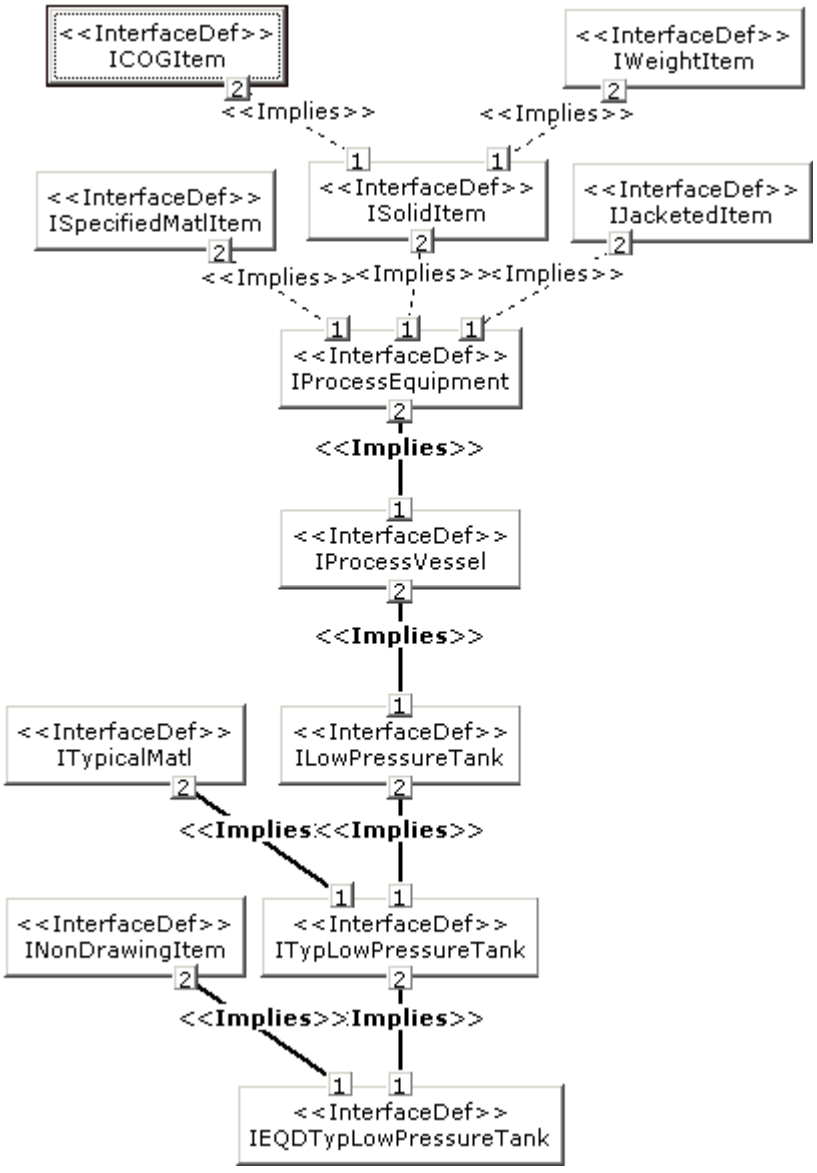
SmartPlant uses the concept of polymorphic interfaces to create an interface, IPump, that is polymorphic with respect to:

- Pump design requirements
- Pumps in catalogs
- Pumps that are ordered from a manufacturer
- Pumps that are installed
- Other kinds of pumps

NOTE For more information on polymorphic interfaces, see Chapter 3 of *A Gentle Introduction to the SmartPlant Schema*.

Typical Means Type

The word *typical* is the concept of *type* or *typeness* of a class. Properties that define the type of a pump are not the same properties as the many other kinds of pumps that exist.



Typical Material Interface Diagram Pattern

The illustration above displays an interface hierarchy for a typical low-pressure tank that can appear on an equipment datasheet.

- IEQDTypLowPressureTank is a non-drawing item because it appears on an equipment datasheet, which is non-graphical.
- It is also a typical low-pressure tank, such as ITypLowPressureTank. The modeler can infer that there can be other interfaces in the future, such as IXYZTypLowPressureTank, that would imply ITypLowPressureTank.

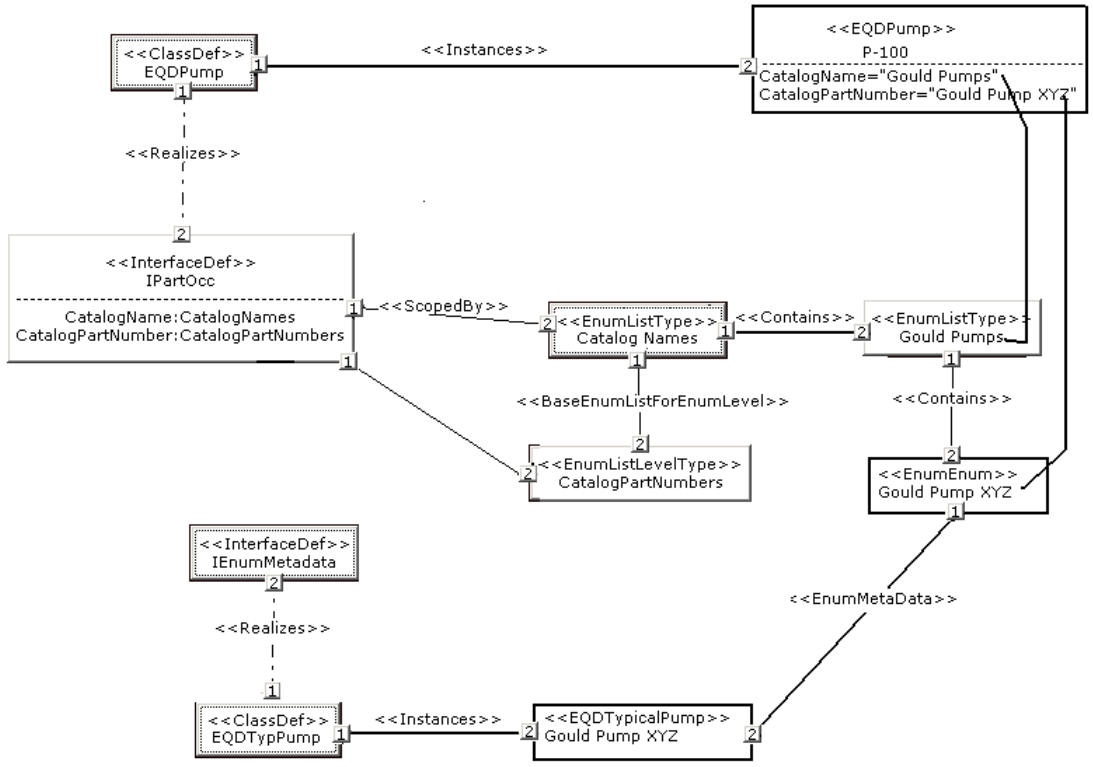
- A typical low-pressure tank, ITypLowPressureTank, is a typical item, such as ITypicalMatl, and a low-pressure tank, such as ILowPressureTank.
- ILowPressureTank is a process vessel (IProcessVessel), which is a type of process equipment (IProcessEquipment), which is a solid item (ISolidItem), which may have a weight (IWeight) and center-of-gravity (ICOGItem). An application that models the equipment may or may not actually publish the weight and center of gravity.

Notice that all the implies relationships up to IProcessEquipment are required, which means that every typical item exposes all the properties within the hierarchy. It also exposes the properties in IProcessEquipment that contain the type hierarchy, ProcessEqType1, 2, 3, 4, 5, 6.

The optional implies relationship between IProcessEquipment and IJacketedItem means that a catalog item, such as a part, may or may not have a jacket.

Part Occurrence Interface Diagram Pattern

The illustration below shows how a part occurrence, a pump, on an Equipment Datasheet carries catalog data through a unidirectional association that links a part to an occurrence.



After a brief introduction, a path can be traced around the outer edge of the diagram, clockwise, starting at *IPartOcc*:

- Interface IPartOcc exposes the CatalogName and CatalogPartNumber property definitions.
- CatalogName and CatalogPartNumber are scoped by a 2-level enumerated list of values, such as Catalog Name (Level 1) and Catalog Part Number (Level 2). In the example, the Gould Pumps catalog contains some Gould pumps, including Gould Pump XYZ.
- An instance of an EQDPump, such as P-100, is published with properties CatalogName="Gould Pumps" and CatalogPartNumber="Gould Pump XYZ".

- The entry Gould Pumps is a level-1 enumerated entry in the Catalog Name list.
- The entry Gould Pump XYZ is a level-2 enumerated entry in the Catalog Part Number list.
- In the lower portion of the example, there is an instance of a typical pump (schema ClassDef EQDTypicalPump), such as Gould Pump XYZ, which provides enumerated metadata for the enumerated entry Gould Pump XYZ (the EnumEnum).

Relationship Patterns

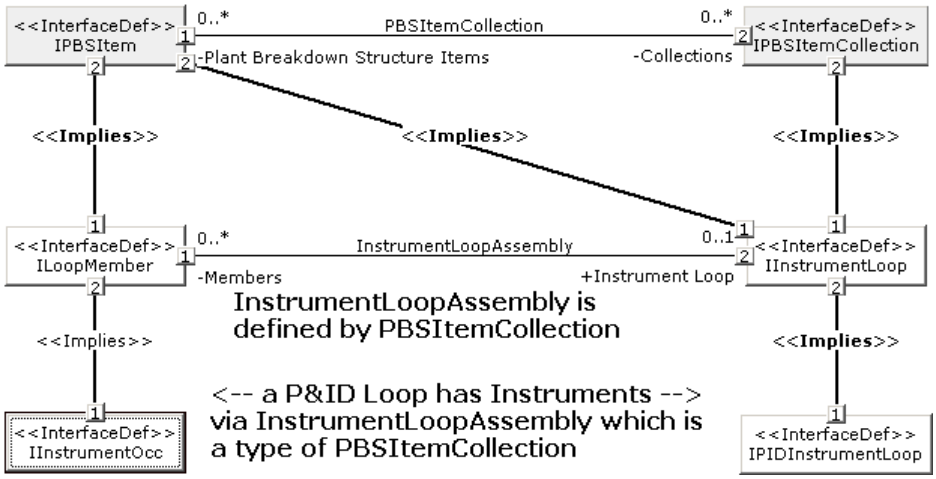
Relationships between interface definitions are defined as a RelDef class. Relationships can have an abstract parent relationship, which controls the behavior.

Plant Breakdown Structure (PBS) Patterns

The following illustration shows high-level, abstract interfaces of IPBSItemCollection, which is a group of IPBSItems. To process the members of the collection, you must traverse each of the PBSItemCollection associations to get each PBS Item one-by-one. But there are no classes called PBSItem or PBSItemCollection. The Interfaces are abstract and the relationship can be considered abstract, too. The relationship is the concept of PBS Items in a collection.

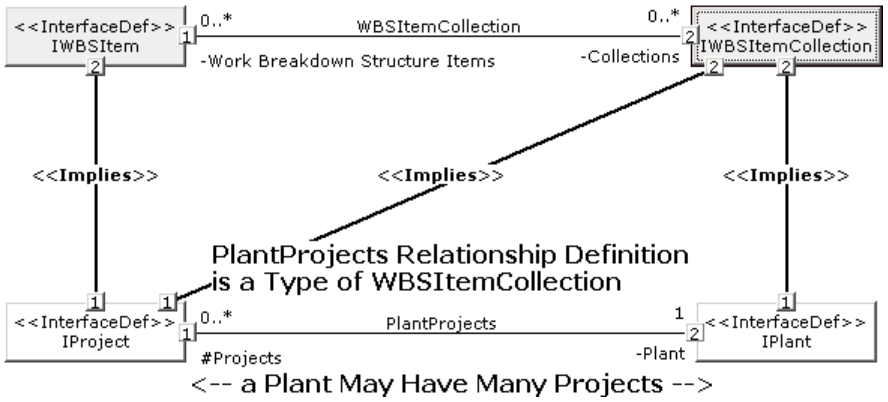
When a ClassDef realizes IPBSItem or IPBSItemCollection, it can be a child, parent, or both in a relationship with other PBS items. Relationships can be derived from the abstract parent relationship definition, and can then be used to define relationships that will exist in the published document.

The illustration below shows that the relationship definition InstrumentLoopAssembly is a type of PBSItemCollection. It is subject to all the benefits and all the semantic rules that PBSItemCollection defines. Other semantic rules can be added to InstrumentLoopAssembly if necessary. It is also valid to have intermediate-level, abstract RelDefs, each of which contributes some kind of semantic meaning to the final, concrete relationship definition.



Work Breakdown Structure (WBS) Patterns

Similar to the previous section, the following illustration shows that the relationship definition PlantProjects is a type of WBSItemCollection. It is subject to all the benefits and all the semantic rules that WBSItemCollection defines. Other semantic rules may be added to PlantProjects if necessary. It is also valid to have intermediate-level, abstract RelDefs, each of which contributes some kind of semantic meaning to the final, concrete, relationship definition.



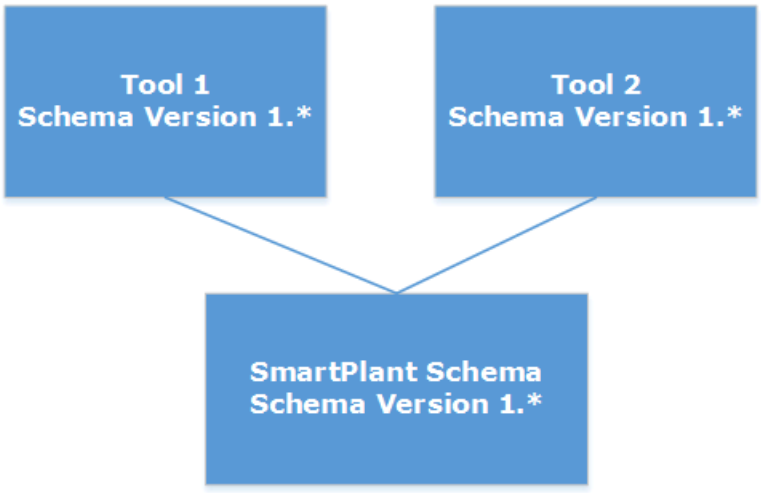
SmartPlant Schema Modification Rules

The SmartPlant schema must be allowed to evolve because the scope of what the schema defines will change over time. This document describes the rules for modifying the SmartPlant schema (and indirectly, all of the component schemas). Unless explicitly stated otherwise, all of the rules defined in this document apply to both the meta-schema and the SmartPlant schema.

The schema evolution rules support four software schema configurations. In the following four examples, Tool 1 and Tool 2 represent applications written to integrate with the SmartPlant schema. The schema version number has two parts (X.X), where the first number is the schema as it was released by Intergraph. The second part is optional and indicates support for customized changes made by the user.

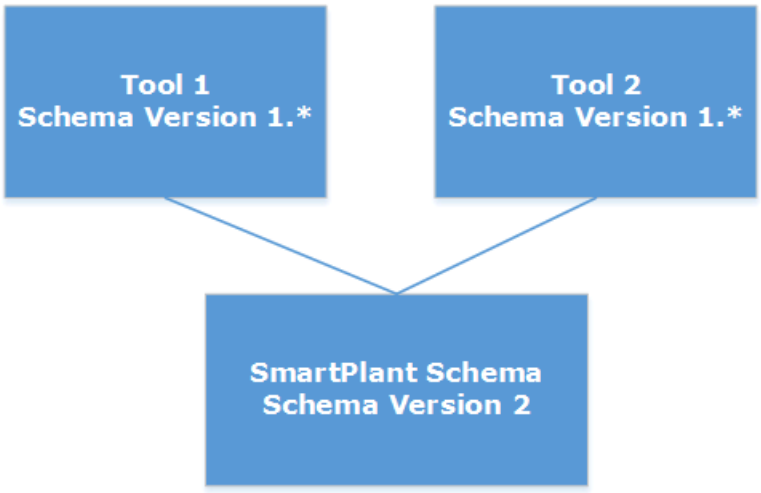
The schema version for a tool indicates the version of the SmartPlant schema against which the tool adapter was written. Including the asterisk (*) indicates that the metadata (for example, a map file) used by that tool has been updated to the latest set of user-specified customizations and delivered by the SmartPlant schema. Because all four configuration examples include the asterisk, you can conclude that the metadata used by the adapter has been updated to work against the latest SmartPlant schema.

Configuration 1 is the simplest case. All of the tools and the SmartPlant schema are written against the same version of the schema, and there should be no issues.



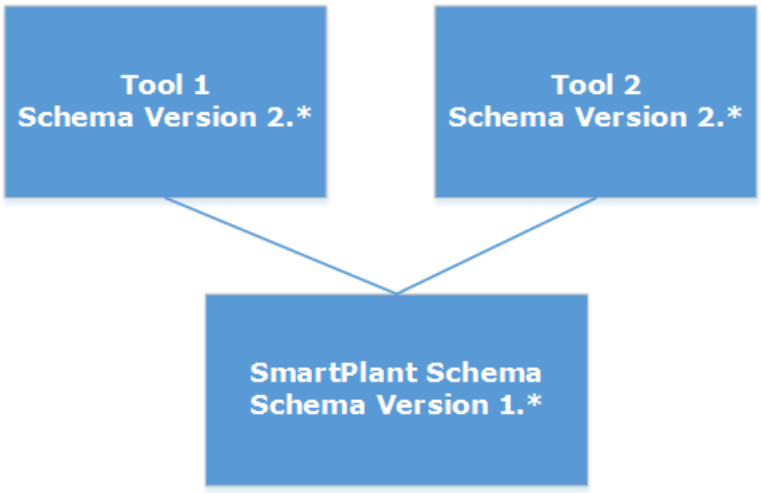
Configuration 1: All tools (adapters) and the SmartPlant schema on the same version

A new version of the SmartPlant schema has been loaded in configuration 2. However, each of the tools is running with its adapter written to work against the previous version. In this configuration, the new SmartPlant schema cannot include any changes that would prevent the old tool adapters from working with the new SmartPlant schema.



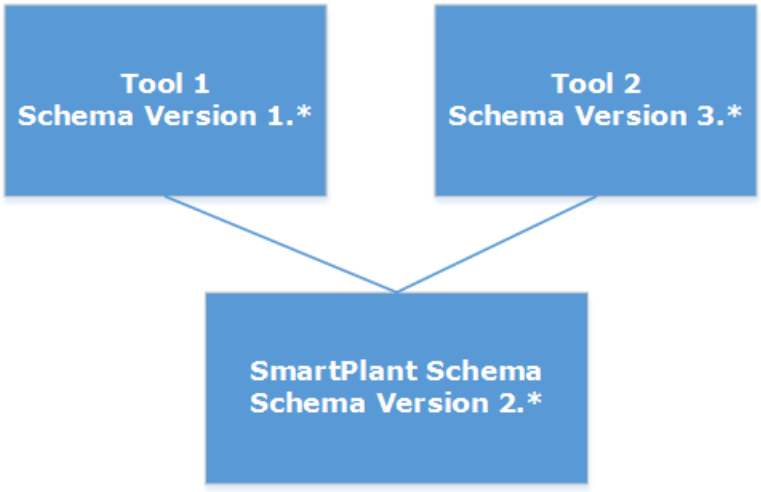
Configuration 2: Tools are on an older version of the schema than the SmartPlant schema

New versions of Tool 1 and Tool 2 have been loaded in configuration 3. These tools have adapters that were written against a version of the SmartPlant schema that is newer than what is currently loaded. The tool adapters must support backward compatibility, and they will have less functionality against the older, loaded version of the SmartPlant schema.



Configuration 3: Tools are on a newer version of the schema than the SmartPlant schema

Configuration 4 is the worst-case scenario. Some of the tool adapters are working against a schema version that is older than the loaded schema, and some of the tools are working against a newer schema version than the loaded schema.



Configuration 4: Tools are on both older and newer versions of the schema than the SmartPlant schema

Schema modification rules

Schema modification rules help you avoid breaking existing tool adapter software (as in configuration 2 and in part of configuration 4). Schema evolution rules also allow newer versions of tool adapters to work with older versions of the SmartPlant schema (as in configuration 3 and the other part of configuration 4).

When you decide to modify—or extend—the schema, consider the impact any modifications could have.

- If you make a conforming change (for example, you add a new, optional property definition), what will happen?
- If you make a non-conforming change (for example, you add a mandatory interface definition), what will happen?
- How will you resynchronize your extended schema with a new schema from Intergraph?
- Do schema files and data files have to be forward-compatible and backward-compatible in order to be useful?

Changes that might break existing publish or retrieval logic are not allowed, because those changes would break configurations 2 and 4. However, you can create explicit rules for the adapter to follow to avoid these problems.

Tool adapters must be written to work with old versions of the SmartPlant schema. Additions made to the SmartPlant schema after initial release must not be hard-coded into a tool adapter unless the coding will allow the adapter to continue working with older schema versions that do not have the additions (for example, using If statements). Likewise, any changes made to the map file after initial release must be delivered as a differential set of instructions rather than as a new map file. The version of the SmartPlant schema then drives the loading of these differential instructions.

Rule enforcement

Every schema object will be stamped with the official revision and version against which it was created. Objects that predate the latest released version must adhere to the rules defined in this document (and as such, are supported by the Schema Editor). The rules defined in this document will be supported by the Schema Editor and by the delivery and setup procedures.

Creation rules

Keep the following rules in mind when making changes that affect object or relationship creation.

- New objects of any type can be created; however, the relationships between new objects and existing objects cannot violate a revision rule.
- New relationships between existing objects can be created; however, the new relationship cannot violate a revision rule.

Deletion rule

An object (including the relationships of the object) that supports ISchemaObj cannot be deleted if its revision or version predates the current software revision or version.

Revision rule

If a schema object type is not included in this section, then any type of revision to that object is allowed.

All object types

Once the SmartPlant schema is delivered, all UIDs for all of the objects in that schema will adhere to UID rules.

- Each UID will be unique (within the context of the engineering data repository)
- The UID of an object will never change
- If an object is deleted, the UID for that object will never be reused

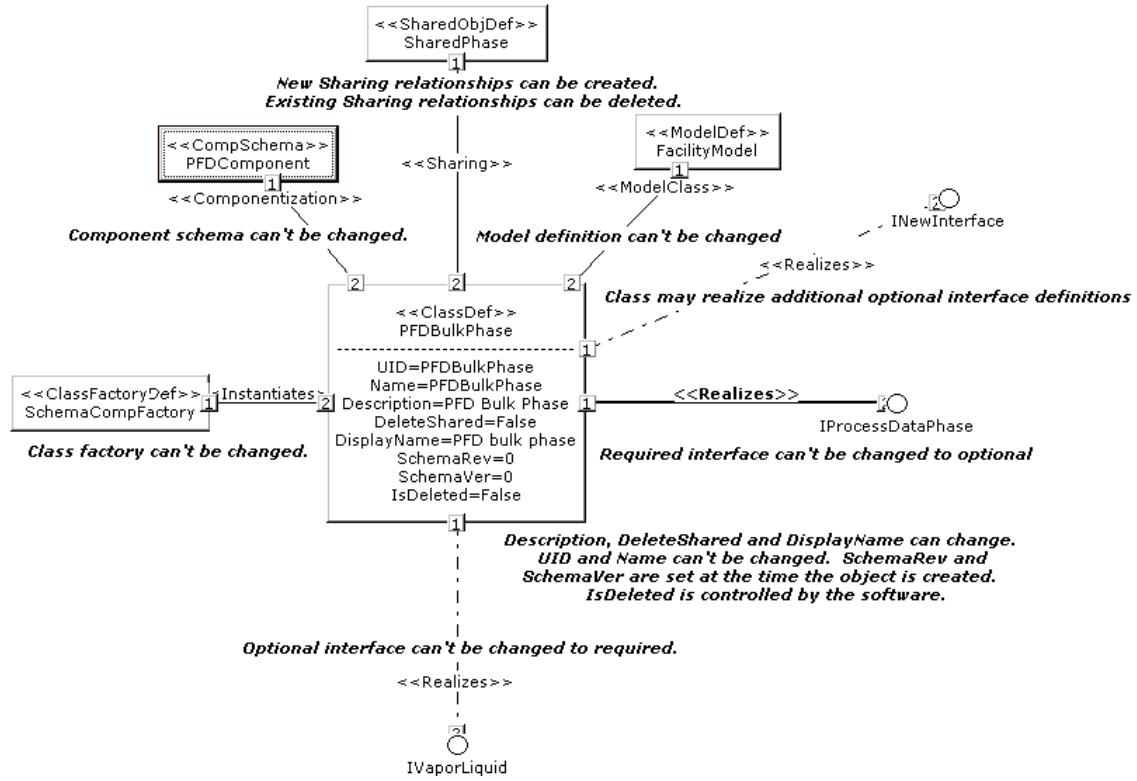
All schema objects must follow the rules defined by the meta-schema. Revisions that would break any of those rules are not allowed.

Class definitions

For existing class definitions, the following rules apply to both the schema and the meta-schema.

- The **UID** property cannot be changed
- The **Name** property cannot be changed
- The **Description**, **DeleteShared**, and **DisplayName** properties of a class definition can be changed
- A class definition can realize additional optional interface definitions
- An optional interface definition cannot be changed to a required definition
- A required interface definition cannot be changed to an optional definition
- The component schema for the class definition cannot be changed
- The model definition associated with the class definition cannot be changed
- The class factory definition associated with the class definition cannot be changed
- New sharing relationships can be created

- Existing sharing relationships cannot be deleted



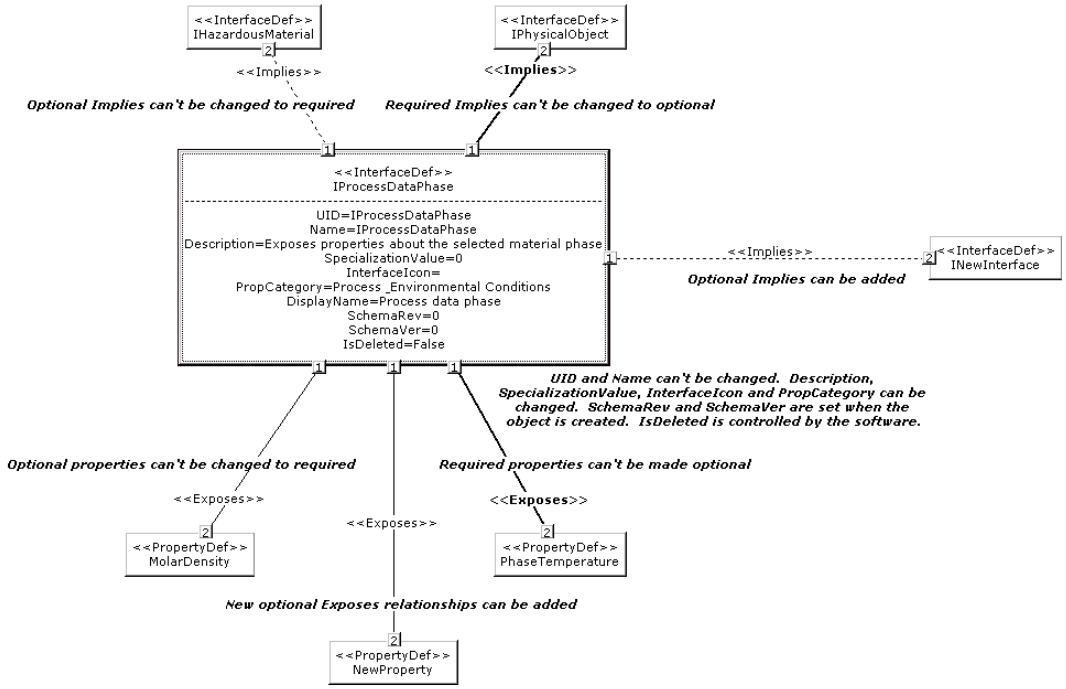
An example of a class definition and its relationships

Interface definitions

Although the COM rules state that that interfaces are immutable, the rules for existing interface definitions in the SmartPlant schema are not so rigid.

- The **UID** property cannot be changed
- The **Name** property cannot be changed
- The **Description**, **SpecializationValue**, **InterfaceIcon**, and **PropertyCategory** properties of an interface definition can be changed
- Optional property definitions can be added to an interface definition
- Existing property definitions exposed by the interface definition cannot be deleted
- Required property definitions cannot be changed to optional definitions
- Optional property definitions cannot be changed to required definitions
- Existing Implies relationships cannot be deleted
- New optional Implies relationships can be added
- Required Implies relationships cannot be changed to optional Implies relationships

- Optional Implies relationships cannot be changed to required Implies relationships

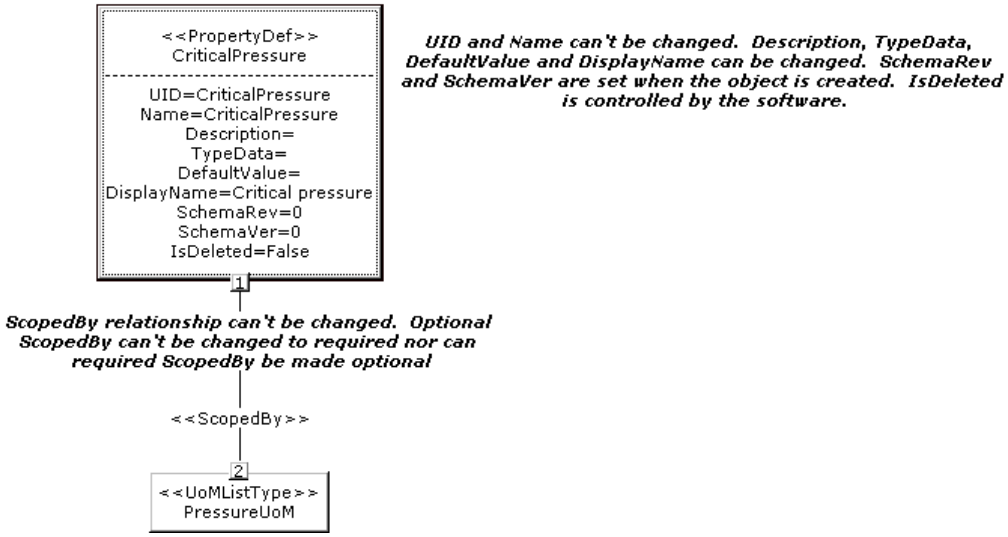


An example of an interface definition and its properties

Property definitions

Existing property definitions must adhere to these rules.

- The **UID** property cannot be changed
- The **Name** property cannot be changed
- The **Description** and **DisplayName** properties of a property definition can be changed
- Although not currently used, the **TypeData** and **Default** properties of a property definition can be changed
- The **ScopedBy type** property of the property definition cannot be changed
- A required **ScopedBy** relationship cannot be changed to an optional one
- An optional **ScopedBy** relationship cannot be changed to a required one



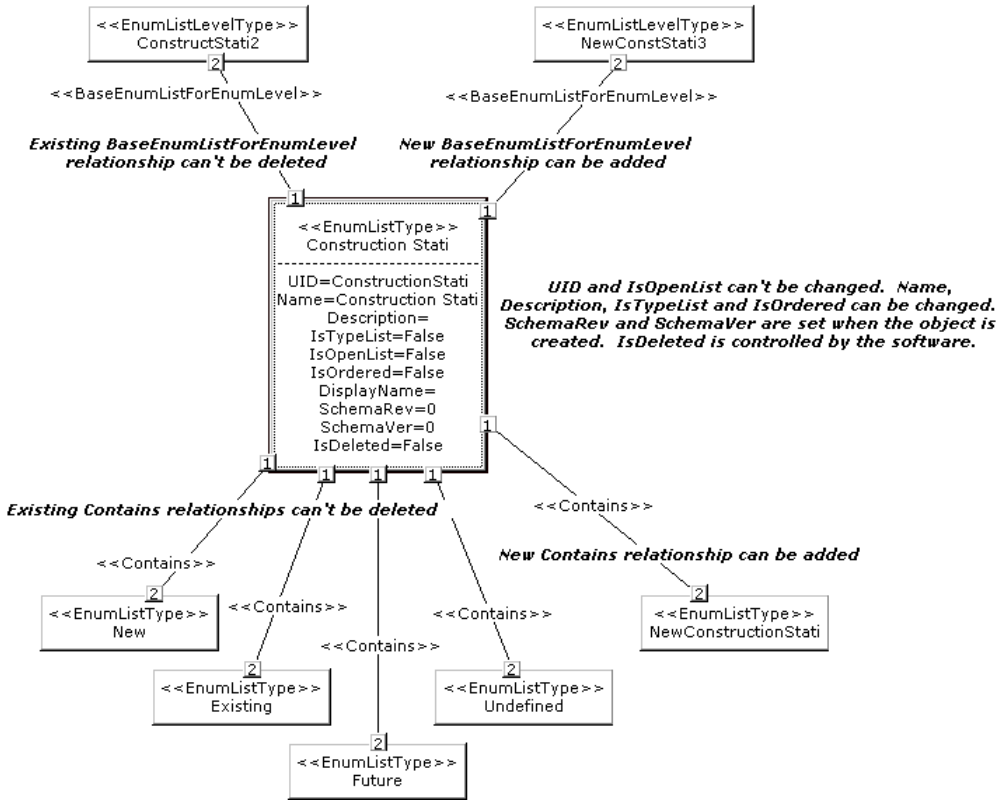
An example of a property definition and its relationships

Enumerated lists

Existing enumerated lists must adhere to these rules.

- The **UID** property cannot be changed
- The **IsOpenList** property cannot be changed
- The **Name** property can be changed because it is not tied to the **UID**
- The **Description** property can be changed
- The **IsTypeList** property (not currently used) can be changed
- The **IsOrdered** property can be changed; however, this rule is likely to be restricted at a later time
- The **EnumNumber** property can be given a value if it has not already been defined

- If the **EnumNumber** property was previously defined, the value cannot be changed
- Existing contains relationships can be added
- Existing BaseEnumForEnumListLevel relationships cannot be deleted
- New BaseEnumForEnumListLevel relationships can be created

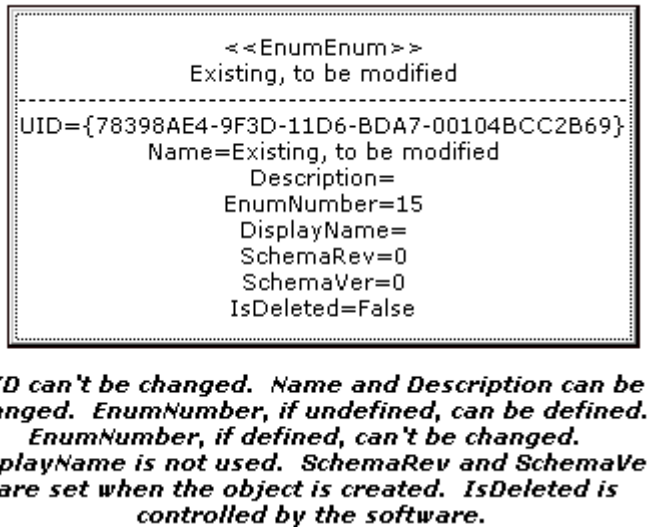


An example of an enumerated list and its properties

Enumerated entries

- The **UID** property for an existing enumerated entry cannot be changed
- The **Name** property (that is, the short description) of an existing enumerated entry can be changed
- The **Description** property of an existing enumerated entry can be changed
- An undefined **EnumNumber** property can be given a value

- If the **EnumNumber** property was previously defined, the value cannot be changed



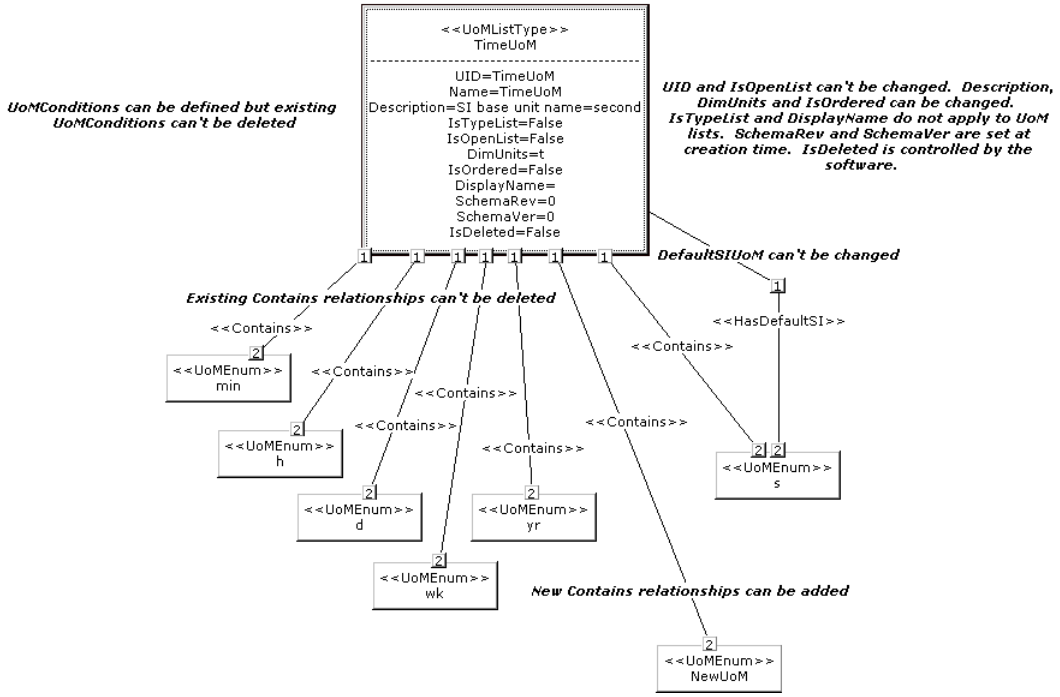
An example of an enumerated entry and its properties

Unit of measure lists

Existing unit of measure (UoM) lists must adhere to these rules.

- The **UID** property cannot be changed
- The **Name** property can be changed because it is not tied to the **UID**
- The **Description** property can be changed
- The **DimUnits** property can be changed
- An undefined **EnumNumber** property can be given a value
- If the **EnumNumber** property was previously defined, the value cannot be changed
- Existing contains relationships cannot be deleted
- New contains relationships can be added
- A UoMConditions relationship cannot be deleted

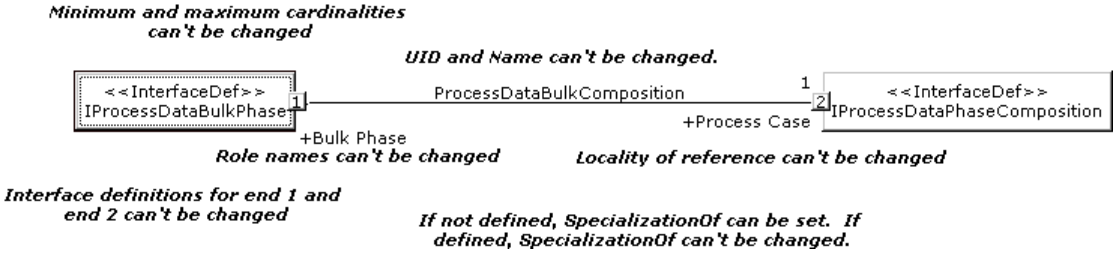
- The default SI unit of measure for the UoM list cannot be changed



Example of a unit of measure list and its properties

Unit of measure

- The **UID** property of an existing unit of measure cannot be changed
- The **Name** property (that is, the short description) can be changed
- The **Description** property can be changed
- The **A** (ACny) and **B** (BCny) conversion factors can be changed. However, you should do this only if there is an error in the value, or if greater precision is necessary

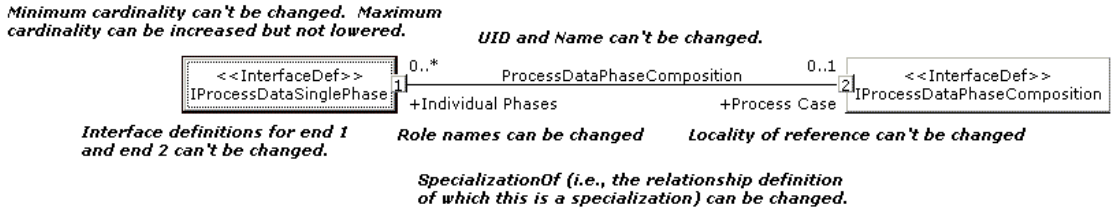


Example of unit of measure property and relationship

Relationship definitions

Existing relationship definitions must adhere to these revision rules.

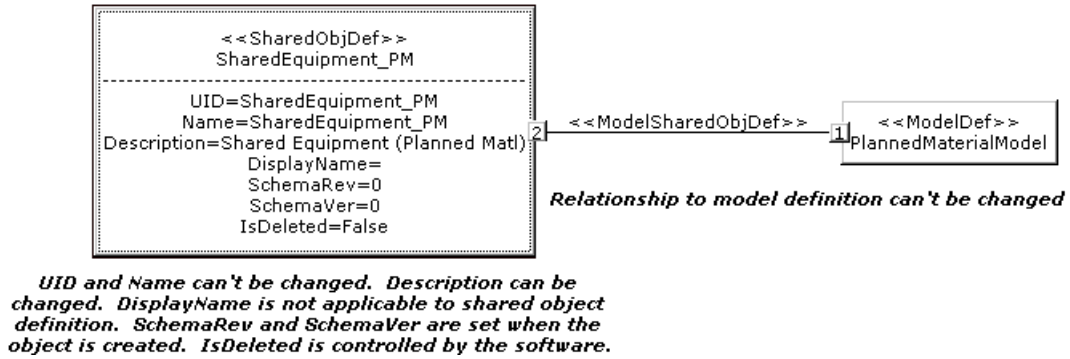
- The **UID** property cannot be changed
- The **Name** property cannot be changed
- Role names cannot be changed
- Localities of reference cannot be changed
- Minimum cardinalities cannot be changed
- Maximum cardinalities cannot be changed
- The interface definition at each end of a relationship must not change
- An undefined **SpecializationOf** property can be defined
- A defined **SpecializationOf** property cannot be changed



An example of a relationship definition

Shared object definitions

- The **Name** property of the shared object definition cannot be changed
- The **UID** property of the shared object definition cannot be changed
- The **Description** property of the shared object definition can be changed
- The **Model Definition** associated with the shared object definition cannot be changed



An example of a shared object definition and its relationship

Create New Schema Objects

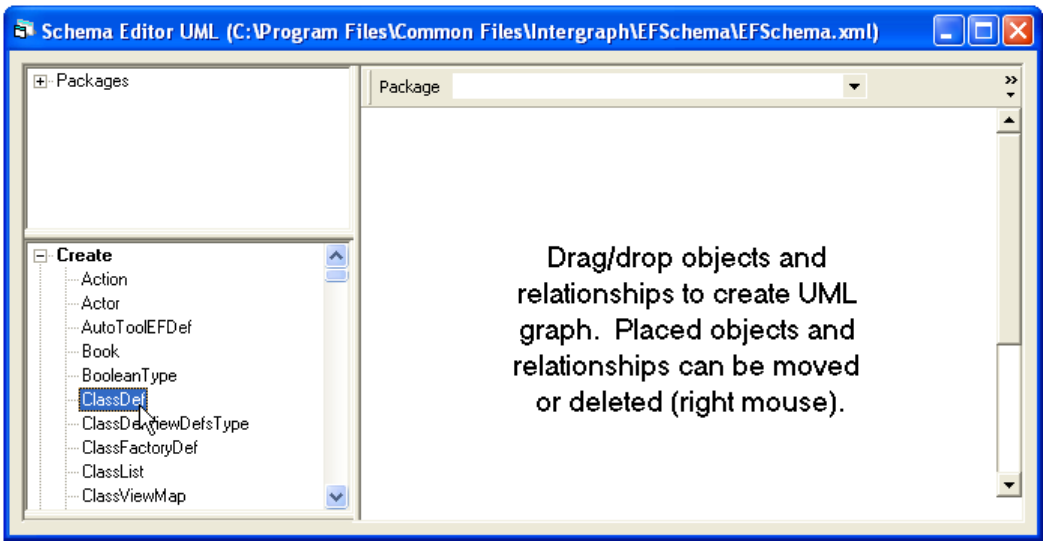
In the Schema Editor, you can modify the schema by creating new objects, interfaces, relationships, enumerated lists, and so on.

To create objects, interfaces, and relationships, you can use the two editor views: **Editor** and **Editor2**, as well as other Schema Editor views. These two editor views provide the same functionality, arranged in a slightly different user interface.

To open either of the editor views, on the **View** menu, click the type of file that you want to modify. Then click **Editor** or **Editor2** in the **Form** box on the simple **View** dialog box or on the **View** tab of the advanced **View** dialog box. For example, to use one of the editor views to modify the active schema, click **View > Schema (All)**.

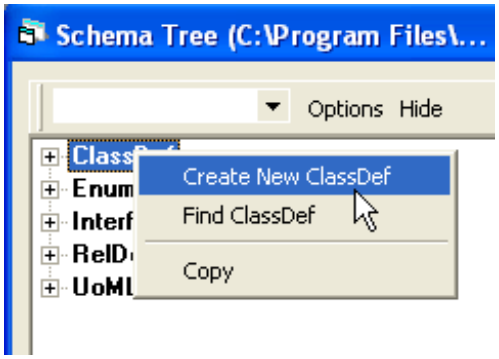
Creating new objects

To create a new object in either of the editor views, you can double-click the object that you want to create in the **Create** tree.



NOTES

- You can also create objects and relationships by right-clicking objects in various Schema Editor views. For example, in the Schema **Tree** view, you can right-click a class and then click **Create New ClassDef** on the shortcut menu to display the **Create** dialog box for class definitions.



- You can also drag and drop objects from the **Create** tree to the **UML** view to create them.

Creating new relationships between objects

In the Schema Editor, you can create relationships in a variety of ways. The following list describes the most common ways that you can use to create relationships:

- Using the editor views

  In the **Editor** or **Editor2** views, you can double-click the **Rel** object in the **Create** tree. Then, you can define the relationship type and objects at end one and end two of the relationship using the **New Relationship** dialog box.

- Editing the object at one end of the relationship

  You can create relationships by editing the object at one end of the relationship using the **Edit** dialog box for that object. In the **Edit** dialog box, you can add, remove, or modify existing relationships by changing the values in the text boxes.

- Modifying individual relationship edges

  For relationships that display individual edges, for example, in the drag-and-drop UML views, you can create relationships by clicking the relationship edge and then clicking the appropriate command on the **Actions** menu. You can also right-click the object in the drag-and-drop UML view to select an edit command.

Create a Class Definition

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **ClassDef** in the **Create** tree.
3. In the **New Class Definition** dialog box, type a name, description, and display name for the new class.
4. Beside the **Model definition** box, click ....

5. In the **Possible ModelForClass Values** dialog box, select the model to which the new class definition belongs. For more information, see *Model Definitions* (on page 20).
6. Beside the **Component schema** box, click ..., and select the component schema to which the new class definition belongs.

   TIP If the component schema that you want to associate the new class with does not already exist, click **Create** to define the object in the schema. For more information about component schemas, see *Create a Component Schema* (on page 170).
7. If the new class definition represents another view of an object represented in the database by another class definition, click ... beside the **Shared object definitions** box, and select the shared object definition for this class. You can select one or multiple shared objects. For more information, see *Shared Object Definitions* (on page 21).

   NOTE Shared objects must exist in the same model. For example, if you select **ActualMaterialModel** for the new class, all shared objects must also exist in the **ActualMaterialModel**.
8. Beside the **Class factory definition** box, click ..., and select the class factory for the new class.
9. Beside the **Primary interface definition** box, click ..., and select the primary interface definition for this class definition.

   TIP If the primary interface definition for the class definition does not exist, create it by clicking **New** in the **Possible PrimaryInterfaceDef for <ClassDef>** dialog box. For more information, see *Create an Interface Definition* (on page 171).
10. Beside the **Realized interface** definition box, click ..., and then select additional interfaces that this class realizes. You can select one or multiple interfaces. Click **OK** when you are done selecting interfaces.

    NOTES

    - Each class definition must realize at least one interface definition.
    - The software automatically creates a realizes relationship with the IObject interface, because all class definitions must realize that interface.
11. In the **New Class Definition** dialog box, click **OK**.

See Also

Create New Schema Objects (on page 168)
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
Class Definitions (on page 19)

Create a Component Schema

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **CompSchema** in the **Create** tree.

3. In the **New Component Schema** dialog box, type a name and description.

   TIPS

   - You can click ... beside the **Class definitions defined by component schema** list and then click **New** on the **Possible ComponentClassDefs** dialog box to create new class definitions to associate with component schema. For more information, see *Create a Class Definition* (on page 169).
   - However, since class definitions must have a relationship to a component schema, there will be no class definitions available for selection in the **Possible ComponentClassDefs** dialog box unless you create new ones.

Create an Interface Definition

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **InterfaceDef** in the **Create** tree.
3. In the **New Interface Definition** dialog box, type a name, display name, and description.
4. If you want to specify a category for properties that are exposed on this interface, click ... beside the **Property category** box and choose the category you want to assign to those properties. For more information, see the **Properties Categories** section of the *Interface Definitions* (on page 22) topic.
5. If you want to specify class definitions that will realize this new interface, click ... beside the **Realized by class definitions** box.
6. In the **Selectable** list in the **Possible RealizedByClassDefs** dialog box, check the boxes for the classes that will realize the new interface, and then click **OK**. You can select one or multiple classes.
7. If the new interface will be the primary interface for any class definitions, click ... beside the **Primary interface for class def** box. In the **Selectable** list, check the boxes for any class definitions for which this interface will the primary interface. Click **OK**.
8. Beside the **Implies interface definitions** box, click ..., and then select the interfaces that will be implied by the new interface. You can select one or multiple interfaces.

   NOTES

   - All interfaces must imply at least one other interface.
   - An interface will inherit all properties exposed by interface that it implies.
9. If necessary, click ... beside the **Implied by interface definitions** box, and then select the interfaces that will imply the new interface.

   NOTE The new interface should be implied by the primary interfaces of all class definitions that realize it.
10. If you want to create properties to be exposed by the new interface definition now, click ... beside the **Exposed property definitions** box, and click **New** on the **Possible ExposedPropertyDefs** dialog box. For more information, see *Create a Property Definition* (on page 172).

    NOTES

    - An interface may expose any number of properties, but it does not have to expose any.

- Since properties are exposed by only one interface definition, there are no selectable properties on the **Possible ExposedPropertyDefs** dialog box until you create new ones.

11. Beside the **Implementation of enumeration** box, click ..., and then select the enumerated list used by this interface, if one is used.
12. In the **New Interface Definition** dialog box, click **OK**.

See Also

Interface Definitions (on page 22)
Create New Schema Objects (on page 168)
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

Create a Property Definition

1. In the **Editor** view**,** double-click **PropertyDef** in the **Create** tree**.**
2. In the **New Property Definition** dialog box, type a name, description, and display name for the new property definition.
3. Beside the **Exposed by interface definitions** box, click ....
4. From the **Selectable** list in the **Possible ExposedByInterfaceDefs** dialog box, select the interface definition that exposes this property definition. Property definitions are exposed by only one interface definition.
5. Beside the **Scoped by property type** box, click ..., and select the property type that scopes this particular property.

   TIPS

   - You can select from standard property types like string, double, integer, Boolean, and so on, or you can select an enumerated list or unit of measure list to scope this property.
   - If the property type that scopes this property definition does not exist, you can create it by clicking **New** in the **Possible ScopedByPropertyType Values** dialog box. For more information, see *Create an Enumerated List* (on page 173).
6. Click **OK**.

See Also

Property Definitions (on page 26)
Create New Schema Objects (on page 168)
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

Create an Enumerated List

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **EnumListType** in the **Create** tree.
3. In the **New Enumerated List** dialog box, type a short description and long description for the new enumerated list on the **Overview** tab.

   NOTES

   - You can change the unique identifier (UID) supplied by the software. However, this unique identifier must be unique within the SmartPlant schema.
   - When you change the short description of the enumerated list, the software replaces the default UID with the short description in the enumerated entries tree. However, while the UID is replaced in the view, it is not replaced in the schema unless you update the value in the **UID** box.
4. If this enumerated list is a sublist nested within another enumerated list, click ... beside the **Contained in** box and select the enumerated list to which the new list belongs.

   NOTES

   - When you select the enumerated list that this list belongs to, the enumerated entries tree view updates with the entries from the selected enumerated list.
   - If the list you select is part of another list, that list appears in the enumerated entries tree also.
5. If the list contains other enumerated lists that already exist in the schema, click ... beside the **Contains** box and select the existing enumerated lists that this list contains. Click **OK**.

   NOTE When you click **OK**, the software updates the enumerated entries tree with the selected enumerated lists.
6. To add a new child enumerated list beneath this list, click **New Child List** and define properties for the list.
7. To add a new child enumerated entry to this list, do the following:
   - Select the list that you want to add an entry to in the tree.
   - Click **New Child Entry**.
   - Define properties for the new entry: UID, name or short description, long description, and number.
8. For each additional new entry, select an existing entry at the same level in the tree, and click **New Sibling**. Provide the details for each new entry.
9. Click **OK**.

NOTES

- You can also import enumerated lists and list entries from the Clipboard by clicking the **Import Lists from Clipboard** and **Import Entries from Clipboard** buttons. For more information, see *Import Enumerated List Entries from a Spreadsheet* (on page 174).
- To delete an entry or child list, select the entry or list in the tree view, and then click **Delete**.

See Also

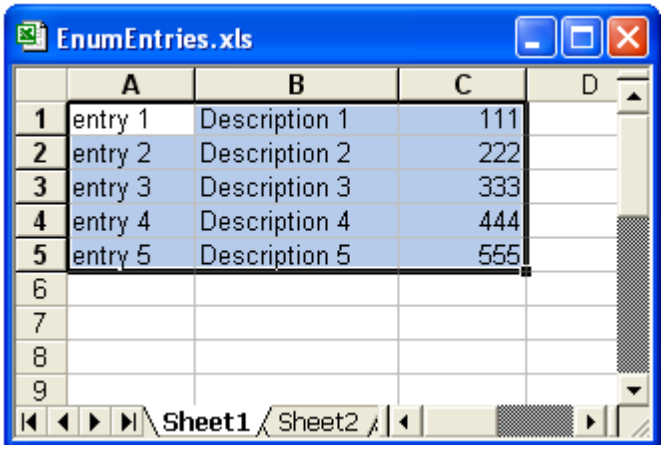
Create New Schema Objects (on page 168)
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)
Enumerated Lists (on page 27)

Import Enumerated List Entries from a Spreadsheet

1. In Microsoft Excel, create a spreadsheet containing your enumerated list entries.

   TIP In an authoring tool map schema, you can also import enumerated list entries into a tool map enumerated list and units of measure into a tool map unit of measure list using this procedure.

2. In Excel, select the rows containing your enumerated list entries, and click **Edit > Copy**.



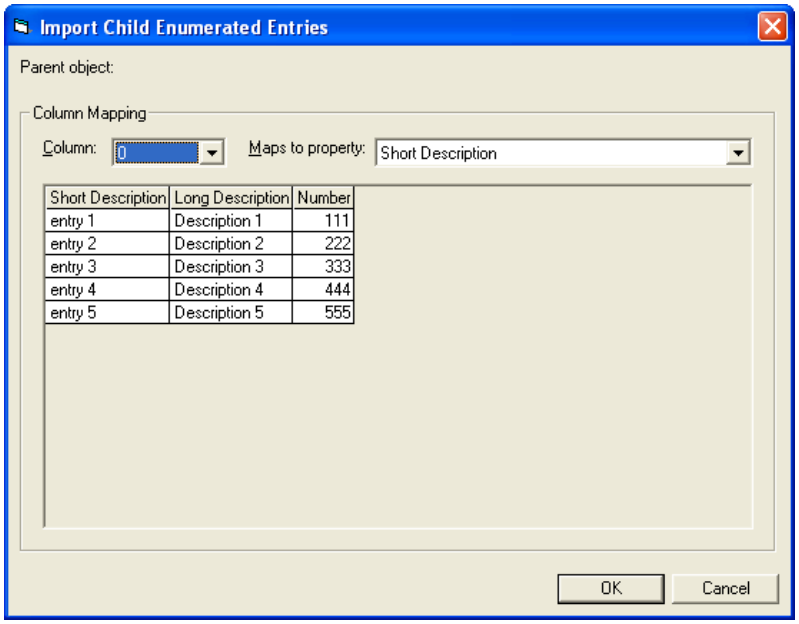
3. In the Schema Editor, create the enumerated list type to which you want to add the list entries. For more information, see *Create an Enumerated List* (on page 173) or *Create a Map Enumerated List Definition* (on page 437).

4. In the **New Enumerated List** dialog box, click **Import Entries from Clipboard**.

   TIP To import entries into a tool map enumerated list or map UoM list, click **Import Clipboard Entries** instead.

5. Verify that the default mapping between the columns in your copied spreadsheet and properties of enumerated list entries is correct. If this mapping is correct, you can skip to step 7 to import the entries.

6. If the default mapping is not correct, click the number for the column that you want to change the mapping for in the **Column** list. Numbering for the columns starts with zero, which is the first column in your spreadsheet.



7. In the **Maps to property** list, select the property that corresponds to the selected column in your copied spreadsheet.
8. When the mapping is correct, click **OK** to import the enumerated list entries.
9. Click **OK** in the **New Enumerated List** dialog box to save the new entries.

Create a Unit of Measure List

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **UoMListType** in the **Create** tree.
3. In the **New Unit of Measure List** dialog box, type a name and description for the new UoM list.
4. Beside the **Conditions list** box, click ..., and select the enumerated list that contains the possible conditions for a property scoped by this unit of measure list, if one exists. For example, the pressure unit of measure list uses a conditions list that identifies whether the pressure value is absolute or gauge.
5. To add a UoM to the list, click **New**.
6. In the **Short description** box, type the name of the UoM.
7. In the **A conversion factor** and **B conversion factor** boxes, provide the values necessary to convert a value in this unit of measure to the SI unit of measure for this unit of measure list. The conversion uses the following syntax: y = Ax + B, where y is the value in this unit of measure and x is the value in SI units.

TIP For example, the SI for unit of length is meters. For converting centimeters to meters, the A value is 100, and the B value is 0.

8. If this UoM is the default SI UoM for the list, select **Yes** in the **Is default SI unit of measure?** list.

   NOTES

   - Each unit of measure list must have one unit of measure designated as the SI unit.
   - The unit of measure designated as the SI unit must haven a value of 1 for the **A conversion factor** and a value of 0 for the **B conversion factor**.

9. To add addition UoMs to the list, repeat steps 5-9.

TIP To test the conversion for UoMs in the list, click **Test**. To be able to test conversions for the list, you must specify that one of the UoMs is the default SI UoM for the list.

See Also

Create New Schema Objects (on page 168)
Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

Create a Relationship Definition

1. From a tree view, right-click on an existing relationship definition and click **Create new RelDef**.
2. In the **New Relationship Definition** dialog box, provide a name and description for the new relationship. For more information about relationship definitions, see *Relationship Definitions* (on page 31).
3. In the **End 1** section of the dialog box, choose the interface definition for End 1 of the relationship.
4. In the **Role** and **Role display name** boxes, provide labels to represent the object or objects that will be on this end of the relationship.
5. Use the **Minimum cardinality** box to indicate whether this relationship is a requirement for objects on this end.

   TIP For example, documents cannot exist without a relationship to an owning group, so for the relationship definition between documents and owning groups, the minimum cardinality on the owning group end is set to 1. If the relationship is not required, you can leave this value 0.

6. Use the **Maximum cardinality** box to indicate how many of the objects at this end can be related to each of the objects at the other end.

   TIP For example, any number of files can be related to each document, so for the relationship definition between documents and files, the maximum cardinality on the file end is set to *, indicating any number.

7. If necessary, choose an option from the **Object at end 1** list to indicate restrictions on creating relationships on published objects. You can restrict the creation of the relationship to objects that are published by the same authoring tool or within the same container, or you can allow the relationship to be created when the object is created by a different authoring tool.

8. If you want to limit access to this relationship in SmartPlant Foundation Desktop Client or when viewing data in Schema Editor using the standard security options, set **Filter out edge in display** to **True**. Otherwise, the command to see this relationship on the end 1 object will be available to all users.
9. In the **End 2** section of the dialog box, use the boxes to provide the same information about the other end of the relationship definition as you provided for end 1.
10. In the **Delete behavior** section of the dialog box, indicate what should happen to the objects on each end of an existing relationship when the object on the other end is deleted.

    TIP For example, if you are creating a relationship between a contract and a document, specify what you want to happen to documents associated with the contract when the contract is deleted or what you want to happen to the contract when the document is deleted. In this section, select **True** if you want the deletion of one object to result in the deletion of the object at the other end. Select **False** if the object on the other end should be kept.
11. In the **Copy behavior** section of the dialog box, indicate what should happen to the objects on each end of an existing relationship when the object on the other end is copied.

    NOTE For each end, you can specify if copying that object will result in copying the other object, copying the other object and the relationship (leaving the relationship pointing to the same objects as the first relationship), or not copying the object or relationship at all.
12. Click **OK**.

Create an Edge Definition

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **EdgeDef** in the **Create** tree.
3. In the **New Edge Definition** dialog box, type a name for the new edge definition in the **Edge name** box on the **Overview** tab.

    NOTES

    - The property category helps organize properties in the **Properties** window in SmartPlant Foundation.
    - If no property definition is defined for the edge definition, properties across the edge definition are assigned a property category based on the property category defined for the interface definition that exposes the properties.
4. Provide a display name for the edge to define how it will appear in SmartPlant Foundation Desktop Client menus.

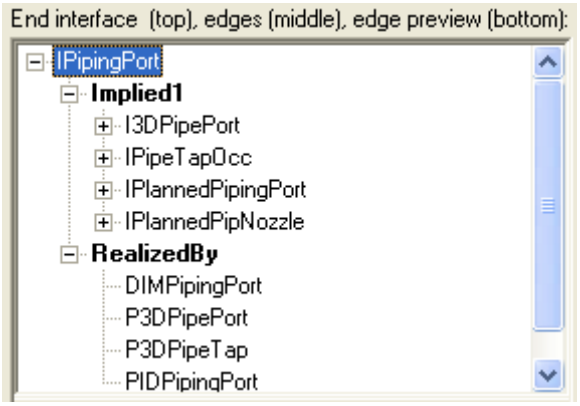
    TIP When this edge definition appears on a shortcut menu in the Desktop Client, the label will be "Show [Display name value]". So, to have the label appear as "Show Nozzles" on the menu, type "Nozzles" in the **Display name** box.
5. Click ... beside the **Starting interface** box, and select the starting interface definition for the new edge definition. Every edge definition must have a starting interface and traverse across at least one relationship definition.

6. If you plan to use SmartPlant Foundation security to restrict which users will see this edge definition in the Desktop Client, select the **Filter out edge in display** check box. If this check box is clear, the edge definition will be available to all Desktop Client users.

   NOTE For example, if you create an edge definition to traverse from a contract to all related documents and you leave the **Filter out edge in display** check box clear, all users will see a command on the shortcut menu of a contract to show the related documents, regardless of the roles of the user. However, if this check box is selected, SmartPlant Foundation will evaluate the role of the user to determine whether they should see that command.

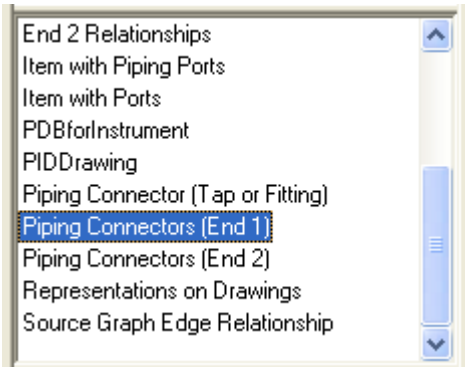
7. Click the **Path** tab.

8. In the **End interface** tree, select the ending interface definition or class definition that will be the starting point for the path of the edge definition.



   TIP The **End interface** tree displays the interface definition at the end of the current path and defines what appears in the **Edges** list. You can expand **Implied** nodes under each interface definition to select from interface definitions implied by the interface definition at the end of the current path. You can expand the **RealizedBy** node to select from class definitions that realize the interface definition.

9. In the **Edges** list, select the role at the other end of an edge (relationship definition or edge definition) that you want to include in the new edge definition. For more information about roles and relationship definitions, see *Relationship Definitions* (on page 31).

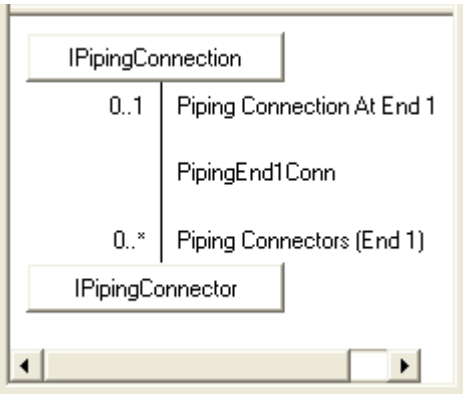


   TIPS

   - By default, the edges (relationship definitions or edge definitions) that appear in the **Edges** list are those that start at the end interface definition or at an interface definition

implied by that end interface definition. You can change the list of available relationships by selecting a different interface definition or class definition in the **End interface** tree.

- When viewing edges in the **Edges** list, you will not see the names of relationship definitions used to traverse the interfaces. Instead, you will see the name of the role assigned to the other end of the relationship from your starting point, as selected in the **End interface** tree.
- When you select an edge, the software displays the UML for that edge definition in the **Edge preview** pane under the **Edges** tree.



10. Click **Add to Path** to add the path for the selected edge to the path for the edge definition you are defining.

    TIPS

    - When you add the edge to the path, the UML in the **Current edge definition** pane is updated, and the **End interface** tree and **Edges** list are also updated for the new ending interface definition.
    - To remove edges from the path definition, click **Remove Last Edge from Path** on the **Path** tab. Clicking **Remove Last Edge from Path** only removes one edge at a time starting with the last one added.

Add a Property Filter to an Edge Definition

If you would like to filter the results that will be returned by an edge definition when it is used in SmartPlant Foundation Desktop Client, use the following steps to further configure the edge definition. For example, you can create an edge definition to find documents associated with a specific contract and then add a filter so that the edge definition returns only documents with a status of CURRENT.

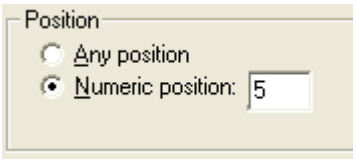
1. Go to the **Filter** tab.
2. Click **Add Filter**.
3. If necessary, change the value in the **Ending interface**, **Relationship path**, or **Interface at end of path** list to the interface that exposes or inherits the property you want to filter on.
4. Choose the property that you want to filter on in the **Property** list.
5. From the **Operator** list, choose the operator to use when evaluating the value for the selected property. The options include equal to (=), less than (<), greater than (>), or not equal to (<>).

6. Type a value in the **Comparison value** box for the selected property. Or, if the property is scoped by an enumerated list or UoM list type, click ... to open the **Possible Values** dialog box. On the **Possible Values** dialog box, select one or more acceptable values from the list, and click **OK**.

   TIP If you want the criteria to include multiple values for a specific property, you can enter multiple values separating the values with a colon.

7. If you want to define criteria for multiple properties, click **Add Filter** again and provide the criteria for the next property.

8. If the path specified for the edge definition includes traversal across an ordered relationship definition, click **Numeric position**, and type the position number that identifies a specific object from the set of objects at the end of the path.



   TIPS

   - For example, if you want to return the fifth nozzle on a piece of equipment, the numeric position should be set to 5.
   - If you specify a position, a maximum of one object will be found when the edge definition is traversed.

Add an Interface Filter to an Edge Definition

If you would like to filter the results that will be returned by an edge definition to only include objects that have instantiated a specific interface, use the steps below to further configure your edge definition. For example, if you want to create an edge definition to display the P&ID an object is on, you could select IPIDDrawing in this list to make sure that only P&IDs, and not other objects that participate in the DwgRepresentationComposition relationship, are returned by the edge definition.

- On the **FIlter** tab, click ... beside the **End objects must support one of the following interfaces** box, and select the interfaces.

See Also

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

Create a Graph Definition

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **DirectedGraphDef** in the **Create** tree.
3. In the **New Graph Definition** dialog box, type a name for the new graph definition in the **Graph name** box on the **Overview** tab.

   NOTE A comparison graph definition is a special type of graph definition that is intended for performing comparisons between the two sets of data described by the comparison graph

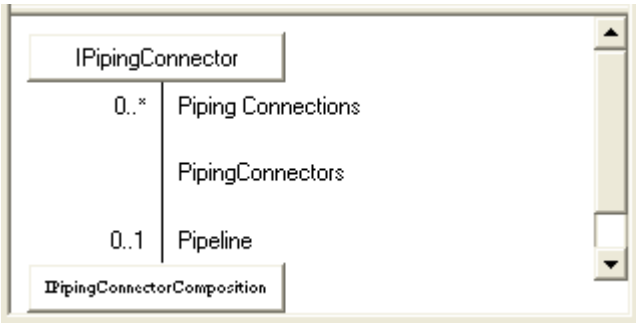
definition. Graph definitions are only used for comparison when you select the option to compare using view definitions during a compare.

4. If you want the display name for the graph definition to be different from the name, type the display name in the **Display name** box.
5. Click ... beside the **Starting interface** box, and select the starting interface definition for the new graph definition.

   NOTE The starting interface determines which edge definitions are available to include in the graph definition.
6. Click the **Layout** tab.
7. In the **Interfaces and classes** tree, select an interface definition or class definition to display edges for that interface definition or class definition in the **Edges** list. The selection here determines which edge definitions and roles appear in the **Edges** list.
8. In the **Edges** list, select an edge definition or a role at the other end of a relationship definition that you want to add to your graph definition.

   NOTES

   - The **Edges** list displays the edge definitions that apply to the interface definition or class definition selected in the **Interfaces and classes** tree. If you select an interface definition in the **Interfaces and classes** tree, these edge definitions include those that start at the selected interface definition or at an interface definition directly or indirectly implied by the selected interface definition. If you select a class definition in the **Interfaces and classes** tree, the edge definitions include those that start at an interface definition realized by the selected class definition.
   - When viewing edges in the **Edges** list, you will not see the names of relationship definitions used to traverse the interfaces. Instead, you will see the name of the role assigned to the other end of the relationship from your starting point, as selected in the **End interface** tree.
   - When you select an edge definition, the software displays the UML for that edge definition in the **Edge preview** pane under the **Edges** tree.

   - If you select an interface definition in the **Edges** list, you can click **Create Edge Definition** to create an edge definition starting from the selected interface definition. For more information about creating edge definitions, see *Create an Edge Definition* (on page 177).
9. Click **Add to Graph** to add the selected edge definition to the graph definition you are defining.

CAUTION Changing the name of a graph edge in the grid beneath the UML display invalidates any view properties that use this graph edge for view definitions that are tied to this graph definition.

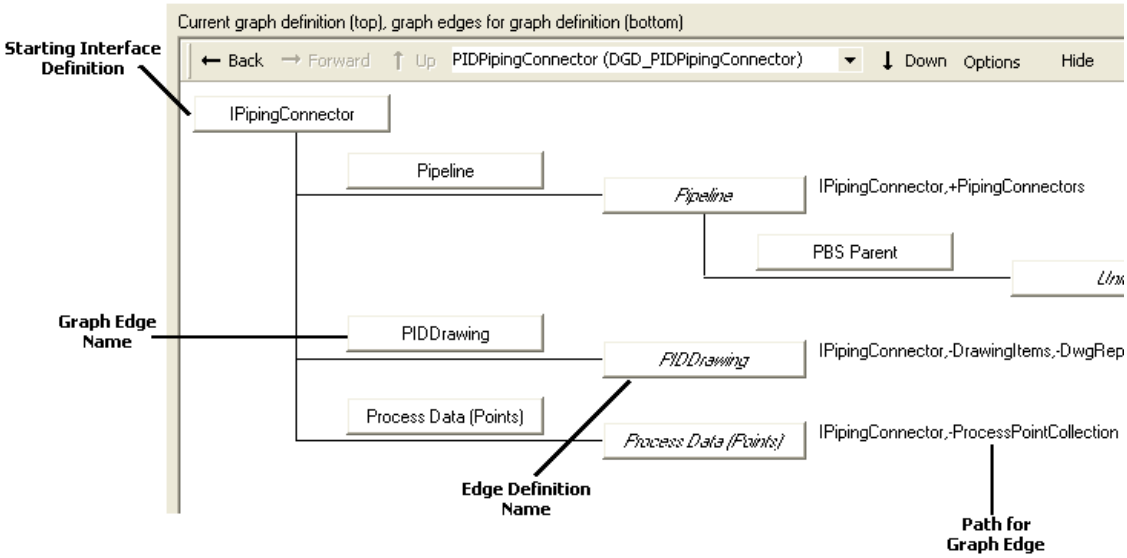
10. Add any other additional edge definitions or relationship definitions that you want to include in your graph definition.

    TIP To remove the active graph edge from the graph definition, click **Remove from Graph**.

    CAUTION Removing a graph edge from a graph definition invalidates any view properties that use this graph edge for view definitions that are tied to this graph definition.

11. Click **OK** to save the new graph definition.

    NOTE As you add edge definitions to the graph definition, the UML in the **Current graph definition** pane is updated. The starting interface for the graph definition appears at the upper left of the UML view. Any graph edges that originate from this starting interface definition are shown with lines coming from the interface definition. For each graph edge in the graph definition, the edge definition traversed and the name of the graph edge are shown in the UML. The edge definition is shown to the left of the graph edge and the graph edge is shown in italics.



See Also

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

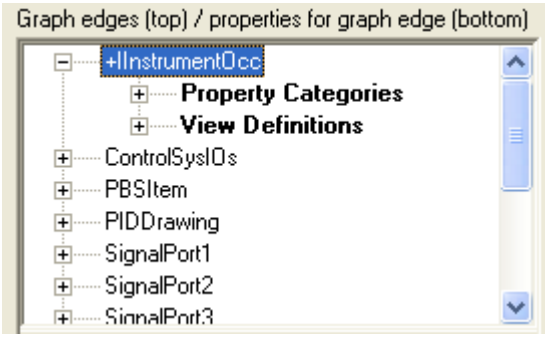
Create a View Definition

1. Open the **Editor** view. For more information, see *Create New Schema Objects* (on page 168).
2. Double-click **ViewDef** in the **Create** tree.
3. In the **New View Definition** dialog box, type a name for the new view definition in the **View name** box on the **Overview** tab.

NOTE A comparison view definition is a special type of view definition that is intended for performing comparisons between the two sets of data described by the comparison view definition.

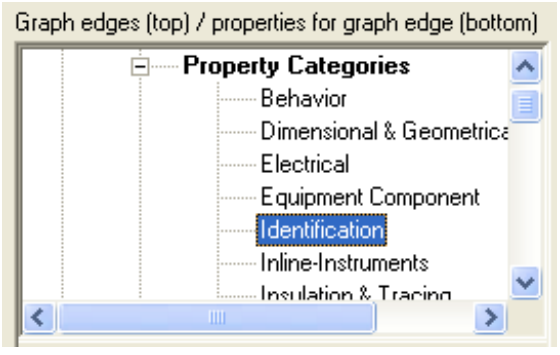
4. If you want the display name for the edge definition to be different from the name, type the display name in the **Display name** box.
5. Select the graph definition that you want to use as the starting point for the view definition from the **Graph name** box.
6. If you want to use this view definition to create ad-hoc reports in SmartPlant Foundation Desktop Client, go to the **Properties** tab and provide a description or name for the view definition on the **Desc** box.

   TIP When you create a new report in the Desktop Client, the value in the **Desc** box is the name you will see for this view definitions in the **New Report** dialog box.
7. Click the **Layout** tab.
8. In the **Graph edges** tree, expand the interfaces and edges that were included in the graph definition you selected at the basis for your view definition. This list will include the starting interface of the graph definition, any interfaces implied by the starting interface, and all edge definitions and relationship definitions included in the graph definition.



NOTE The **Graph edges** tree contains all of the graph edges for the directed graph definition used by the view definition.

9. Select the property category or graph edge for which you want to add properties to the new view definition.



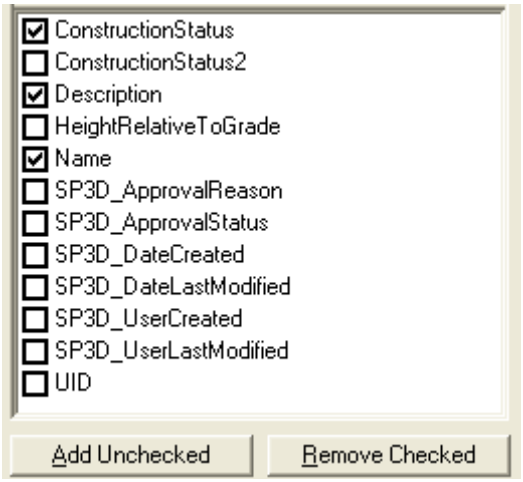
TIPS

- Under the graph edge node, expand the **Property Categories** node if you want to select from property categories for the graph edge or the **View Definitions** node if you

want to select from other view definitions that start at the end interface definition for the graph edge.

- If no other view definitions start at the end interface definition for the selected graph edge, the **View Definitions** node is not available.
- When you select a property category, the software displays properties for that category in the **Properties for graph edge** list. Properties exposed on interfaces that were not assigned a property category are included under **No Category**. For more information about property categories, see *Create an Interface Definition* (on page 171).
- When you select a graph edge under a view definition node, the software displays the view properties for that view definition that use the selected graph edge in the **Properties for graph edge** list.

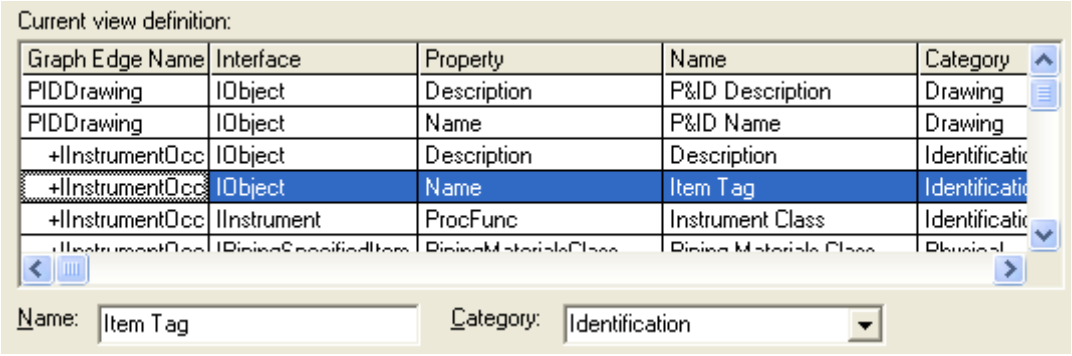
10. In the **Properties for graph edge** list, select the check box beside property names to add them to the view definition.



TIPS

- When you select the check box beside a property name, the property appears in the **Current view definition** table.
- If a property with the same display name already exists in the view definition, the display name for the property you add is changed to <graph edge name>_<property definition name>. To change the display name, see step 11.
- To find a property to add, you can type all or part of the property name that you want to search for in the text box below the **Properties for graph edge** list, and then click **Find**. To find the next property that matches the string you typed, click **Find Next**.
- You can also remove a property from the view definition by right-clicking the property in the **Current view definition** table and then clicking **Delete** on the shortcut menu or by clearing the check box beside the property name in the **Properties for graph edge** list.

11. To change the displayed name for a property in the view definition, select the property in the Current view definition table, and type a new display name in the Name box below the table. To change the property category for a property in the view definition, select the property in the table, and choose a different value in the **Category** list below the table..



TIPS

- Defining a different display name for a property allows you to differentiate among multiple properties with the same display name in the view definition. For example, many properties have a display name of "Name" but the properties actually represent the names of different objects. You can modify the display name to be more specific than "Name" for all the name properties you add to the view definition.
- When a custom view definition is configured to display properties for an object selected in the SmartPlant Foundation Desktop Client, the property category is used to organize those properties in the **Properties** window.
- You can change the property category for multiple properties by selecting several properties in the table and then selecting a new property category in the **Category** list.

12. Click **OK**.

Change the Order of Properties in a View Definition

The order that properties are listed in the grid is the default order in which they will appear in ad-hoc reports in SmartPlant Foundation Desktop Client. To change the order of properties in the view definition, do the following:

1. In the **Current view definition** table, select the property or properties that you want to move. To select multiple properties, hold down the CTRL key while you click multiple properties.
2. Right-click one of the selected properties.
3. Click **Cut** on the shortcut menu.
4. Right-click the row below where you want to move the properties in the table.
5. Click **Paste Insert**. Properties are inserted above the row you select.

See Also

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

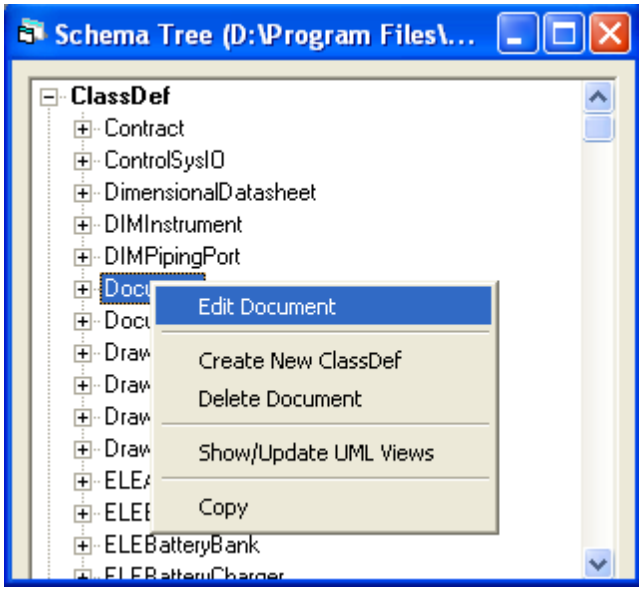
Create a Relationship Between Schema Objects

There are a number of ways to create relationships between schema objects. Since the relationships between objects are defined by the meta schema, you can create instances of those relationships, but the easiest way to create these relationships is by editing one of the objects. For example, if you want to add a realizes relationship between a class definition and an interface definition, the most direct way is to either edit the class definition and add the interface to the list of realized interfaces or to edit the interface and add the class definition to the list of class definitions that realize that interface.

Creating Relationships by Modifying the Object at One End of the Relationship

You can create relationships by editing the object at one end of the relationship using the **Edit** dialog box for that object. In the **Edit** dialog box, you can add, remove, or modify existing relationships by changing the values in the text boxes.

1. From any of the standard views, find and right-click the object you want to create a relationship for.
2. On the shortcut menu, click **Edit <Object Type>**.



3. In the **Edit** dialog box, use the provided boxes to create the new relationship.

   TIPS

   - Click ... to display a dialog box that allows you to select objects for each relationship type.
   - To remove existing relationships, click selected objects in the list to clear them.
   - To remove all currently related objects, click **Clear**.

4. Click **OK** to save the new relationships.

See Also

Relationship Definitions (on page 31)
Create New Schema Objects (on page 168)

Naming Conventions in the SmartPlant Schema (on page 146)
SmartPlant Schema Overview (on page 18)

Modifying Objects and Relationships in the Schema

In the Schema Editor, you can modify schema objects in many of the available views. For example, you can modify objects, interfaces, relationships, enumerated lists, and so on.

To modify objects, you can use any of the Schema Editor views.

You can also modify objects and relationships by right-clicking objects in the Schema Editor views. For example, in the Schema Editor tree view, you can right-click an existing map class definition, and then click **Edit <Map Class Definition Name>**.

See Also

Modify Existing View and Graph Definitions (on page 187)

Modify Existing View and Graph Definitions

The following topic provides an example of modifying view and graph definitions in the **InstrLoop** map class definition in the Schema Editor with the **Edit View Definition** command.

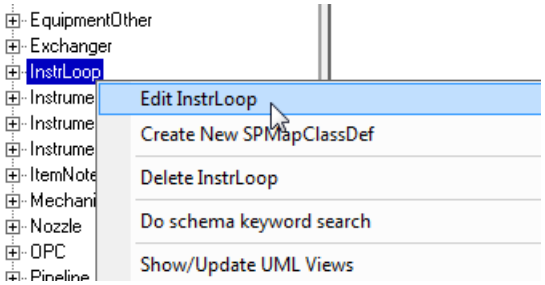
1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

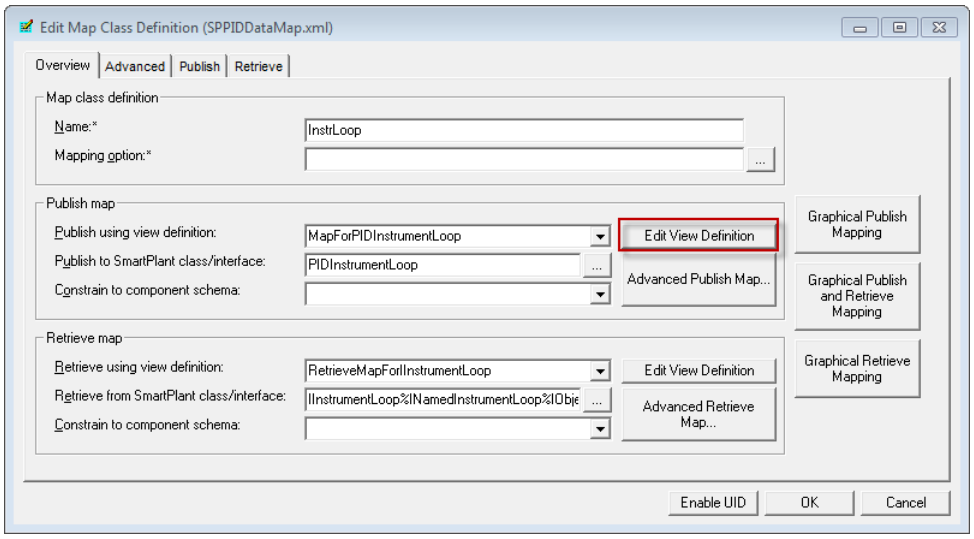
   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

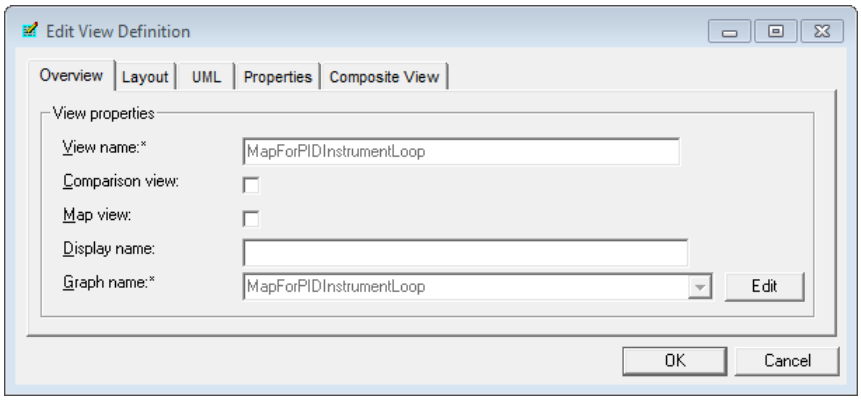
2. Click **View > Tool Schema**.
3. Click **OK** in the **View Tool Schema** dialog box.
4. In the tree view of the **Schema Editor**, right-click **InstrLoop** and select **Edit InstrLoop**.



5. Click **Edit View Definition** in the **Publish map** section of the **Edit Map Class Definition** dialog box.

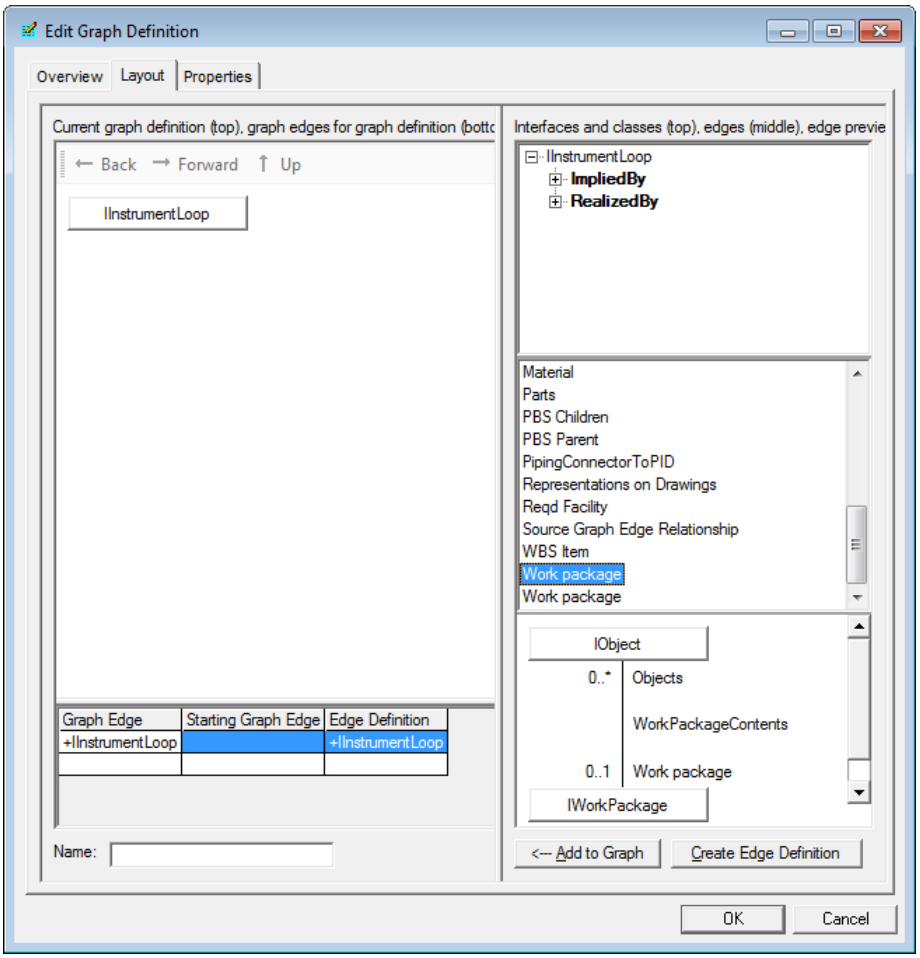


6. Click **Edit** next to the **Graph name** field in the **Edit View Definition** dialog box.

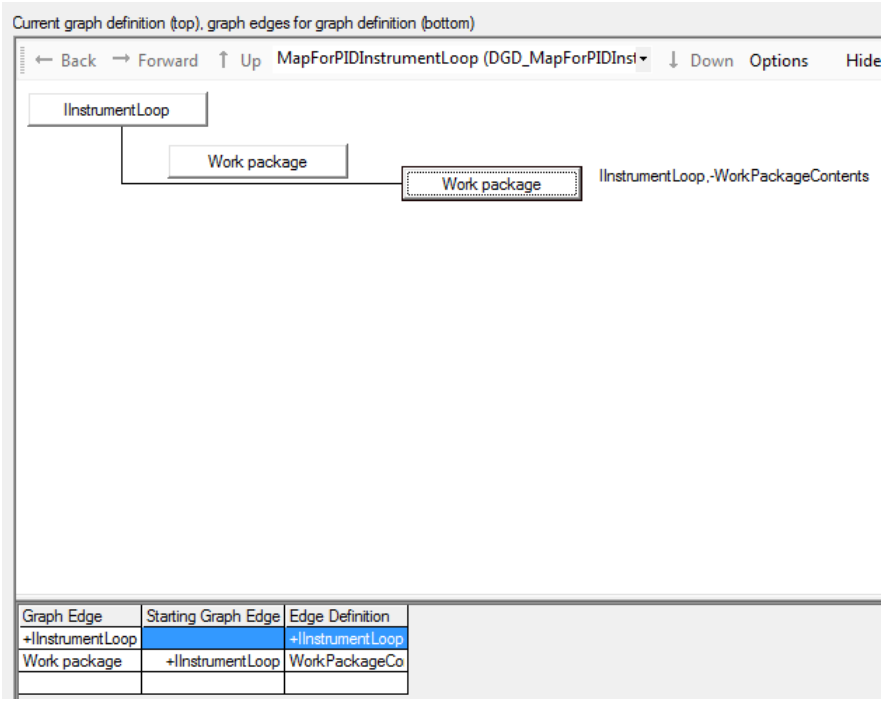


7. Select the **Layout** tab in the **Edit Graph Definition** dialog box.
8. Select the first instance of **Work package** in the edges list.

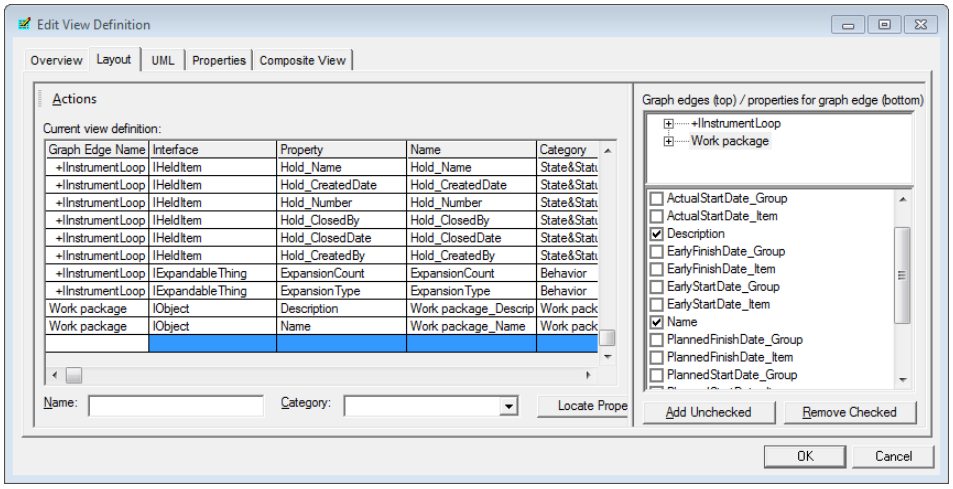
TIP The edges list is the middle pane on the right side of the dialog box.



9. Click **Add to Graph** to add the **Work package** edge to the current graph definition.



10. Click **OK** to save your changes and exit the **Edit Graph Definition** dialog box.
11. Select the **Layout** tab in the **Edit View Definition** dialog box.
12. Select **Work package** in the **Graph edges** section (top of the right pane), and select **Description** and **Name** from the properties list located in the bottom of the right pane.



13. Click **OK** to save your changes and exit the **Edit View Definition** dialog box.

NOTES

- The properties you added to the InstrLoop map class are available in all mapping interfaces. For example, you can map these properties to other properties with the graphical mapping user interface or the **Integrated Map Environment**.

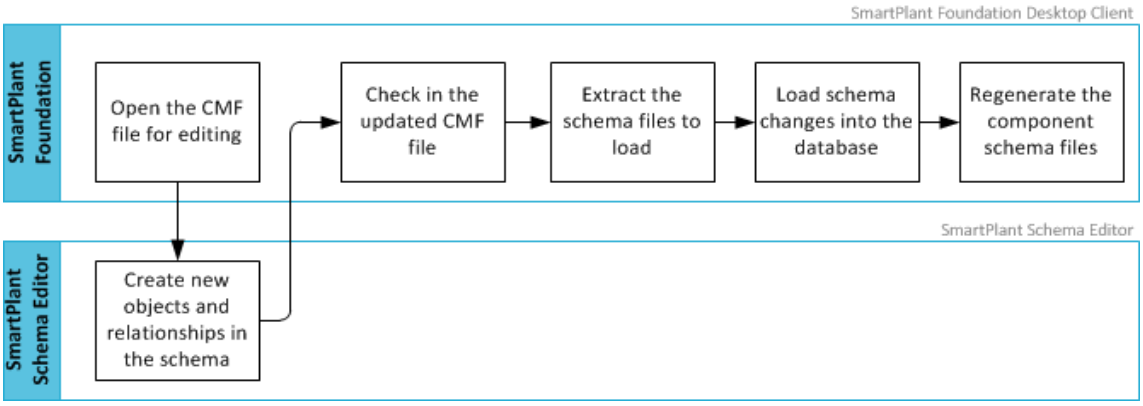
- You can use this workflow to modify the retrieve map view and graph definitions. For example, in the **Edit Map Class Definition** dialog box, click **Edit View Definition** in the **Retrieve map** section.
- Dynamic view definitions are not persisted. Any properties added to the view definition *that are not mapped* will not appear after exiting the Schema Editor.
- View properties across edges *that are mapped* and corresponding graph edges will be automatically added to the view definition and corresponding graph definition when the Schema Editor is re-opened.

Extending the Schema for SmartPlant Integration

When you are using the SmartPlant schema in an integrated environment, you should manage the schema CMF file using SmartPlant Foundation's document management functionality, which allows you to control changes to the schema and manage changes to multiple versions of the SmartPlant schema the same time.

When you manage the schema in the Desktop Client, you must check out the schema CMF file to make changes to it and then return it to the database to incorporate your changes.

The following diagram illustrates how to modify the delivered integration schema.

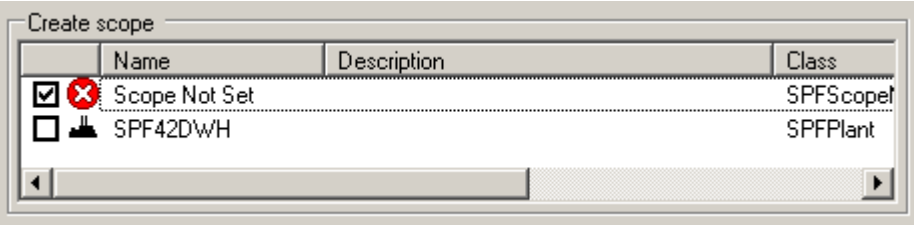


What do you want to do?

- *Open the CMF File for Editing* (on page 192)
- *Create new objects and relationships in the schema* (on page 198)
- *Check in the updated CMF file* (on page 198)
- *Extract the schema file to load* (on page 198)
- *Load schema changes into the database* (on page 199)
- *Regenerate the component schema files* (on page 203)

Open the CMF File for Editing

1. In the SmartPlant Foundation Desktop Client, set your update scope to **Scope Not Set**.



   For more information about setting the active scope, see *Set user options* in the Desktop Client Help.

2. Click **Find > Integration > Schema Documents**.

   IMPORTANT You must be logged on to the SmartPlant Foundation data warehouse database and have a schema administrator role to access commands on the **Integration** menu.

3. Type **CMF*** in the **Find** dialog box, and click **OK**.
4. Right-click the CMF file, and then click **Edit > Check Out**.
5. Click **OK** in the **Check Out** dialog box.
6. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
7. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
8. Click **OK** to set the configuration.

   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.You are now ready to start extending your integration schema. By default, your changes are stored in the **Customization** pseudo file; however, you can also perform any of the following options, if necessary:

- *Create a New Pseudo File in the Schema Editor* (on page 193)
- *Merge a Previously Generated Schema into a Pseudo File* (on page 195)

- *Convert an Existing Pseudo File into a Physical Pseudo File* (on page 197)

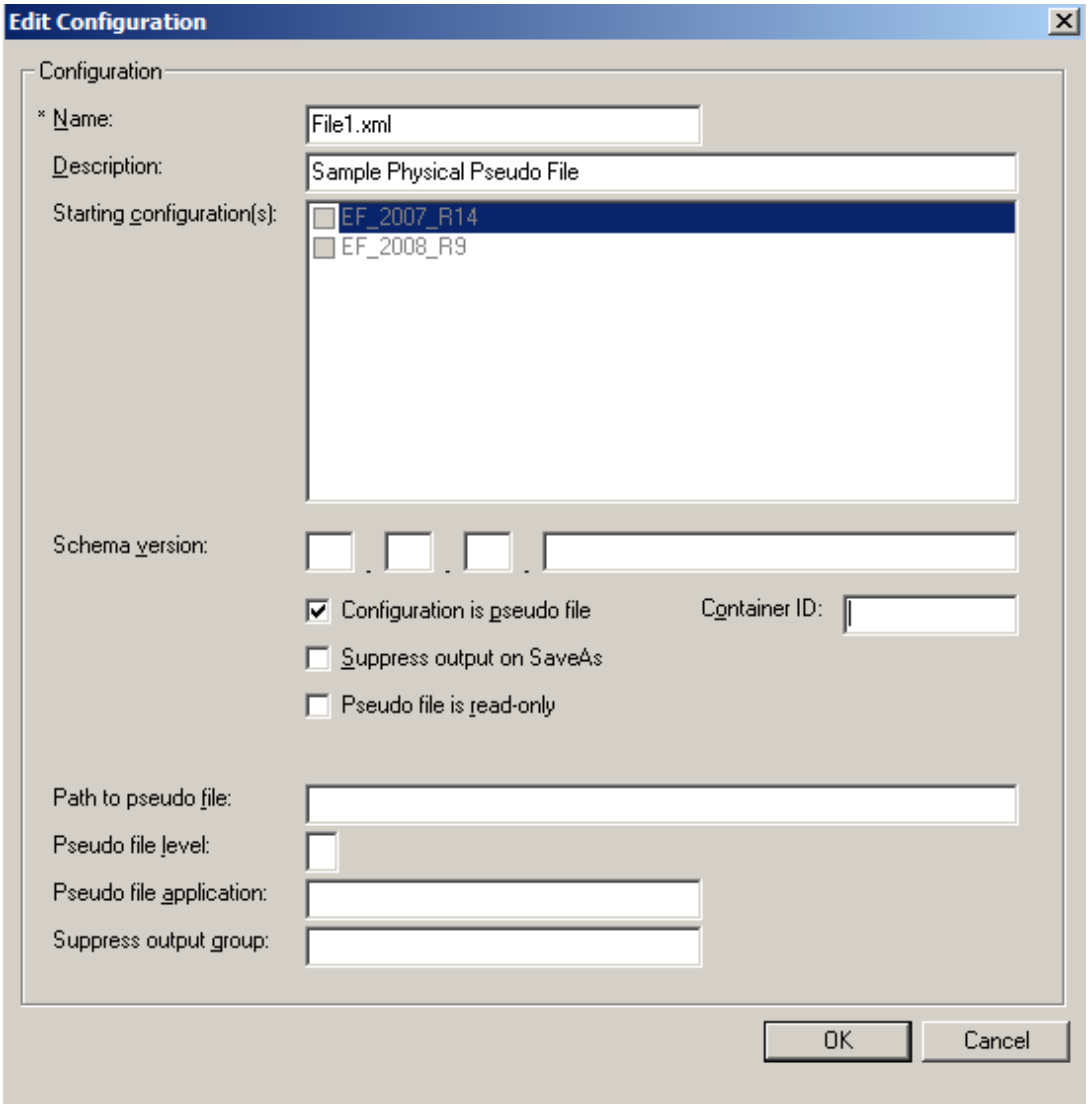
Create a New Pseudo File in the Schema Editor

The following steps explain how to create a pseudo file in Schema Editor. Pseudo files allow for the objects within the configurations to be logically organized into files. Creating a pseudo file can be helpful to use when making changes to the SmartPlant Foundation schema or creating customizations.

Create the Pseudo File

1. After you check out the schema CMF file from the Desktop Client and launch the Schema Editor, click **Tools > Schema > Advanced > Create Configuration**.
2. On the **Edit Configuration** dialog box, type a name and description.

IMPORTANT The name must end in .xml in order to create the configuration as a physical file.



3. Select the option **Configuration is a pseudo file** to create a pseudo file.
4. Click **OK**.

NOTES

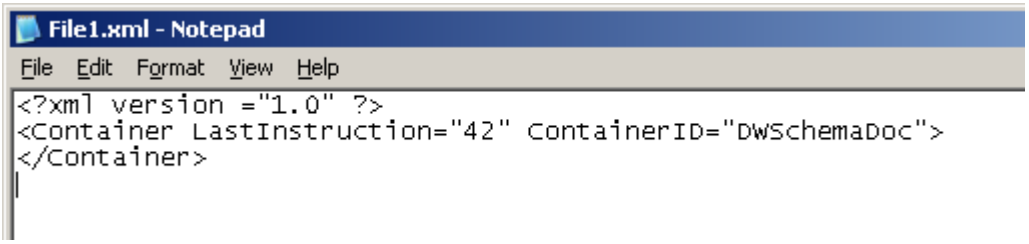
- You do not need to select an option for **Starting configuration** when creating a pseudo file. Selecting the option **Starting configuration** creates the configuration as a primary configuration rather than a pseudo file.
- You do not need to enter values in the **Path to pseudo file** box. Leaving the **Path to pseudo file** box blank creates the pseudo file in the same folder as the CMF file.
- The ContainerID for the file is set automatically.

Set up the Correct Configurations for New Objects

1. In the Schema Editor, click **Tools > Schema > Set Active Configuration** to open the **Set Active Configurations** dialog box.
2. Select the options to set up the correct configurations for new objects.
3. Select the new pseudo file from the **Pseudo file for new objects** list.
4. Click **OK**.
5. Click **File > Save > Schema File** to save schema files, including the new pseudo file.
6. Exit the Schema Editor.

Set the Container ID for the New Pseudo File

1. Open the pseudo file in a text editor.
2. Edit the pseudo file to manually set the ContainerID to a value that represents the use of the schema.



3. Save the changes.

NOTES

- The default ContainerID for a pseudo file is a numeric value automatically set when it was created.
- The ContainerID identifies schema previously loaded into the SmartPlant Foundation database when performing the compare during a load.
- The ContainerID in the header updates to match that in the pseudo file the next time the CMF file is opened.
- After setting the ContainerID for the pseudo file, the new schema can be modeled in the Schema Editor, or it can be cut and pasted into the physical pseudo file.

Merge a Previously Generated Schema into a Pseudo File

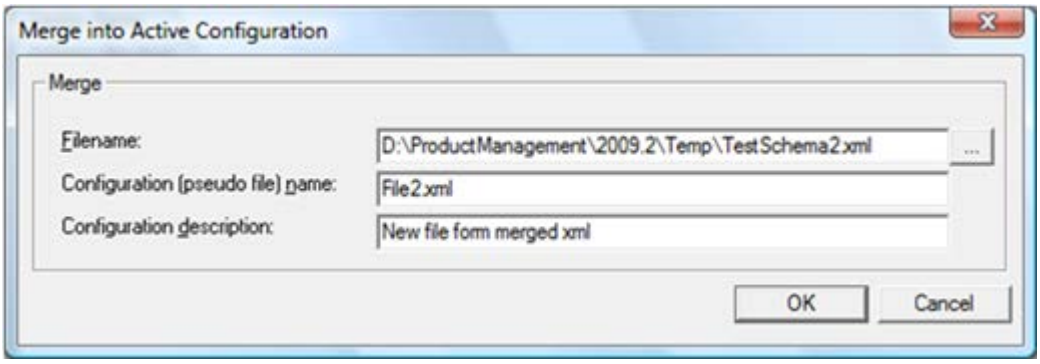
Schema created in a stand-alone .xml file can be imported into a new pseudo file using the Merge command.

Merge a Previously Generated Schema into a Pseudo File

1. In the Schema Editor, click **Tools > Schema > Set Active Configuration** to open the **Set Active Configurations** dialog box.
2. Select the options to set up the correct configurations for new objects.

NOTE You do not need to select the pseudo file in the **Pseudo file for new objects** list. The Active configuration in the command refers to the SmartPlant Foundation configuration.

3. Click **Tools > Schema > Advanced > Merge file into Active Configuration** to open the **Merge into Active Configuration** dialog box.



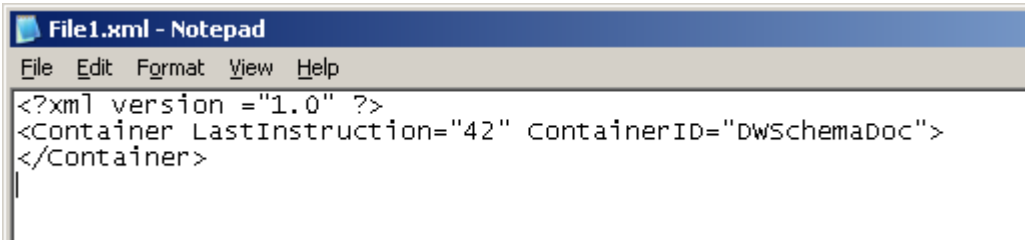
4. In the **Filename** box, select the .xml file to load.
5. Type a file name into the **Configuration (pseudo file) name** box.

   IMPORTANT The name must end in .xml in order to create the configuration as a physical file.

6. Type a description in the **Configuration description** box.
7. Click **OK**.

Set the Container ID for the Pseudo File

1. Open the pseudo file in a text editor.
2. Edit the pseudo file to manually set the ContainerID to a value that represents the use of the schema.



3. Save the changes.

Save Schema Files

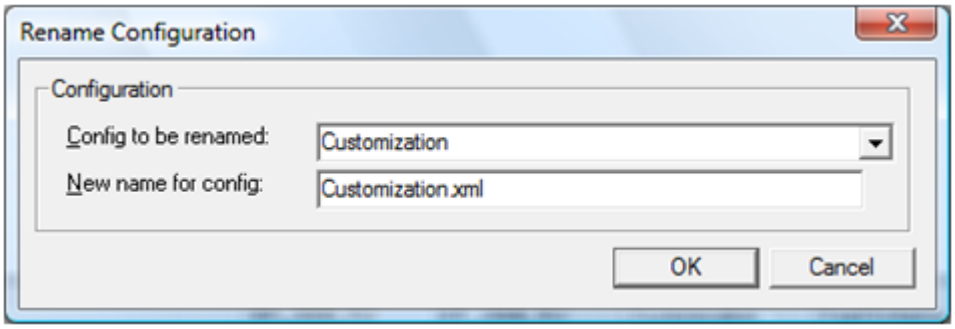
- In the Schema Editor, click **File > Save > Schema File** to save the schema files.

Convert an Existing Pseudo File into a Physical Pseudo File

If the schema is already modeled in the CMF in a pseudo file such as Customization, the schema can be moved to a physical file to make use of the improved schema loading process.

Convert the Pseudo File to a Physical Pseudo File

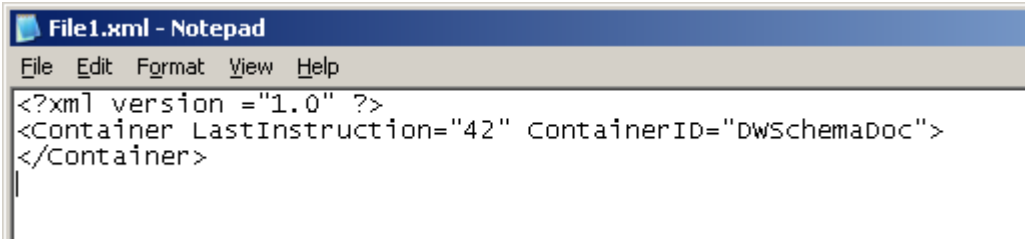
1. Click **Tools > Schema > Advanced > Rename Configurations** to open the **Rename Configuration** dialog box.



2. Select the file to be renamed in the **Config to be renamed** list.
3. Add the .xml extension on the name in the **New name for config** box to make the file a physical pseudo file.
4. Click **OK**.

Set the Container ID for the Pseudo File

1. Open the pseudo file in a text editor.
2. Edit the pseudo file to manually set the ContainerID to a value that represents the use of the schema.



3. Save the changes.

Save Schema Files

- In the Schema Editor, click **File > Save > Schema File** to save the schema files.

Create new objects and relationships in the schema

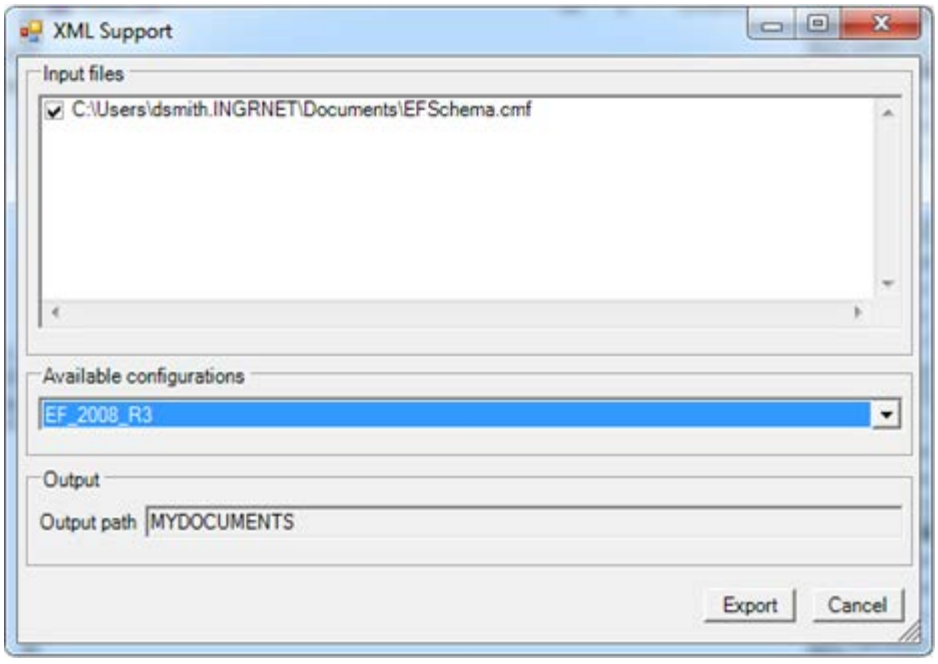
Once you have opened the appropriate environment in the schema editor, you can begin editing the integration schema to for use in an integrated SmartPlant environment. For detailed instructions for creating and editing schema objects, see *Create New Schema Objects* (on page 168).

Check in the updated CMF file

1. Open the SmartPlant Foundation Desktop Client.
2. Click **Find > Integration > Schema Documents**.
3. Click **OK** on the **Find Schema Documents** dialog box to find the CMF file.
4. Right-click the checked out CMF file, and click **Edit > Check In**.
5. Click **OK** to close the **Check in** dialog box and store the updated CMF file in the database.

Extract the schema file to load

1. Right-click the newest version of the CMF_Ver document, and click **Launch CMF File Manager (XML)**.

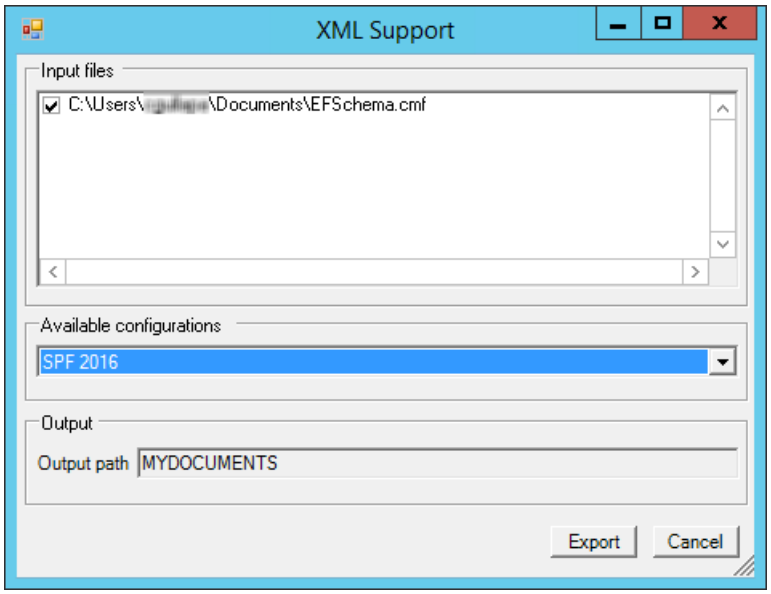


2. Select the CMF from the **Input file** list.
3. Verify that the latest configuration is selected in the **CMF** list under **Available configurations**.
4. Click **Export** to create an XML file with the schema information for the selected configuration.

TIP By default the output directory is the **My Documents** folder for the current user.

Prepare the pseudo file data to be loaded into SmartPlant Foundation

1. In the Desktop Client find the **configuration file** (CMF_ver). To find the CMF file, click **Find > Integration > Schema Documents**.
2. Right-click the new version of the .cmf file and select **Launch CMF File Manager (XML)**.



3. Select the XML file from the **Input file** list.
4. Verify that **SPF 2016** is selected in the **CMF** list under **Available configurations**.
5. Click **Export** to prepare the data to be loaded into SmartPlant Foundation.

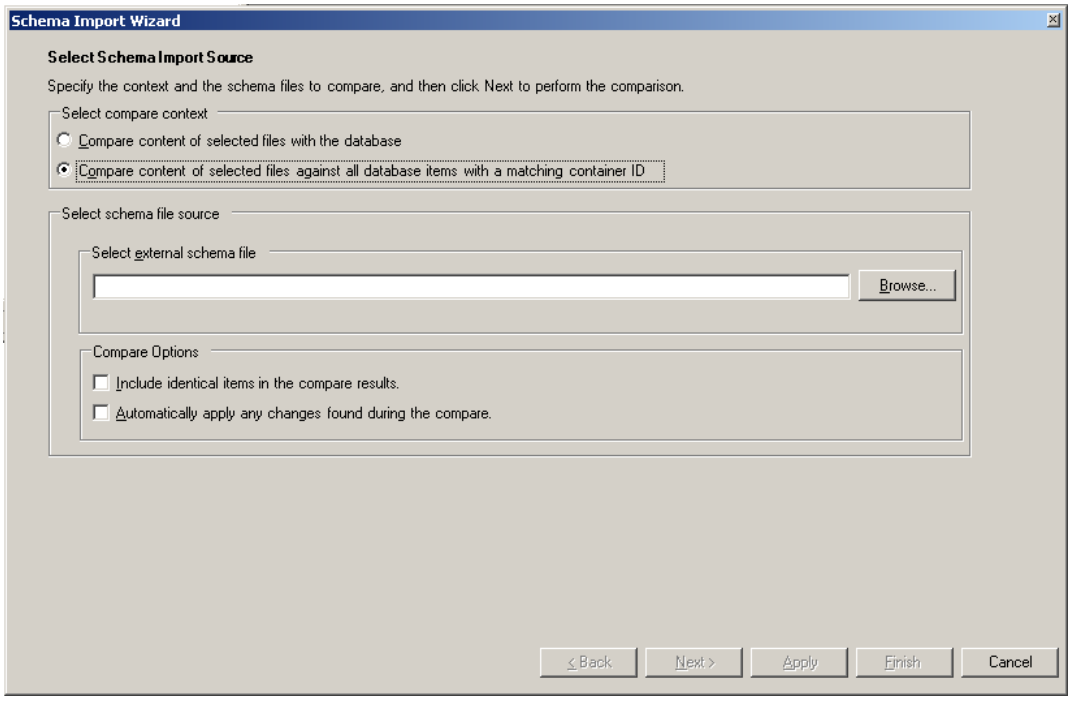
TIPS

- By default the output directory is the **My Documents** folder for the current user.
- If several pseudo files are attached to the schema document, you can select all of the files, including the CMF file, to process at one time. You can also select specific files to load by selecting individual files to process.

Load schema changes into the database

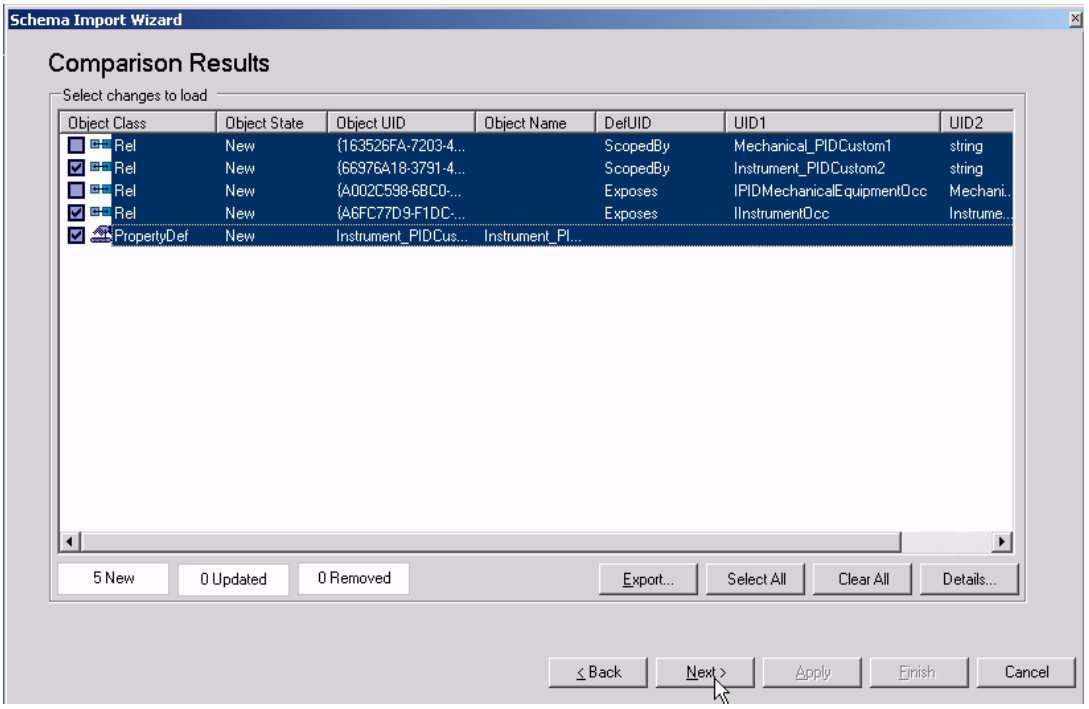
1. In the Desktop Client, click **Administration > Schema Import Wizard**.

2. Under **Select compare context**, click **Compare content of selected files against all database items with a matching container ID**.

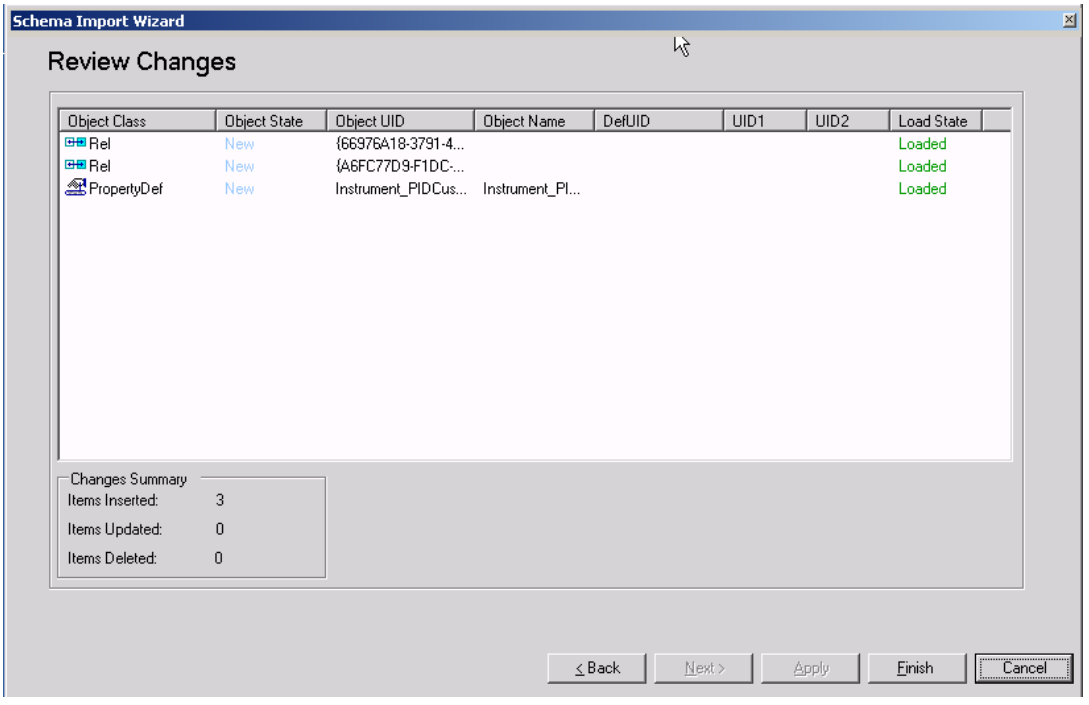


3. Click **Browse** beside **Select external schema file**, and browse to the schema file in the output folder defined in the **Output path** box in the **Build Support** dialog box.
4. Click **Next**.

5. In the **Select changes to load** table, select the schema changes you want to load into the SmartPlant Foundation database, and then click **Next**.



6. Under **Review Changes**, review the schema changes, and then click **Finish**.

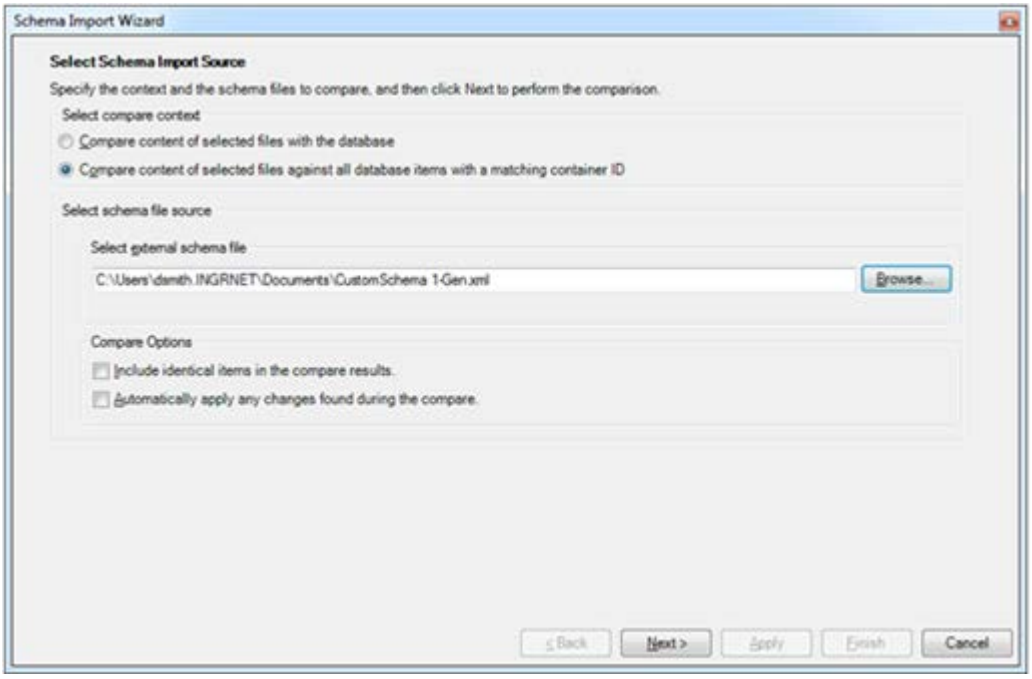


NOTE If you want to be able to see your new schema information in the Desktop Client, it must be available in a view definition. You should create the view definition in the Schema Editor prior to importing.

Load the pseudo file changes into the data warehouse database

1. In the Desktop Client, click **Administration > Schema Import Wizard**.
2. In the Schema Import Wizard, select **Compare content of selected file against all database items with a matching container ID**.

   Under **Select external schema file**, browse to schema files generated by CMF File manager in the directory where the software created the CMF files you prepared to be loaded.



3. Click **Next**.

   NOTE Schema Import Wizard compares the generated schema with the SmartPlant Foundation database and displays the results. This process can takes time, so please be patient.

4. Under **Comparison Results**, select all the items, and then click **Next**.

   NOTE If the items were all successfully loaded, then status in the **Load State** column is set to **Loaded**. If not, the status is either **Passed** or **Failed**.

5. Click **Finish**.

Regenerate the component schema files

The adapters use the component schema files, and these need to be regenerated after changes have been made to the integration schema. There are commands in the Schema Editor for regenerating all, or just a selected component schema file. You can also regenerate all the component schemas from the SmartPlant Foundation Desktop Client.

- If the CMF file has changed, it is recommended you regenerate all the component schemas because the change may affect many components.
- If a pseudo file has been changed, the change may be localized to one component, in which case the single component schema can be regenerated.

The schema component file differs from the actual schema file just generated; it is a full stand-alone definition of the complete schema including all dependencies, such as the property definitions, all scoped by enumerated lists, and so on.

You can regenerate all the component schemas in one of two ways:

- By using the **Regenerate Schema Files** command on the shortcut menu of the CMF document in SmartPlant Foundation. This uses the scheduler to regenerate the files and automatically put them in the delivered location.
- By using the Schema Editor:
  - Open the CMF file with the Schema Editor, selecting the applicable schema version.
  - Click **Tools > Schema > Generate Component Schema** to create the component schema files in the same directory as the CMF files. Then move the scheme files to the location specified for schema files for the site in the Server Manager.

    TIP It may be necessary to manually remove any copies of these files from the user's temp directory.

To regenerate a new component schema file for a single schema component, use the Schema Editor:

- Open the CMF file with the Schema Editor, selecting the applicable schema version.
- Click **Tools > Schema > Generate Component Schema**.

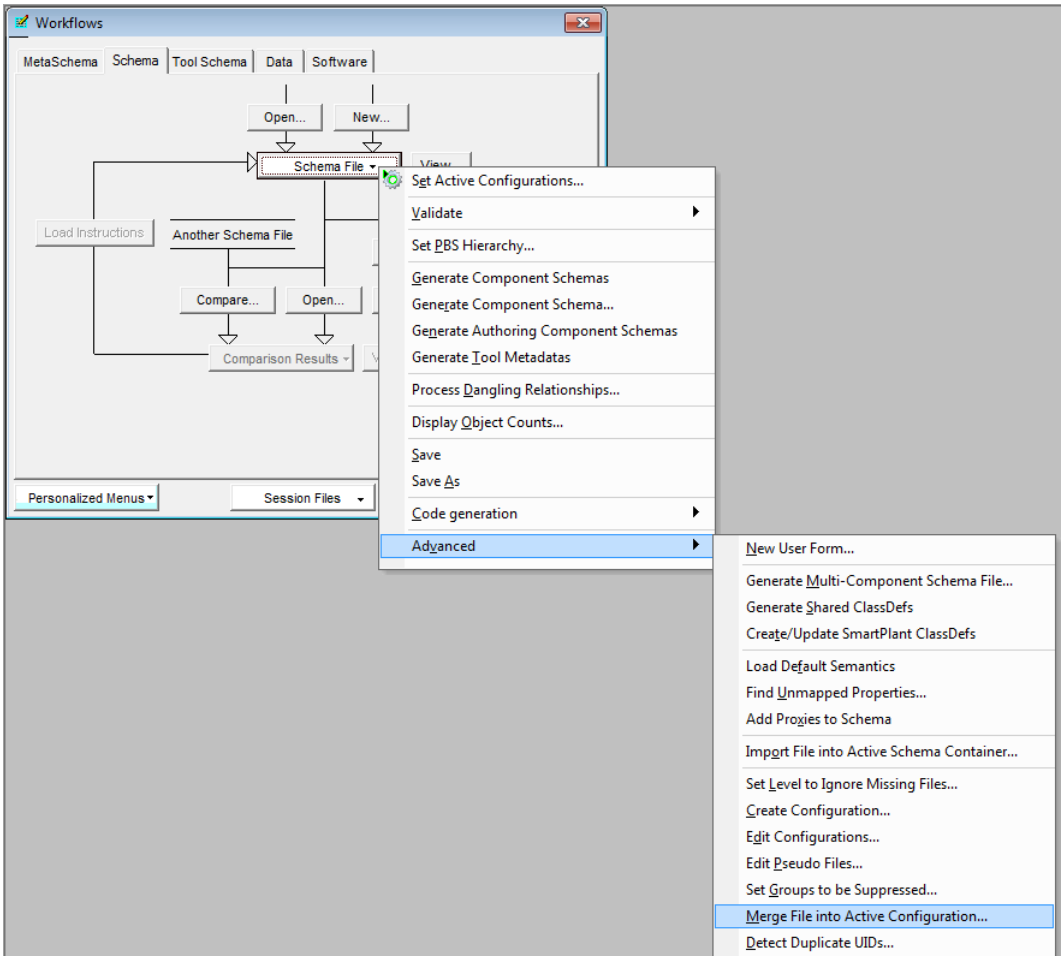
  This presents a list of component schemas for selection and then creates the new component schema file in the same directory as the CMF files. This needs to be moved to the location specified for schema files for the site in the Server Manager.

Import Existing Customizations in Extensions.xml into the CMF File

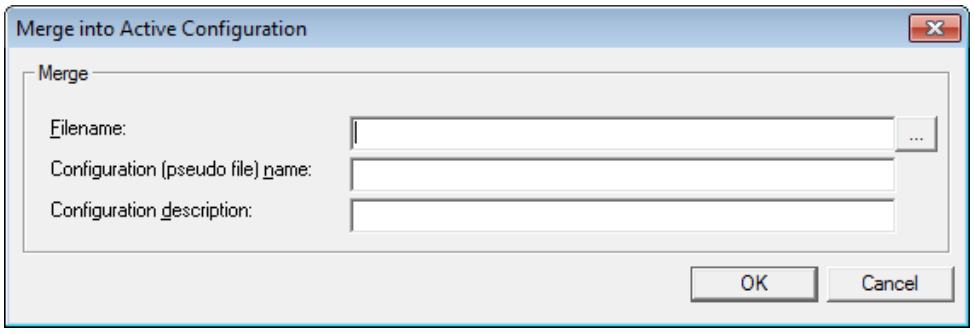
To migrate from 2007 (3.8) to 2008 (4.X) or higher, you can import any customizations made into the upgraded CMF file. Since these customizations were made with the 2007 (3.8) schema, it is recommended that the customizations go into both the 2007 (3.8) and 2008 (4.X) or higher configurations in the CMF file. The following procedure explains how to import these customizations.

1. Check out the CMF file in SmartPlant Foundation and launch Schema Editor.
2. Set the active configurations to both configurations.
3. Click **Tools > Schema > Advanced > Merge File into Active Configuration**.

You can also select the equivalent command from the **Workflows** menu.



4. On the **Merge into Active Configuration** dialog box, identify the name of the file (…\Extensions.xml).



5. Enter a name for a new pseudo-file configuration (such as Extensions)
6. Click **OK**. The objects in the specified extensions file should then be added to both the configurations.

Move Customizations from Project to Project

The process for moving customizations from one project to another is:

- Compare the delivered schema configurations with the customized schema configurations and generate instructions for the differences.
- Use these instructions to update the non-customized project with the customizations.

An alternative procedure that will work if the project to be customized is based on the same schema delivery and hasn't yet been customized is:

- Compare the non-customized project schema with the customized project schema and generate instructions for the differences.
- Use the differences to update the non-customized project schema.

If the non-customized project is based on the same schema delivery and this project has no customizations, then the CMF file from the customized project can simply be copied and replace the CMF file from the non-customized project.

Comparing Delivered and Customized Schema Configurations

To compare the delivered and customized schema configurations, the delivered schema file must be available. If this file no longer exists on a customer's machine, then the EFHistory.cmf file being delivered with the SmartPlant Foundation upgrade software can be used to come up with the delivered schema for the corresponding schema configuration.

The comparison of the delivered and customized schema configurations should be done by opening the delivered schema file and setting the appropriate active configuration. A comparison should then be made with the corresponding configuration in the customized schema file and the instructions generated for the differences between the two configurations.

To update the non-customized schema, the non-customized schema file should be opened and the active configuration set. The previously generated instructions should then be loaded into this configuration.

Direct Comparison of Non-Customized and Customized Schema Files

If the non-customized schema file can simply be replaced by the customized schema file, then copying the customized CMF file and replacing the non-customized CMF file with this file is the simplest solution.

If either there has been some unique customizations done to the "non-customized" schema file or if some but not all of the customizations done to the "customized" schema file are to be retrofitted into the "non-customized" schema, then the process should be to compare the non-customized schema file with the customized schema file and selectively update the non-customized schema file with the desired subset of differences between the two files.

This type of comparison is done by opening the non-customized schema file, setting the appropriate active configuration and doing a comparison with the corresponding configuration in the customized schema file with the command to Selectively Update based on the comparison. The desired subset of differences to reconcile can then be user selected.

Extend the Existing Stand-Alone SmartPlant Foundation Schema

Regardless of whether you want to extend the SmartPlant Foundation schema to create custom class definitions and relationships or a simple custom property, the following work process describes how you open the stand-alone schema files in the SmartPlant Foundation Schema Editor, save your changes to an extension file, and load the changes into the SmartPlant Foundation database.

As you make changes to the schema, you should store your modifications in a schema extensions file, which is added to a configuration file. The extension file is dependent on the original SmartPlant schema, which is set to read-only in the configuration file. For more information, see *Create a schema extension file* (on page 207) and *Modify a File Configuration* (on page 94).

When you make modifications to the schema using an extension file, any modifications that you make to the schema are saved in that extension file, not the delivered SmartPlant schema files. When you create new relationship definitions in your extension file, at least one object at one end of the relationship definition must exist in your extension file. For example, you cannot create a new relationship between two objects in the SmartPlant schema. However, you can create relationships between objects in your schema extension file and the SmartPlant schema.

For more information about using configuration files to manage schema changes, see *File Configurations* (on page 93).

The delivered model files are normally located in: C:\Program Files (x86)\SmartPlant\Foundation\<*software version*>\Models.

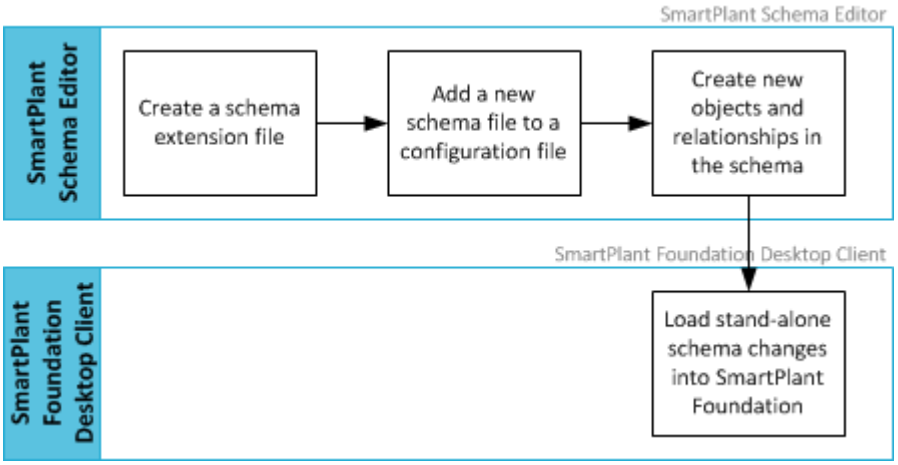
NOTE Many of these directories contain configuration files and sub directories for schema diagrams.

The root directory contains model definition files (.xmlsrc) that control which sets of model files are to be built into a database using a SmartPlant Foundation administrator role as described later in this document.

For example:

Authoring.xmlsrc lists the following layers

- SchemaComponent
- Bootstrap
- Core
- Common
- Auth
- AuthExample
- AuthPlantData



What do you want to do?

- *Create a schema extension file* (on page 207)
- *Add a new schema file to a configuration file* (on page 208)
- *Create new objects and relationships in the software* (on page 209)
- *Load stand-alone schema changes into SmartPlant Foundation* (on page 209)

Create a schema extension file

You should create your own extension file to organize your extensions to the delivered schema.

1. Open the SmartPlant Schema Editor without opening any existing configuration or schema files.
2. Click **File > New > Schema File**.

   TIP You can also click the **New** button above the **Schema File** button in the **Workflows** dialog box to create a schema extensions file.
3. In the **New Schema File** dialog box, define the name and location for your schema extension file, and then click **Save**.

NOTES

- Save your extension file to the same folder as the delivered schema files on which you want to create your extensions.
- To use your user-defined extensions file, you must link it to the SmartPlant schema using dependencies and a configuration file. For more information about setting dependencies and creating a configuration file for using your extension file, see *Add a new schema file to a configuration file* (on page 208).

Add a new schema file to a configuration file

1. In Schema Editor, click **File > File Configurations > Open Configuration** to open the configuration file that includes the delivered schema files on which you want to build your custom schema extensions. An example is: [Product Directory]\SmartPlant\Foundation\[*software version]*\Models\Core\900-AllCore.cfg.
2. Open the new schema file that will store your extensions.
3. Click **File > File Configurations > Edit Configuration**. The **Edit Configuration** dialog box appears.
4. Using the check boxes in the last column on the right, choose the delivered schema files on which the new file will be dependent.
5. In the **Active schema file** box, select the custom extension file.
6. In the last row of the table, find the custom extension file and make sure the value in the **Editable?** column is **Yes**. If the value is **No**, click the cell to switch the value to **Yes**.
7. Switch the values in other cells in the **Editable?** column as necessary to ensure they are all set to **No**.
8. Close the **Edit Configuration** dialog box.
9. Click **File > File Configurations > Save As Configuration** to create a new configuration file that will provide the working environment for you to create custom schema objects.

   IMPORTANT Do not click **Save Configuration** as this will replace the delivered configuration file rather than create a new one.

   TIP When prompted, be sure to save the changes made to the custom extension file. Information about the active file and which files are editable is saved in the configuration file, but all file dependency information is saved in the extension file.

   NOTE It is best to save the .cfg and schema file to your own custom directory. This means that the relative paths in the .cfg file will no longer point to the delivered schema files. Open the configuration file in a text editing application, and replace the relative paths with full paths.

   For example:

   ```
   `ASPF`SPF

   `MS`C:\Program
   Files\SmartPlant\Foundation\2016\Models\BootStrap\SCHEMA_SchemaCompo
   nentExtensions.xml

   `S`C:\Program
   Files\SmartPlant\Foundation\2016\Models\BootStrap\SCHEMA_BootStrap.x
   ml!N

   `AS`C:\Program
   Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_MinEFSchema-Gen.
   xml!N

   `AS`C:\Program
   Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFCoreSchema.xm
   l!N
   ```

```
`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFCoreDocMgtSch
ema.xml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFClaimSchema.x
ml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFDesignDocumen
tSchema.xml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFProgressSchem
a.xml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFSchedTaskSche
ma.xml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFWorkflowSchem
a.xml!N

`AS`C:\Program
Files\SmartPlant\Foundation\2016\Models\Core\SCHEMA_SPFCoreXmtlSchem
a.xml!N

`AS`.\SCHEMA_CNDocs.xml!Y

`ActS`.\SCHEMA_CNDocs.xml!Y
```

Create new objects and relationships in the software

Once you have opened the appropriate environment in the schema editor, you can begin editing the stand-alone SmartPlant Foundation schema or creating custom schema objects for use with SmartPlant Foundation. For detailed instructions for creating and editing schema objects, see *Create New Schema Objects* (on page 168).

Load stand-alone schema changes into SmartPlant Foundation

1. In the Desktop Client, click **Administration > Schema Import Wizard**.
2. Under **Select compare content**, click **Compare content of selected files with the database**.
3. Click **Browse** beside **Selected external schema file**, and locate the schema file that includes the changes you want to load.
4. Click **Next**.
5. In the **Selected changes to load** table, select the schema changes you want to load into the SmartPlant Foundation database, and click **Next**.
6. Under **Review Changes**, review the schema changes, and click **Finish**.

Creating an alternate plant breakdown structure (PBS)

In an integrated environment, engineering data is organized using a plant breakdown structure (PBS). For example, a PBS for organizing plant data might be plant/area/system/subsystem or plant/area/unit.

The PBS hierarchy defined in the SmartPlant schema is used to create the PBS hierarchy in SmartPlant Foundation. This hierarchy is communicated to integrated authoring tools when SmartPlant Foundation publishes the plant breakdown structure and the tools retrieve it.

Integration Hierarchy Limitations

The integrated environment imposes the following limitations on custom hierarchies, based on requirements from the authoring tools. This list includes the tool imposing each restriction, so that customers can factor these limitations into their hierarchy setup based on their current and future tool usage.

- The hierarchy must have a minimum of 3 levels. This limitation is imposed by SmartPlant Instrumentation.
- The names of hierarchy items cannot be changed once created. This limitation is imposed by SmartPlant Foundation.
- The hierarchy scheme cannot be modified after the plant is created. This limitation is imposed by SmartPlant P&ID and SmartPlant Foundation.
- If using a unit in the hierarchy, the unit number must be unique. Unit names can be duplicated across the hierarchy. This limitation is imposed by SmartPlant Instrumentation.

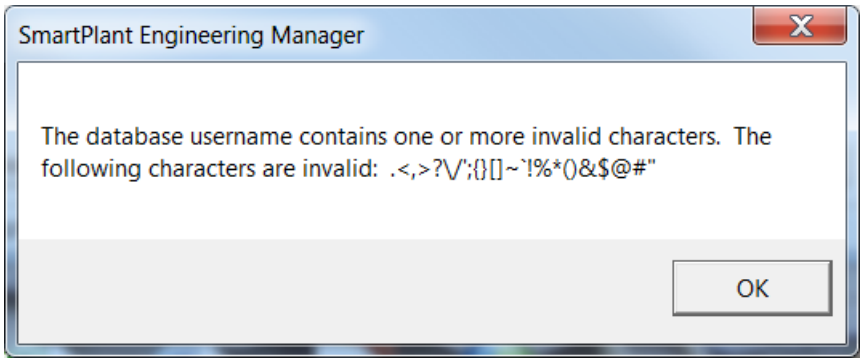
Plant Naming Guidelines

Use the following guidelines when naming your plant, or top-level hierarchy items, which will be used in an integrated environment. These limitations are imposed by the authoring tools and the database software. For these guidelines, "plant" is used as an example of a top-level hierarchy item.

IMPORTANT

- Do not use special characters at all in any part of a plant name or plant breakdown structure such as area or unit. These include .<,>^=+:?|/';{}[]~`!%*()&$@#".

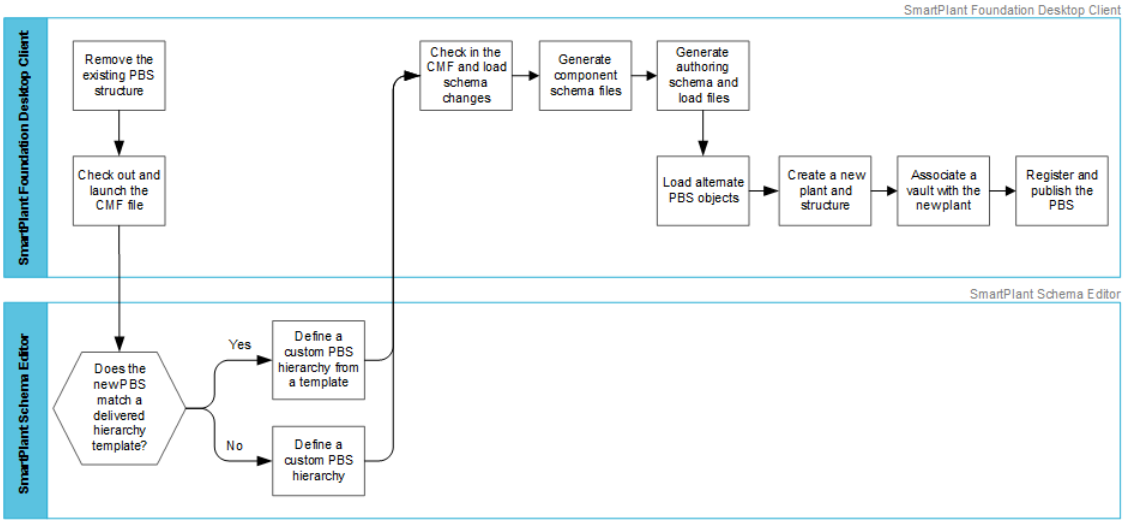
- A warning message that the characters are invalid is displayed if you use any special characters.



NOTES

- The plant name is case-sensitive and cannot exceed 20 characters.
- The plant name must be unique within the current domain. This limitation is imposed by SmartPlant Instrumentation.
- Intergraph recommends that the SmartPlant Instrumentation domain schema name and domain name match the plant name.
- Intergraph recommends that the plant name be the same name across all authoring tools.
- You can use digits and spaces within the body of the plant name, for example Plant _1. However, SmartPlant Foundation treats spaces the same as underscores; so if you name a new plant with a name that already exists, but use a space or an underscore to differentiate, an error message will state it is already in existence. For example Plant 1 and Plant_1 are considered the same.

CAUTION If you enter a space as the first character of your plant, area, or unit name, SmartPlant Foundation allows you to register the plant. However, when you publish the Plant Breakdown Structure (PBS), an error message is displayed. See the *SmartPlant Foundation Troubleshooting Guide* for further information.



What do you want to do?

- *Remove the existing PBS structure* (on page 212)
- *Check out and launch the CMF file* (on page 213)
- *Define a custom PBS hierarchy from a template* (on page 214)
- *Define a custom PBS hierarchy* (on page 215)
- *Check in the CMF and load schema changes* (on page 218)
- *Generate component schema files* (on page 220)
- *Generate authoring schema and load files* (on page 221)
- *Load alternate PBS objects* (on page 224)
- *Create a new plant and structure* (on page 225)
- *Associate a vault with the new plant* (on page 226)
- *Register and publish the PBS* (on page 226)

Remove the existing PBS structure

If you are starting from a database that already contains the PBS structure, such as spf42adw (the delivered database for integration), you must remove the PBS structure before you can create a custom hierarchy. To remove the existing PBS structure from the spf42adw database, you can use files delivered by default with SmartPlant Foundation. The files are located in the *<SmartPlant Foundation Installation Location>\Samples\SampleData* directory in the AlternativePBS\AdwDatabaseWithPBS folder.

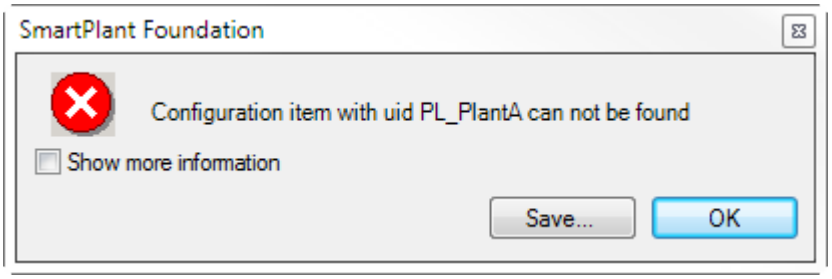
The RemoveExistingPBS.xmlldr file in this directory references all the scripts that are required to remove the existing PBS.

NOTE Make sure the Query Scope is not defined before running the load files.

IMPORTANT

- You may need to edit the paths to the scripts in the RemoveExistingPBS.xmlldr file before you run it.
- The delivered files will remove the sample plants delivered with the SmartPlant Foundation sample data. They will not remove custom plants you have created in the site. However, once the PBS has been replaced, you will no longer be able to use any plants that existed with the previous PBS structure.
- If any of the plants or projects contain data, it needs to be removed prior to starting this process. If existing data is not removed, you will receive the following error when processing the 03-DATA_Remove_ProjectsAndPlants.xml file: "Cannot terminate configuration item because it contains at least one object or rel." If you receive this error, you must terminate any objects and relationships in the plant or project and process the XML file again.

1. To remove the PBS, run **RemoveExistingPBS.xmlldr** in the Desktop Client loader. When you load the file, the following scripts are loaded and run in the correct order:
   - 01-ADMIN_Remove_AreaAndUnit.xml
   - 02-DATA_Remove_AreasAndUnits.xml
   - 03-DATA_Remove_ProjectsAndPlants.xml
   - 04-SCHEMA_Remove_PBSGraphDefs.xml
   - 05-SCHEMA_Remove_PBSViewDefs.xml
   - 06-SCHEMA_Remove_AreaAndUnit.xml
   - 07-SCHEMA_Remove_PlantExpansionDefaultGraphDef.xml
2. If you receive the following message that the plant you are removing cannot be found, click **OK**.



3. Process the load files again.
4. After you load the files, log out of the Desktop Client, and recycle the Application Pools. For more information on recycling the Application Pools, see the *Recycling the Application Pools* section of *Server Settings for SmartPlant Foundation Desktop Client* in the *Integration Setup Guide*.

NOTE This process may take several minutes to complete.

Check out and launch the CMF file

1. Log on to the SmartPlant Foundation Desktop Client.
2. Click **Find > Integration > Schema Documents**.

   IMPORTANT You must be logged on to SmartPlant Foundation Desktop Client with a schema administrator role to access the commands on the **Integration** menu.
3. In the **Find Schema Documents** dialog box, click **OK**.
4. Right-click the **CMF_Ver** document, and click **Edit > Check Out**.
5. Browse to the location to where you want to check out the file. (The **My Documents** folder is the default location).
6. Click **OK**.
7. In Windows Explorer, browse to the location where you checked out the CMF_ver document, find the EFSchema.cmf file, and launch that file in the SmartPlant Schema Editor.

TIP If the SmartPlant Schema Editor is not an option on the shortcut menu when you right-click on the EFSchema.cmf file, then open the SmartPlant Schema Editor, click **File > Open > Starting File**, and open the EFSchema.cmf file in the location you specified.

8. In the **Set Active Configurations** dialog box, select the schema version you want to view in the Schema Editor.

   NOTE Clicking **Advanced** opens a larger version of the **Set Active Configurations** dialog box with additional boxes. Make sure all configurations are selected in the **Set configurations for new objects** list and that **Customization** is selected in the **Pseudo file for new objects** list.

9. Click **OK.**

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

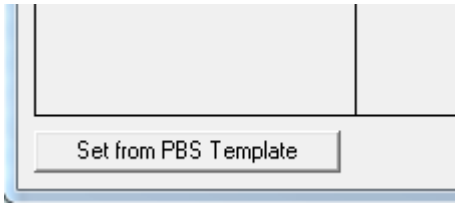
Define a custom PBS hierarchy from a template

NOTE If you do not want to define the PBS structure from an existing template, but would rather create a custom PBS hierarchy, see *Define a custom PBS hierarchy* (on page 215).

1. After you check out the schema CMF file from the Desktop Client and launch the Schema Editor, click **Tools > Schema > Set PBS Hierarchy.**

   TIP You can also click the **Schema File** button in the **Workflows** dialog box, and then click **Set PBS Hierarchy** on the shortcut menu.

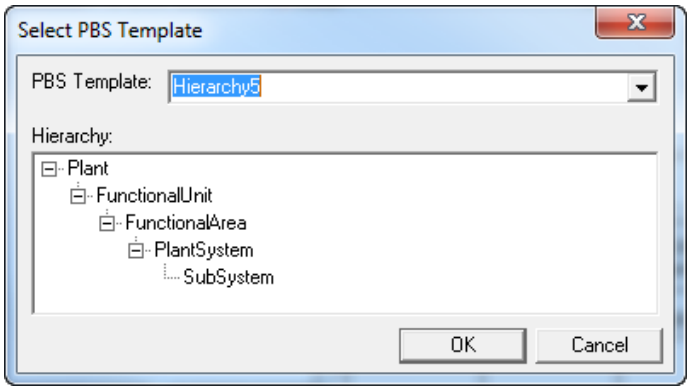
2. Click **Set from PBS Template**.



3. Select a PBS Template from the drop down menu in the **Select PBS Template** dialog box, and click **OK**.

   *The PBS hierarchy populates automatically in the **Define PBS Hierarchy** dialog box.*

   NOTE The Hierarchy2 through Hierarchy7 PBS templates match the delivered templates available in SmartPlant Engineering Manager and SmartPlant Electrical.



4. Click **OK** to save the new hierarchy.
5. In the **Identify Configurations for Instructions** dialog box, select the configuration or configurations to which you want to apply the PBS changes. Select all the versions of the schema used by your authoring tools.
6. Click **OK**.
7. Click **File > Save > Schema File** to save your changes to the SmartPlant schema.
8. Close the Schema Editor.

NOTE After you make changes in the schema, you must load the updated schema into the SmartPlant Foundation database. See *Check in the CMF and load schema changes* (on page 218).

Define a custom PBS hierarchy

NOTE Intergraph recommends you create the PBS hierarchy from a template. This workflow is optional and should only be used if you intend to create a PBS hierarchy without using a template. For more information on creating a PBS hierarchy from a template, see *Define a custom PBS hierarchy from a template* (on page 214).

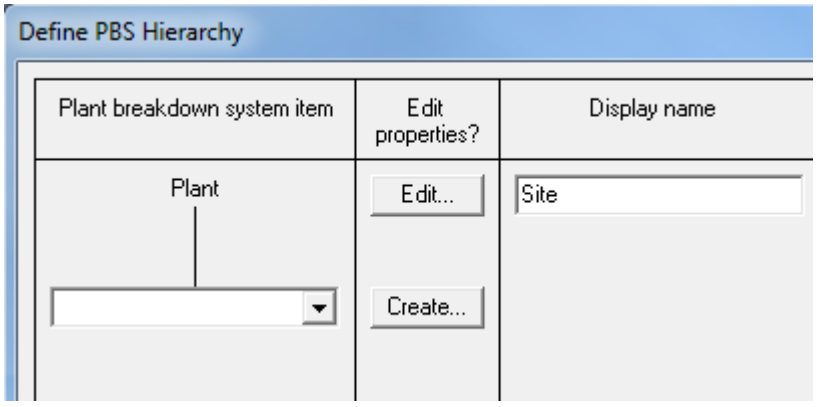
1. After you check out the schema CMF file from the Desktop Client and launch the Schema Editor, click **Tools > Schema > Set PBS Hierarchy**.

   TIP You can also click the **Schema File** button in the **Workflows** dialog box, and then click **Set PBS Hierarchy** on the shortcut menu.

2. If you do not want to use the default PBS items in your hierarchy, click **Delete Bottom Relationship** to delete the PBS items and relationships from the bottom up.

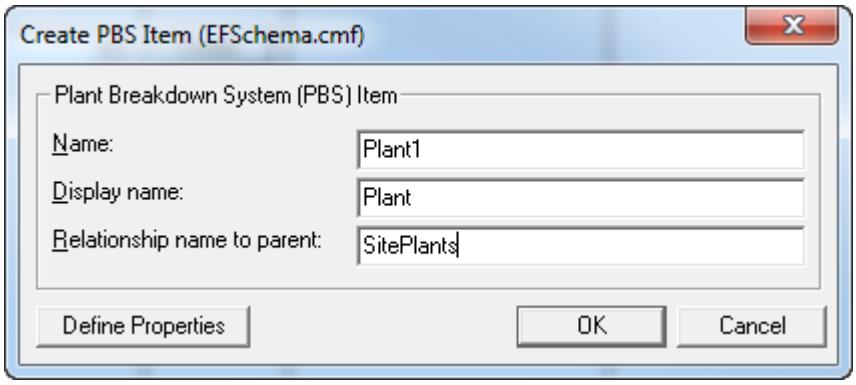
   TIP You cannot delete the Plant item at the top of the hierarchy. In the schema, the top-level item in the PBS is always a plant. However, you can change the display name for the plant.

3. To change the display name for the plant, type the new name in the text box in the **Display name** column.



4. To define additional PBS items, click **Create**.

5. In the **Create PBS Item** dialog box, define the name and display name. The software automatically builds the relationship name to the parent item in the hierarchy.



TIPS

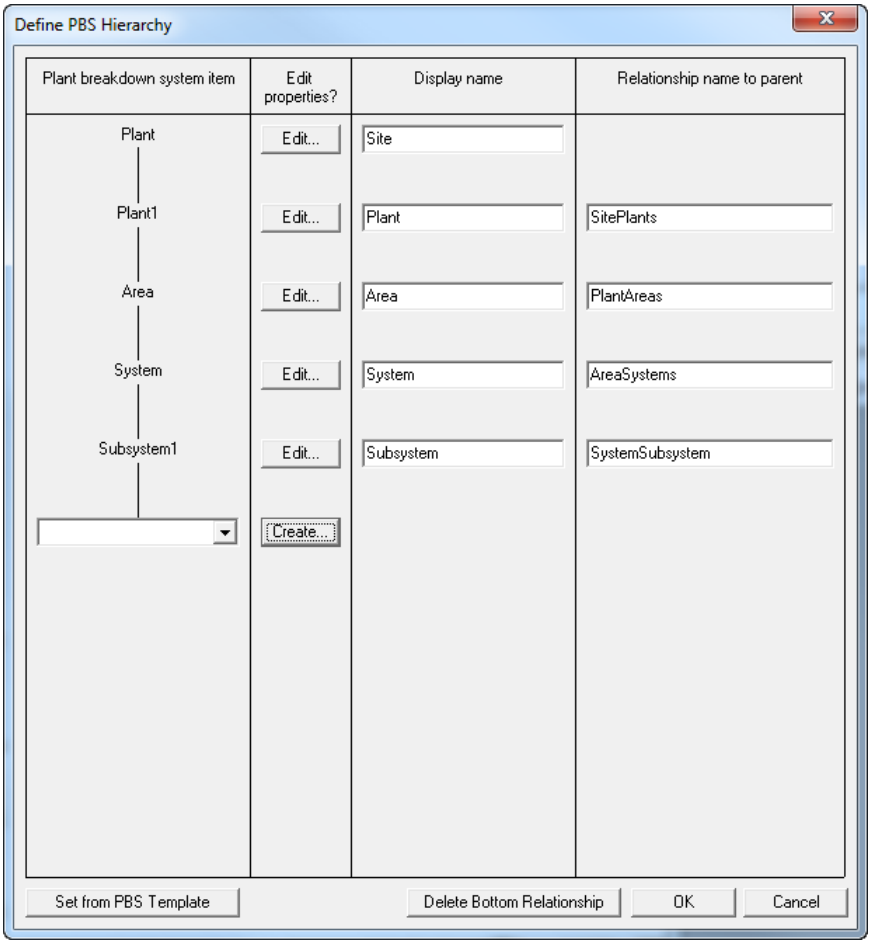
- Because plant is always the top-level class definition in PBS hierarchy in the schema even if you change its display name, you cannot use "Plant" as the name of another object in the PBS hierarchy.
- The recommended naming convention for the relationship between an item and its parent is <*Parent*><*Item*>s, for example, SitePlants. However, you can name this relationship any name that you want.
- PBSHierarchy DirectedGraphDef is automatically created when you create the PBS hierarchy.

6. Click **OK**.
7. Repeat steps 4-6 for each additional item that you want to add to the PBS hierarchy.

IMPORTANT

- Your PBS hierarchy must include at least 3 levels.

- PBS hierarchy cannot be changed after it is created in the SmartPlant Foundation client.



8. Click **OK** to save the new hierarchy.
9. In the **Identify Configurations for Instructions** dialog box, select the configuration or configurations to which you want to apply the PBS changes. Select all the versions of the schema used by your authoring tools.
10. Click **OK**.
11. Click **File > Save > Schema File** to save your changes to the SmartPlant schema.
12. Close the Schema Editor.

NOTE After you make changes in the schema, you must load the updated schema into the SmartPlant Foundation database. See *Check in the CMF and load schema changes* (on page 218).

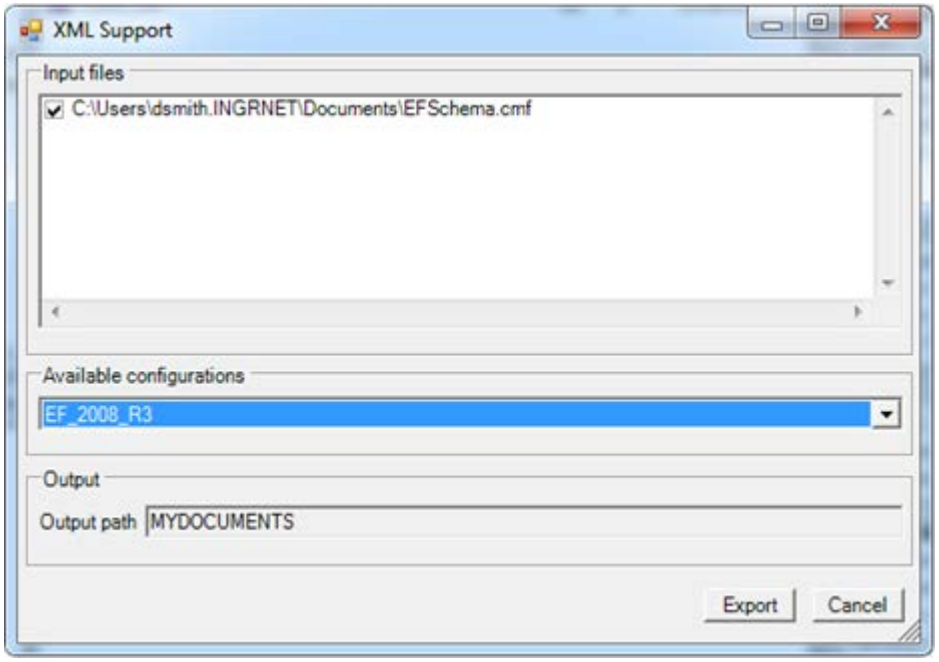
Check in the CMF and load schema changes

Check in the CMF file

After you modify the alternate PBS structure, check the CMF file back into SmartPlant Foundation. This operation currently compresses the file and checks the compressed file into the database.

Prepare the CMF data to be loaded into SmartPlant Foundation

1. In the Desktop Client find the **configuration file** (CMF_ver). To find the CMF file, click **Find > Integration > Schema Documents**.
2. Right-click the newest version of the CMF_Ver document, and click **Launch CMF File Manager (XML)**.



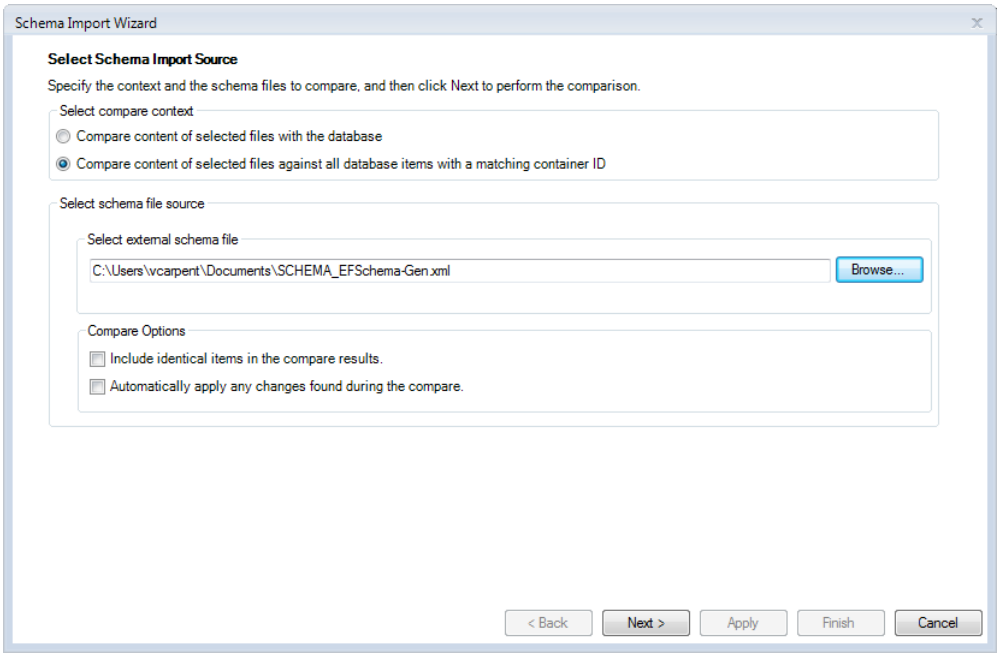
3. Select the CMF from the **Input file** list.
4. Verify that the latest configuration is selected in the **CMF** list under **Available configurations**.
5. Click **Export** to create an XML file with the schema information for the selected configuration.

TIP By default the output directory is the **My Documents** folder for the current user.

Load the CMF data changes into the Data Warehouse database

1. In the Desktop Client, click **Administration > Schema Import Wizard**.
2. In the **Schema Import Wizard**, select **Compare content of selected file against all database items with a matching container ID**.

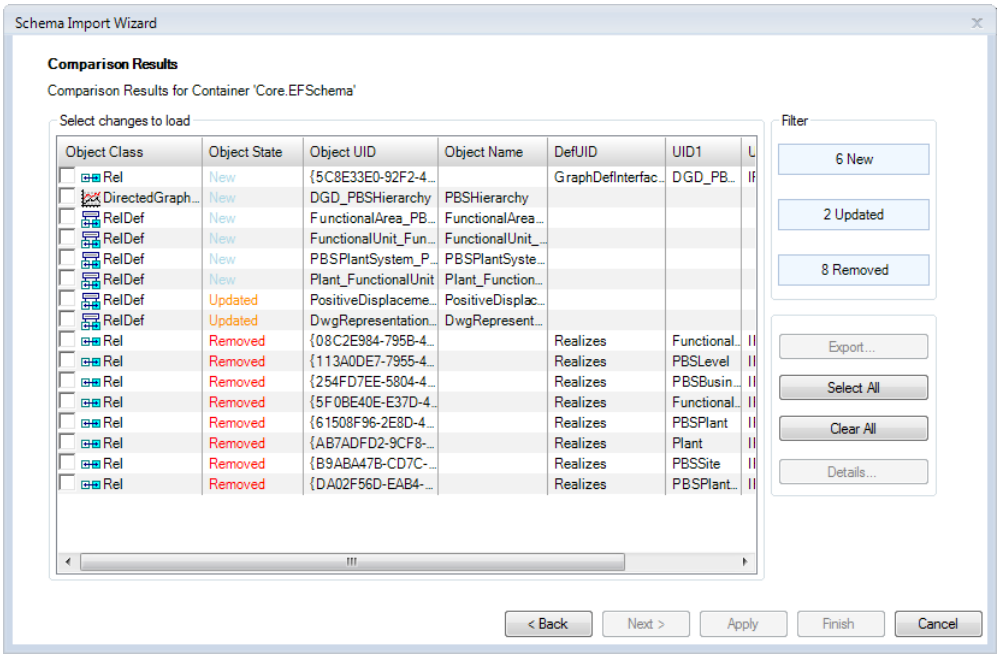
3. Under **Select external schema file**, browse to SCHEMA_EFSchema-Gen.xml in the directory where the software created it in the previous procedure, *Prepare the CMF Data to be Loaded into SmartPlant Foundation.*



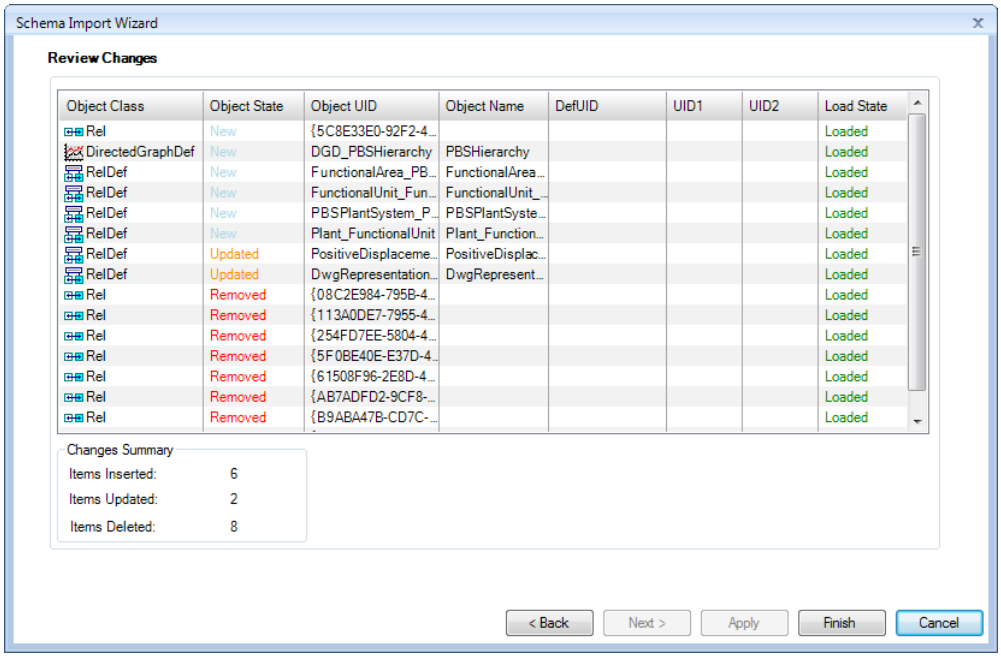
4. Click **Next**.

   The **Schema Import Wizard** compares the generated schema with the SmartPlant Foundation database and displays the results. This process can take several minutes.

5. Under **Comparison Results**, click **Select All**, and then click **Next**.



NOTE If the items were all successfully loaded, the status in the **Load State** column is set to **Loaded**. If not, the status is either **Passed** or **Failed**.



6. Click **Finish**.

Generate component schema files

1. In the Desktop Client, right-click the latest version of the CMF_ver document.
2. On the shortcut menu, click **Regenerate Schema Files**.

   NOTE This command creates a scheduler task that regenerates the component schema files based on the information in the current version of the CMF.
3. Verify the scheduler is started so that the component schema files can be generated.

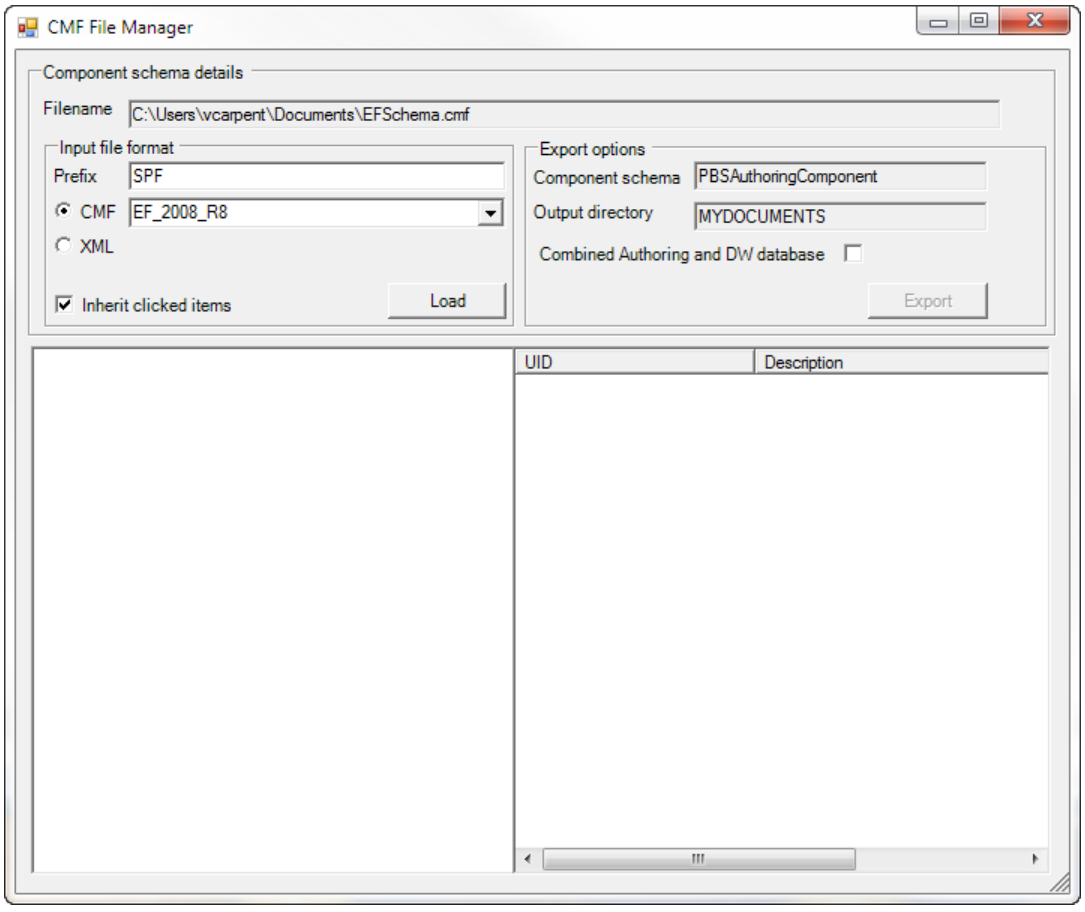
This command generates the SmartPlant component schemas in the location defined for your site in Server Manager.

NOTE This process can run in the background while you proceed with other operations; however, it must be completed before you attempt to publish or retrieve from an authoring tool.

Generate authoring schema and load files

This procedure generates an XML file that creates the custom PBS components in the authoring schema and a new PBS authoring tool map schema. In addition, this step creates new PBS methods, forms, and menu items.

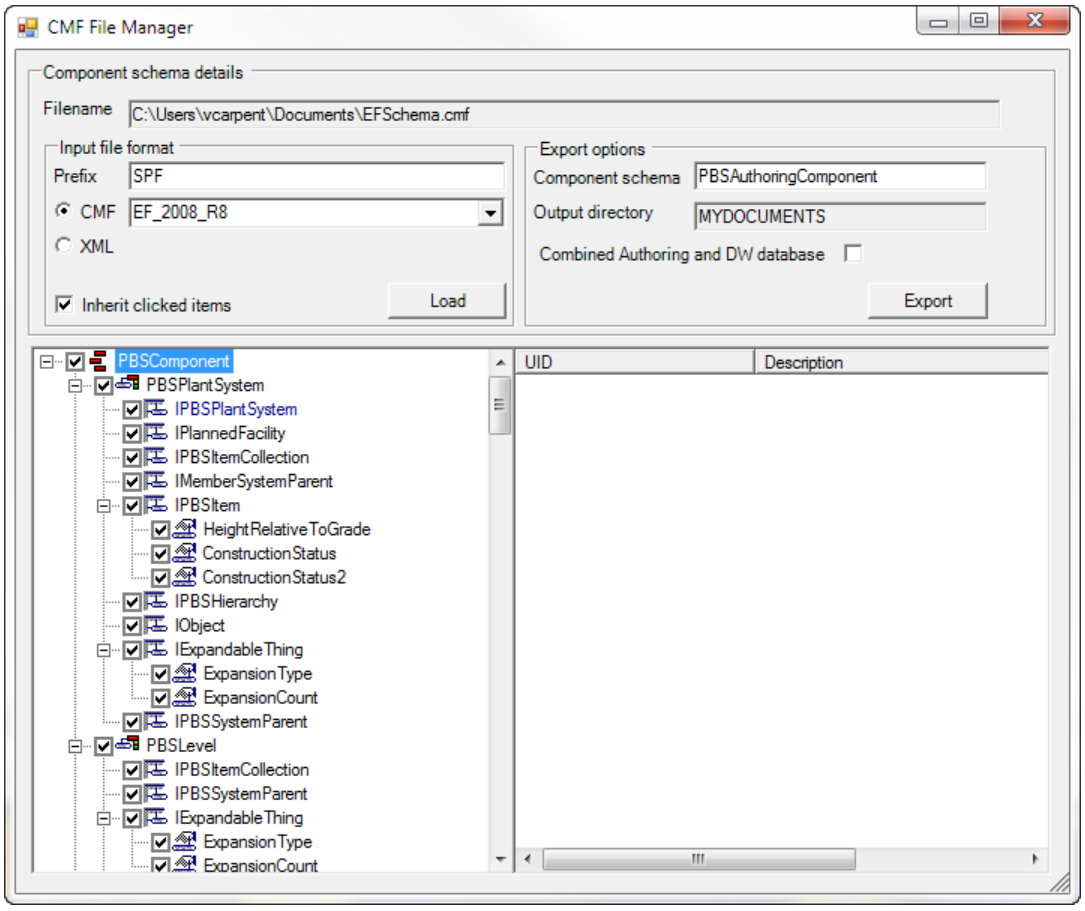
1. Find the CMF you modified in the Desktop Client (**Find > Integration > Schema Documents**).
2. Right-click the CMF file version (CMF_Ver), and click **Launch CMF File manager (PBS)**.



NOTE The component schema and prefix are populated from the associated method.

3. Select the **Combined Authoring and DW database** option.
4. Click **Load**.

NOTE One node should appear in the tree view.



5. Expand the **PBSComponent** node.
6. Make sure the **Inherit Clicked Item** option is selected.
7. Select **PBSComponent** to select all items.

   NOTE You can unselect items that are not needed for export, but always include **PBSComponent** and **PBSDocument**. For example, if you are implementing a new structure of plant/unit/system/subsystem, you should find and clear the check box beside FunctionalArea to ensure that menu items and forms for creating and managing areas are not added back to the Desktop Client user interface.
8. Click **Export**.
9. Close the **CMF File Manager** dialog box.

This operation generates the schema, some basic SmartPlant Foundation administration objects, and a tool map schema file with mappings to the SmartPlant schema. This information is contained within seven files in the output directory.

File Name	Description	Comments
SPF_01_AuthoringSchema.xml	SmartPlant Foundation PBS authoring schema	This contains the sub-schema for the new PBS hierarchy.
SPF_02_AuthAdminData.xml	New PBS Methods and Menus	This file holds the methods and menus configuration (used to display your new item in the authoring environment).
SPF_03_AuthFormInstruction.xml	PBS Form Instructions	The instructions to generate forms for newly generated objects.
SPF_04_DWAdminData.xml	Data Warehouse Methods and Menus	Optional - only required if you want to see the data as it is loaded in the data warehouse.
SPF_05_DWFormInstruction.xml	Data Warehouse Form Instructions	Optional - only required if you want to see the data as it is loaded in the data warehouse.
SPF_SelectedItems.xml	Copy of Selected Items	Do not load - contains the classes selected that form the new hierarchy. You can use this file to see what classes were selected when the export was done.
ToolMapSchema_SPF.xml	New PBS Tool Map Schema	Do not load - contains an updated version of the map filed used by the SmartPlant Foundation Desktop Client to publish the PBS document, based on the PBS schema information in the CMF file.

Load alternate PBS objects

Load the new schema and administration objects

1. In the Desktop Client, click **File > Load,** and select the following files, which were generated in the previous procedure:
   - **SPF_01_AuthoringSchema.xml**
   - **SPF_02_AdminData.xml**
   - **SPF_03_AuthFormInstruction.xml**
   - **SPF_04_DWAdminData.xml**
   - **SPF_05_DWFormInstruction.xml**
2. Click **Process** to load the changes into the SmartPlant Foundation database.

Update access group for the new PBS graph definition

The Schema Editor automatically generates a graph definition called DGD_PBSHeirarchy; the change will be loaded with the Schema Import Wizard.

A few updates need to be made to the PBSHierarchy graph def. These changes are made when you load the UpdatePBSHierarchyGraphDef.xmlldr file delivered in the <*SmartPlant Foundation Installation Location*>**\Samples\SampleData\AlternativePBS** folder. This file references the scripts that will make the necessary updates to the PBSHeirarchy graph def.

Use **File > Loader** to open and load this file. The scripts below are loaded and run in the correct order:

- 01-ADMIN_Update_PBSHierarchy.xml
- 02- SCHEMA_Update_PBSHierarchy.xml

Replace existing PBS map file

1. To replace the existing PBS map file, find the existing tool schema PBS tool map schema in SmartPlant Foundation by selecting **Find > Adapter Documents,** type **PBS Map Schema**, and click **OK**.
2. Check out the **PBS Map Schema** file.
3. Replace the checked out document (ToolMapSchema_PBS.xml) with the newly-generated map schema **ToolMapSchema_SPF.xml** (Rename and replace).
4. Check in the new **PBS Map Schema** file.
5. Log out of Desktop Client.

Create a new plant and structure

Create a plant

1. Log into the Desktop Client.
2. Click **File > New > Organization / Support Items > Plant**.
3. Type the name of your plant based on the guidelines in Plant Naming Guidelines.
4. In the **Status** list, select **Active**.

Create a unit

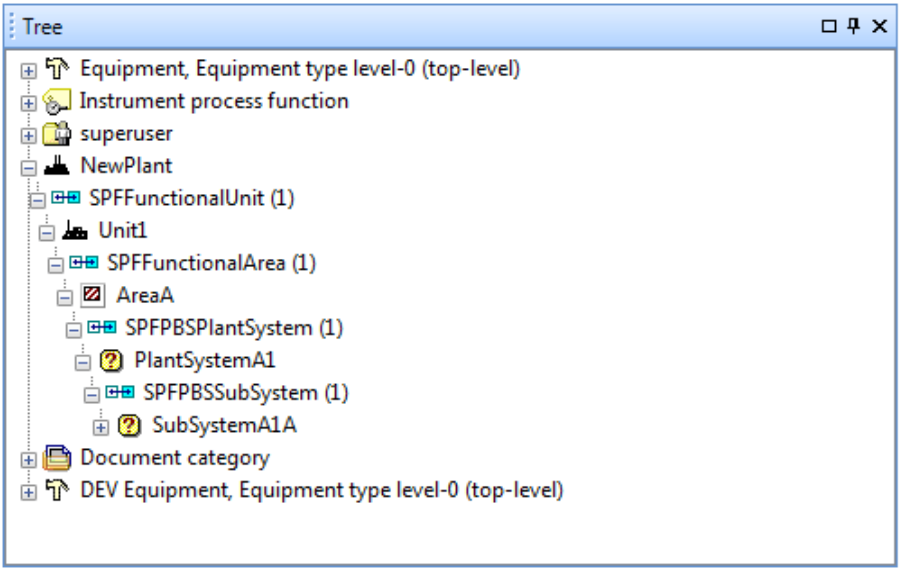
1. Click **File > New > Organization / Support Items > Unit**.
2. Type the name of your unit.
3. Select **New** for **Construction** status.

Create a system

1. Click **File > New > Organization / Support Items > System**.
2. Type the name of your system.
3. Select **New** for **Construction** status.

Create a sub system

1. Click **File > New > Organization / Support Items > Sub System**.
2. Type the name of your sub system
3. Select **New** for **Construction** status.
4. The customized PBS structure is created.



Associate a vault with the new plant

1. Click **Find > Administration > Vault** to find the vault that you want to associate with the new plant.
2. Drag the vault to the new plant in the tree view or the **New Items** window.

NOTE You must define role assignments for the new configuration.

Register and publish the PBS

Register the PBS

1. Click **SmartPlant > PBS > Register**.
2. In the **SmartPlant Foundation URL** box of the **Register** dialog box, type the node name and virtual directory of the SmartPlant Foundation database to which you want to register your plant. Use the following format: http://*[SPFServer]/[VirtualDirectory]*.

   TIPS

   - To register to a secure server, type https:// in your SmartPlant Foundation URL.
   - Replace *[SPFServer]* with the name of your SmartPlant Foundation Web server.
   - Replace *[Virtual Directory]* with the name of the virtual directory for the SmartPlant Foundation Web Client. By default, the virtual directory for the first instance of the Web Client that you install is SPFASP. However, if you install multiple instances of the Web Client to connect to multiple databases, the virtual directory name may be different.
3. Enter your SmartPlant Foundation user name and password in the **User name** and **Password** boxes.
4. From the **Plant name** list, select the plant with which you want to register your authoring tool.
5. Click **Finish**.

Publish the PBS

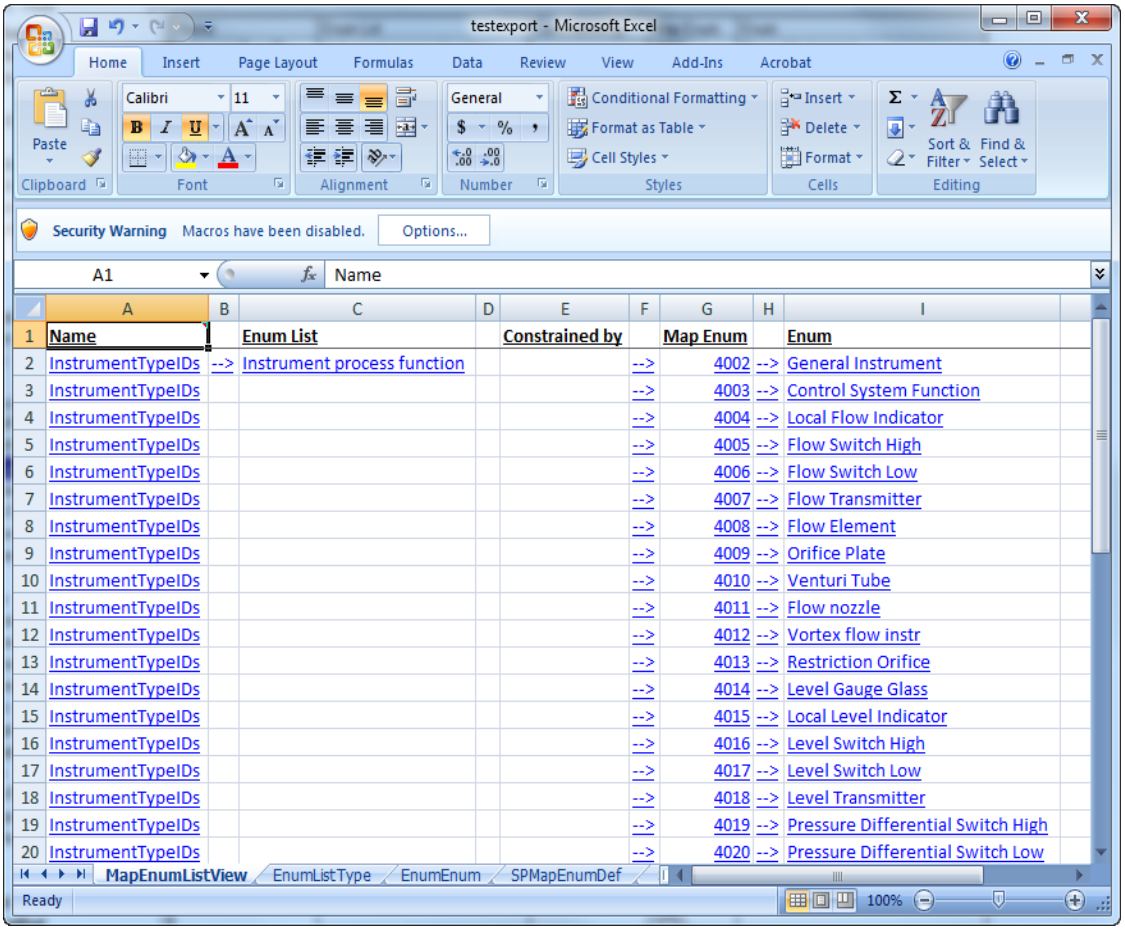
1. Right-click the plant in the Tree window, then click **Publish PBS**.
2. In the **Publish** dialog box, select a revision scheme and the first revision for the PBS document.
3. Click **OK** to publish the PBS.

NOTE The PBS document must be attached to a workflow that has a load step. The PBS document must be loaded successfully in SmartPlant Foundation for tools to retrieve it.

SECTION 10

Using Excel to Export and Import Schema

You can export schema data from Schema Editor to Microsoft Excel to make changes. You can import the edited schema from Excel to Schema Editor.



	A	B	C	D	E	F	G	H	I
1	Name		Enum List		Constrained by		Map Enum		Enum
2	InstrumentTypeIDs	-->	Instrument process function			-->	4002	-->	General Instrument
3	InstrumentTypeIDs					-->	4003	-->	Control System Function
4	InstrumentTypeIDs					-->	4004	-->	Local Flow Indicator
5	InstrumentTypeIDs					-->	4005	-->	Flow Switch High
6	InstrumentTypeIDs					-->	4006	-->	Flow Switch Low
7	InstrumentTypeIDs					-->	4007	-->	Flow Transmitter
8	InstrumentTypeIDs					-->	4008	-->	Flow Element
9	InstrumentTypeIDs					-->	4009	-->	Orifice Plate
10	InstrumentTypeIDs					-->	4010	-->	Venturi Tube
11	InstrumentTypeIDs					-->	4011	-->	Flow nozzle
12	InstrumentTypeIDs					-->	4012	-->	Vortex flow instr
13	InstrumentTypeIDs					-->	4013	-->	Restriction Orifice
14	InstrumentTypeIDs					-->	4014	-->	Level Gauge Glass
15	InstrumentTypeIDs					-->	4015	-->	Local Level Indicator
16	InstrumentTypeIDs					-->	4016	-->	Level Switch High
17	InstrumentTypeIDs					-->	4017	-->	Level Switch Low
18	InstrumentTypeIDs					-->	4018	-->	Level Transmitter
19	InstrumentTypeIDs					-->	4019	-->	Pressure Differential Switch High
20	InstrumentTypeIDs					-->	4020	-->	Pressure Differential Switch Low

The following data can be exported to Excel:

- Any schema displayed in a tabular view
- Pseudo file configuration (CMF)
- Schema container (XML file or CMF active configuration)

NOTE Schema Editor does not support exporting tool map schema to Microsoft Excel for modifications and importing back.

File Generated by Export Commands

For every container, pseudo file configuration, or view that is exported, two files will be generated:

- .xlsm file – Excel file containing the exported objects
- .UIDs file – Text file containing the UID for every object and relationship that is exported

Export the Schema to Excel

Export Schema Containers

You can export and import a schema container. Schema containers are large files, so the export and import processes will take some time.

Before exporting your schema container data to Excel, consider what changes you plan to make in Excel. When you make changes to the exported container in Excel, it could cancel/nullify/invalidate other changes being made to the schema (outside of the exported Excel file). Intergraph recommends exporting smaller schema files to Excel to edit.

To export a schema container the following process is used:

- Open the schema file
- Set one or more active configurations if the schema file is a CMF file
- Export the schema container to Microsoft Excel.

Export the Schema Container

1. Click **Tools > Schema > Export an Importable Excel Workbook for Container** to export the entire schema container.
2. Type the name of the exported Excel file in the **Save Spreadsheet File** dialog box, then click **Save** to export. The file is saved as a .xlm.

   NOTE Exporting the container can take a considerable amount of time. Please be patient. In addition, the file size of the exported container is large.

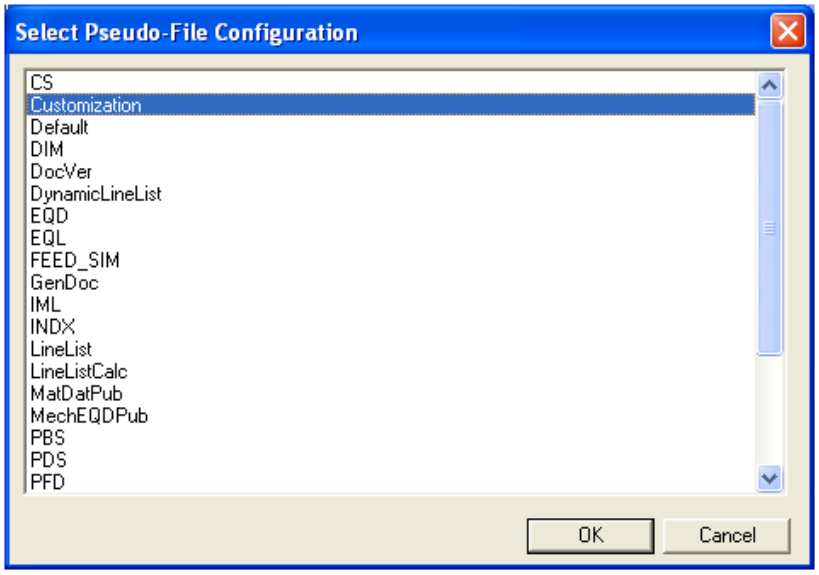
Export Pseudo File Configurations

You can export pseudo file configurations to Microsoft Excel for editing. When you export a pseudo file configuration, only objects within the pseudo file will be included in the generated Excel file.

Export the Pseudo File

1. Click **Tools > Schema > Export an Importable Excel Workbook for Pseudo-File Configuration** to export a pseudo file.

2. Select the pseudo file to export on the **Select Pseudo File Configuration** dialog box, and then click **OK**.



3. Type the name of the exported Excel file in the **Save Spreadsheet File** dialog box, then click **Save** to export. The file is saved as a .xlm.

Export a Displayed View

You can export any view that displays the schema in a tabular format, such as Table (hor), ClassView, and MapClassView.

This command is only available on the shortcut menu for the Schema Editor's view window when you:

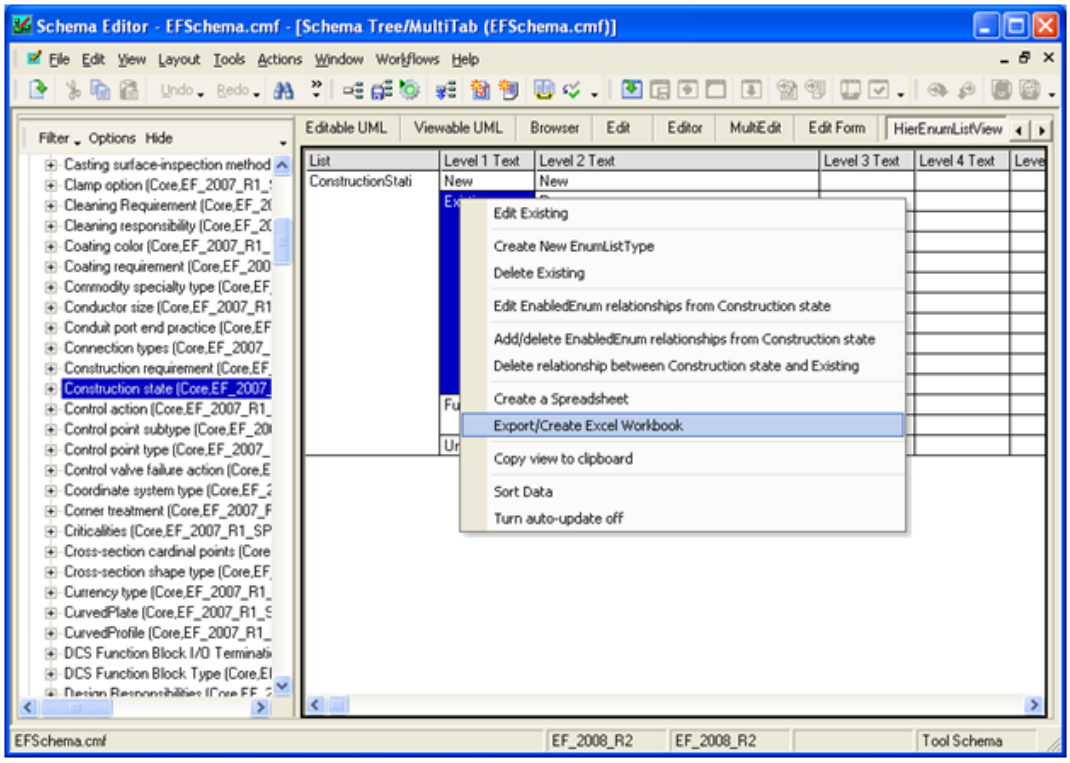
- Display the desired set of objects to be exported in a view window. See *Navigating Files and Data in the Schema Editor* (on page 101).
- Right-click inside the view window and select the export command

Export a View

1. To export a view, such as the construction status enumerated list, click the appropriate view tab, such as **HierEnumListView**.

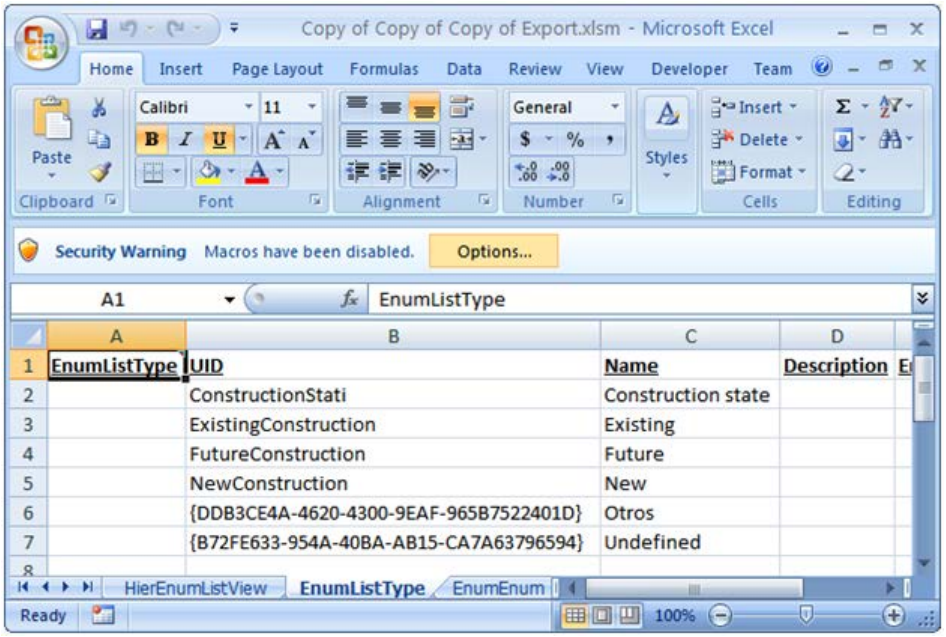
   NOTE The view tabs are available by setting **View Options**. For example, to set **HierEnumListView,** click the **MultiTab** tab, then click **View Definitions** on the **View Options** dialog box. See *Set View Options* (on page 123).

2. Right-click inside the view, and then click **Export an Importable Excel Workbook** to export the view.



3. Type the name of the exported Excel file in the **Save Spreadsheet File** dialog box, then click **Save** to export. The file is saved as a .xlm.

   The following is an example of an exported enumerated list.



Edit in Excel

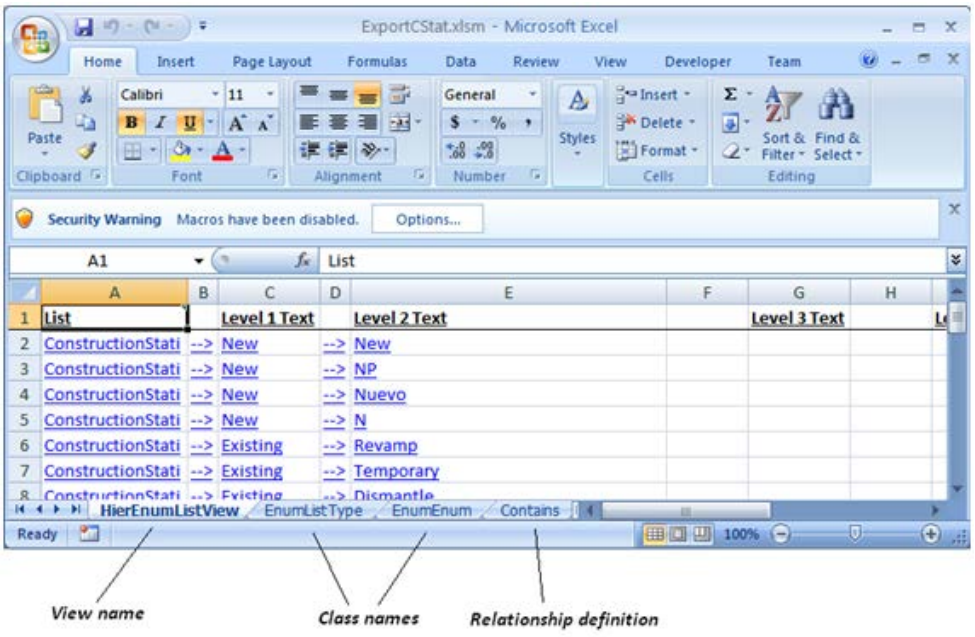
Excel (.xlsm) File

Worksheets in .xlsm File

For every class definition of an object that is exported, an Excel worksheet will be created and named the same as the class definition's UID. Because UID and Name are the same for class definitions, the worksheet's name is also the class definition's Name.

For relationship objects (class definition UID = Rel), a separate Excel worksheet will be created containing every relationship type (DefUID). The worksheet will be named based on the UID for the relationship definition for that relationship.

If a view is exported, the first worksheet in the Excel workbook will contain the view and will be named based on the exported view.

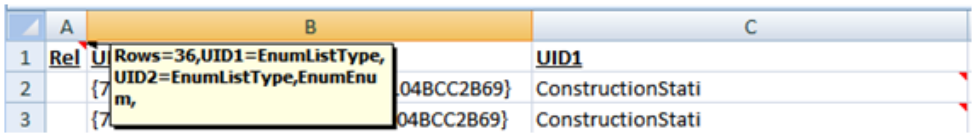


Rows in Excel Worksheet

The first cell on the first row contains the class definition for the objects displayed on that worksheet. For all worksheets except those containing relationships, the text in this cell matches the name for the Excel worksheet. This cell will contain a comment that includes the following:

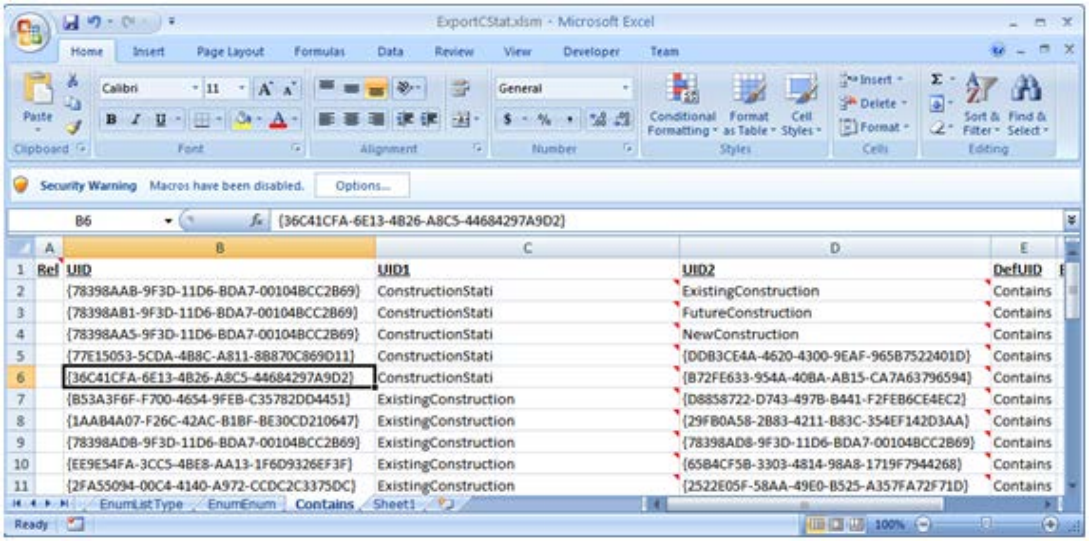
- Rows=# of exported rows (all worksheets except first worksheet for exported view)
- View=name of exported view (only first worksheet of exported view)
- UID1=comma-delimited set of class definitions (worksheet names) that contain legitimate objects for UID1 (only for relationship worksheets)

- UID2=comma-delimited set of class definitions (worksheet names) that contain legitimate objects for UID2 (only for relationship worksheets)



The rest of the first row of each Excel worksheet contains the names of each of the properties for the objects displayed on that worksheet.

For every object that is exported, a row in an Excel worksheet will be added. This row will contain all the property information for that exported object.

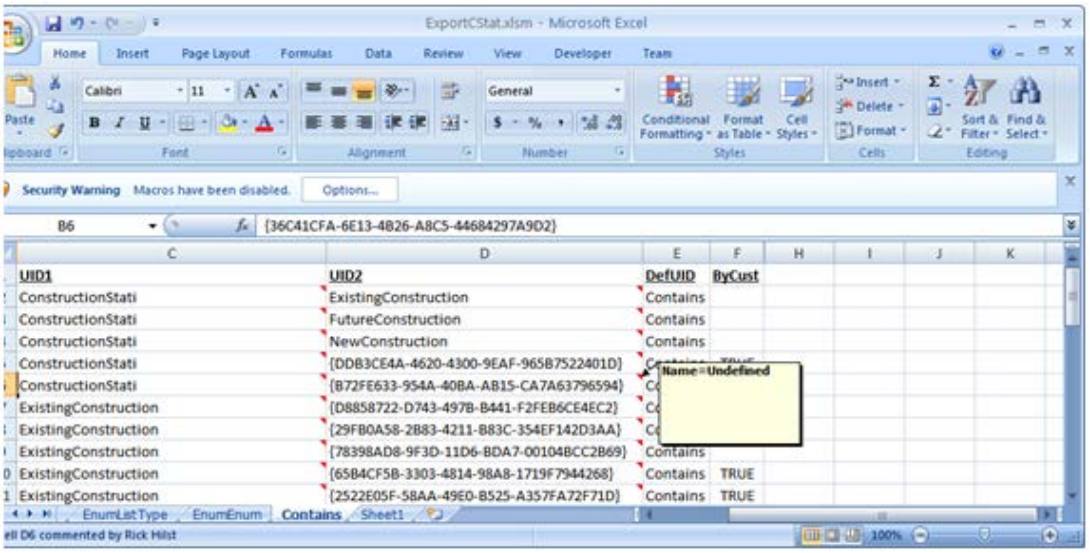


The number of objects initially displayed on a particular worksheet should match the number of rows in the Rows= comment in the upper left cell for that worksheet.

Cell Information

Each cell (other than those on the first row) should contain the value for the particular property (from the first row) for the exported object for that row.

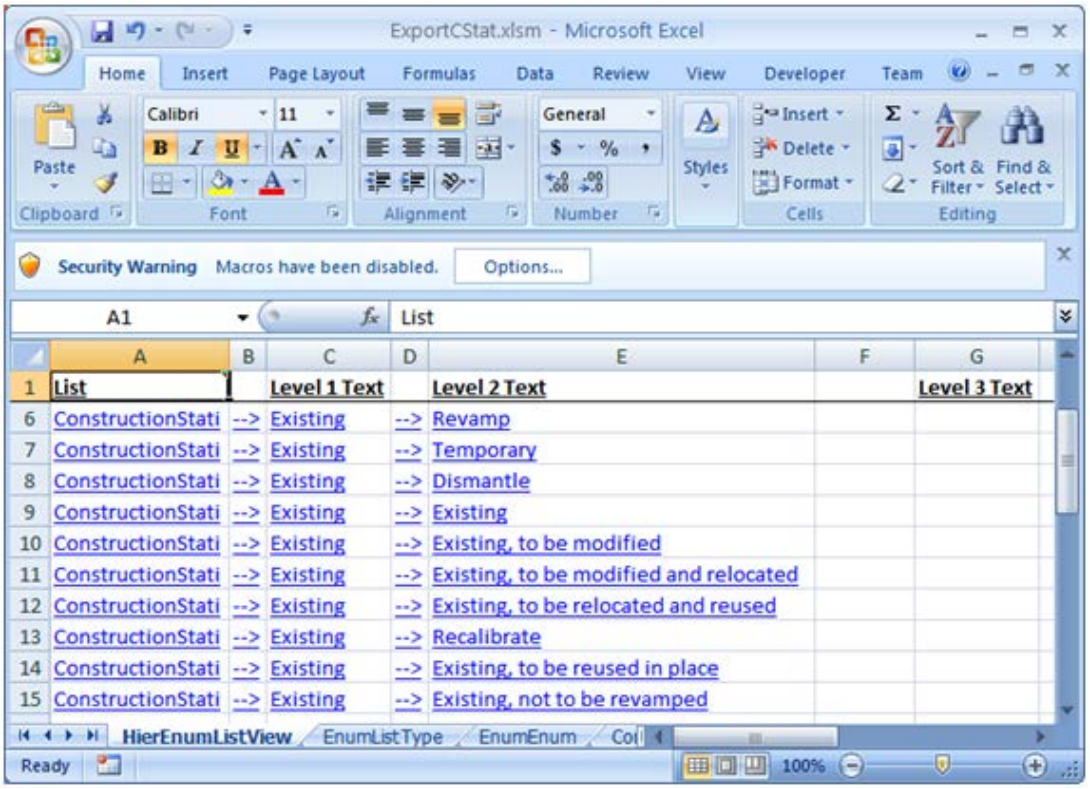
For relationship worksheets, the cells for the UID1 and UID2 columns contain a comment that identifies the name for the object corresponding to the UID in that cell.



UID1	UID2	DefUID	ByCust
ConstructionStati	ExistingConstruction	Contains	
ConstructionStati	FutureConstruction	Contains	
ConstructionStati	NewConstruction	Contains	
ConstructionStati	{DDB3CE4A-4620-4300-9EAF-965B7522401D}		
ConstructionStati	{B72FE633-954A-40BA-AB15-CA7A63796594}		
ExistingConstruction	{D8858722-D743-497B-B441-F2FEB6CE4EC2}		
ExistingConstruction	{29FB0A58-2B83-4211-B83C-354EF142D3AA}		
ExistingConstruction	{78398AD8-9F3D-11D6-BDA7-00104BCC2B69}	Contains	
ExistingConstruction	{65B4CF5B-3303-4814-98A8-1719F7944268}	Contains	TRUE
ExistingConstruction	{2522E05F-58AA-49E0-B525-A357FA72F71D}	Contains	TRUE

SchemaRevVer property values are exported along with all the other property information. However, since these values for existing objects should never be changed by the user and these values are automatically assigned to new objects, the column for this property is hidden.

The cells in the first worksheet for an exported view contain the values for that view. Cells that are shown as underlined blue text are hyperlinked to the corresponding objects in the workbook. Clicking a hyperlinked cell takes you to the corresponding object in the workbook. Cells on this workbook that contain arrows, when clicked, take you to the corresponding relationship in the workbook.



Special Behavior Supported While Editing the Excel Workbook

Each of the generating .xlsm files comes with some built-in macros that support special editing behavior. By default, Excel disables these macros.

1. Click **Options** on the **Security Warning** message to enable these macros In the Excel workbook.

2. Click **Enable this content** in the **Microsoft Office Security Options** dialog box, and then click **OK**.

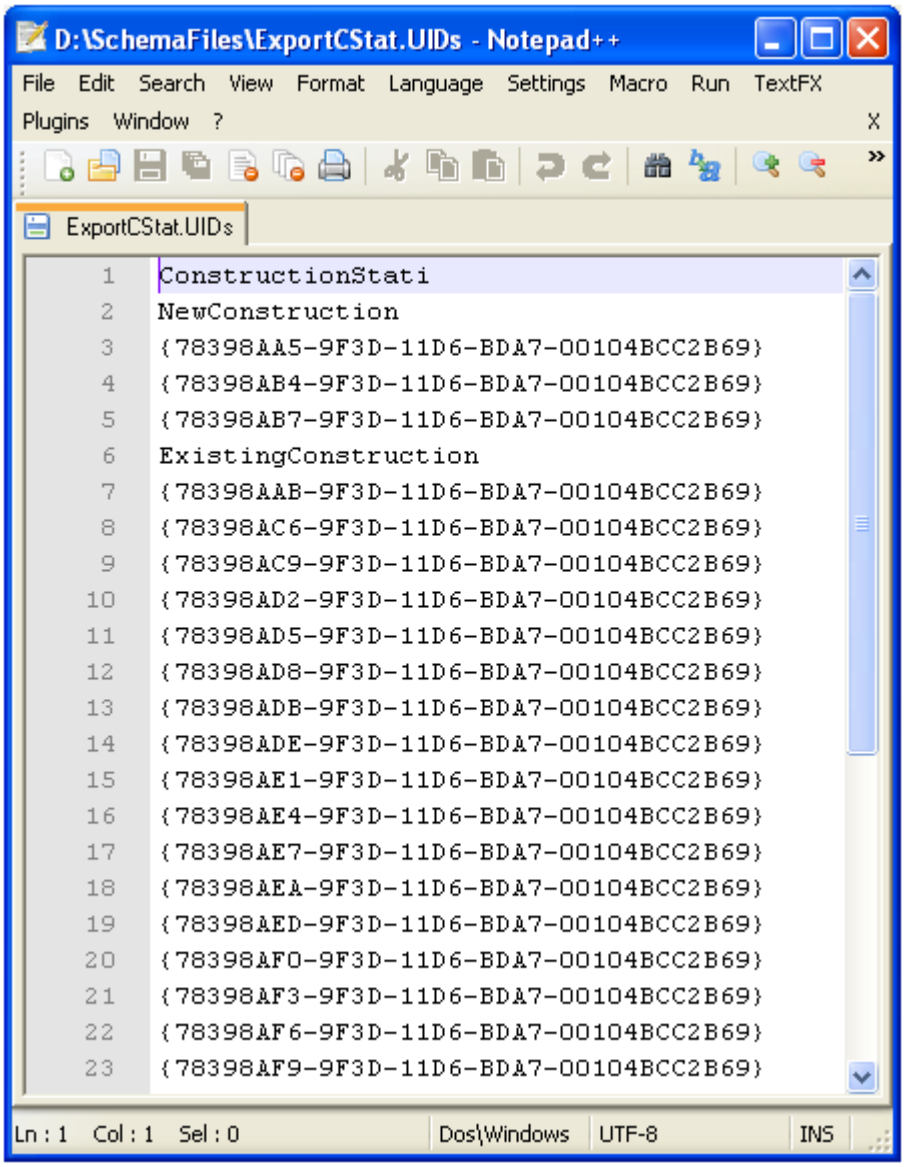


Once macros are enabled, the following behavior is supported:

- Clicking an empty UID cell or selecting a range of empty UID cells will automatically populate the selected cells with an automatically generated GUID
- Clicking an empty DefUID cell (or range of cells) will automatically populate the selected cells with the relationship type for the worksheet
- Clicking a cell for UID1 or UID2 will display a dialog box that contains a combo box of potential values (from the workbook) for that cell

UIDs File

In addition to the Excel file that is generated by the export process, a .UIDs file is created. This text file contains the UID for every object and relationship that is exported.



The .UIDs file is used during import to determine the following:

- New objects and relationships – objects in the .xlsm file that have a UID not in the .UIDs file
- Resurrected objects and relationships – objects in the .xlsm file and in the .UIDs file that don't exist within the schema container
- Deleted objects and relationships – Objects within the schema container and in the .UIDs file that no longer exist within the .xlsm file

Without the .UIDs file (or something equivalent), it would be impossible to detect whether an object in the .xlsm file is a new object versus a resurrected object. Likewise, without the .UIDs file, it would be impossible to determine objects that were deleted from the .xlsm file.

Import Data into Schema Editor

You can import your changes back into Schema Editor once you have completed your edits to the data.

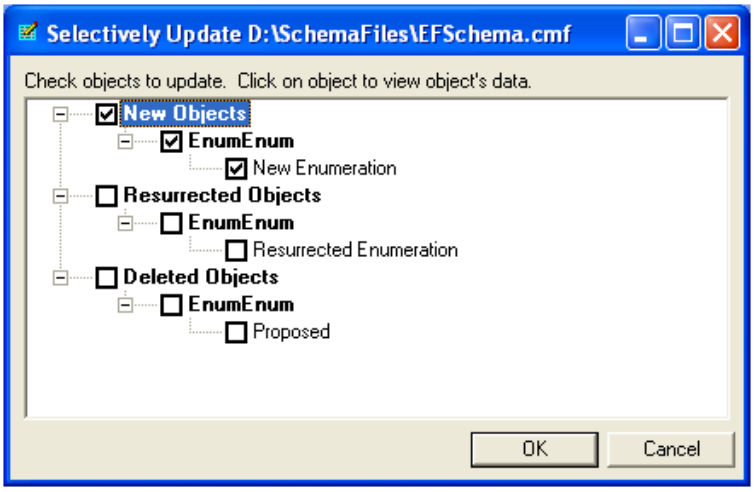
Import from Excel

1. Click **Tools > Schema > Import/Delete Excel Workbooks** to import the modified exported Excel files back into Schema Editor.
2. Select the file to import.

   The import process begins. The import process consists of the following steps:

   - Parsing the .UIDs file and populating a container will all the current objects and relationships corresponding to the UIDs in this file. All UIDs for which no current object or relationship is found are tracked and later used to differentiate between new objects and resurrected objects
   - The .xlsm file is parsed and objects are placed into a second container based on this parsing
   - A comparison is performed between the objects in the two containers
   - All objects in the second container (.xlsm file) not in the first container are validated. If any errors are detected during this validation, the validation correction dialog box is displayed and the user allowed the opportunity to correct these errors before proceeding. If the user fails to correct any error, then the import process is terminated.
   - A dialog box is displayed showing all the differences between the first and second containers. New objects are identified as either new or resurrected based on whether their UIDs exist in the .UIDs file.

- The user identifies the set of changes that are to be made by checking the appropriate check boxes on this dialog box. As a general rule, all new objects should be selected and all resurrected objects should not be selected. Normally deleted objects should be selected. Each modified object should be evaluated on a case by case basis to determine whether the changes should be selected.



- Once the user selects OK, the checked changes are then made to the current schema. After the changes are made the .UIDs and .xlsm files used for the import are deleted.

Parsing the .xlsm File - Other than the first worksheet for an exported view, all worksheets in the .xlsm file are parsed. If the workbook contains a Sheet1 worksheet, then this worksheet is ignored during parsing.

The upper left cell for the worksheet identifies the class definition for all objects for that worksheet.

The remaining cells on the first row identify the properties for the objects for that worksheet.

The other rows on that worksheet contain the property information for each object for the class definition identified by the upper left cell.

Parsing of object information is done on a row by row basis. If the UID cell for a row has a blank value, that row will be ignored during parsing (even if other property values are defined on that row). If the user deletes an object or relationship by deleting the corresponding row from the workbook, he should leave that row (with blank cells) in the workbook as a reminder of the deleted object (also to keep hyperlinks for exported views correct).

Validation Errors - Any new or resurrected objects encountered during the import process are automatically validated. If any errors (warnings don't count) are detected during this validation, the validation message correction dialog box will appear and the user can correct these errors (and warnings) via this dialog box.

If any errors are not corrected during this process, then the import process is terminated.

Selective Update Form - The dialog box that appears at the end of the update process and is used to identify the differences to apply is the same dialog box used for all selective update comparisons.

SECTION 11

Publish and Retrieve File Types for Tools

To allow users to define the document types that authoring tools publish and retrieve, the integrated SmartPlant Enterprise authoring tools store some of their authoring tool metadata in the SmartPlant schema instead of in the tool adapter. Metadata that can be defined in the SmartPlant schema includes:

- Publishable document types - Types of documents each authoring tool can publish.
- Retrievable document types - The types of documents each authoring tool can retrieve.
- Publishable document classes - The class definitions that describe published data for each publishable document type. Each publishable class definition must realize a primary interface definition that has a Componentization relationship with one of the publishing tool's component schemas.
- Retrievable document classes
- Tombstone document types - The type of document published by the tool to describe deleted objects in a previously published document
- Publishable component schema - A subset of the SmartPlant schema that describes the class definitions and realized interface definitions published by an authoring tool. Each authoring tool may have one or multiple component schemas. Each class definition typically belongs to one and only one component schema.

Defining this metadata externally provides users with the flexibility to add new document types for publish and retrieve without requiring the authoring tool adapter software to undergo any changes. You can modify the publish and retrieve document types and other metadata for an authoring tool in the Schema Editor. To be able to edit the SmartPlant schema in the Schema Editor, you must temporarily change the SmartPlant schema, which is marked read-only by default, to editable.

After you make the required changes to the tool metadata in the SmartPlant schema, you can use the **Generate Tool Metadatas** command to generate metadata XML files for each authoring tool that has metadata defined in the SmartPlant schema.

The SmartPlant software supports both internal (defined by the tool adapter API) and external definition of this type of tool metadata. The SmartPlant software looks for the external tool metadata schema file on the integration server in the folder where EFSchema.xml is located first. The format for tool metadata schema file is <ToolID>_meta.xml, where ToolID is the authoring tool's ID that matches the tool ID passed in the Connect and Register calls of IEFCommonUI interfaces by the authoring tools. This file format is created by default when you use the **Generate Tool Metadatas** command in the Schema Editor.

If no tool metadata file is available, the SmartPlant software calls the authoring tool adapter API to get the internal tool metadata.

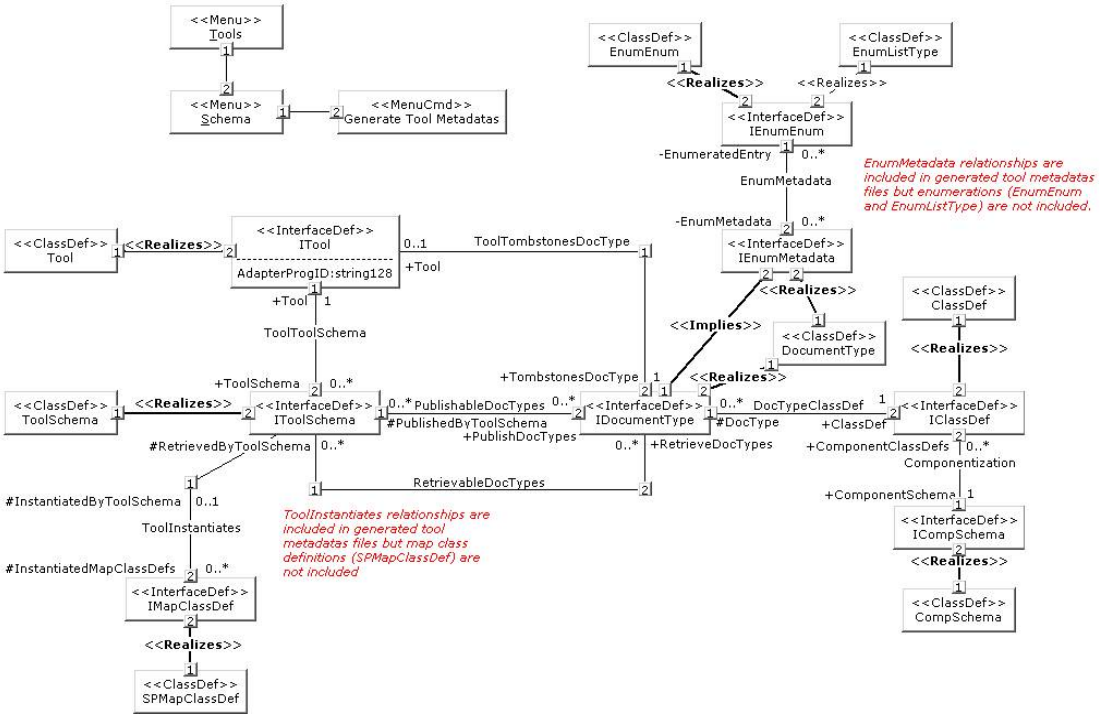
See Also

Create a New Document Type that Existing Tools Can Publish and Retrieve (on page 251)
Update Document Types that Existing Tools Can Publish and Retrieve (on page 255)
Externalize Authoring Tool Metadata (on page 240)
Generate Authoring Tool Metadata Files (on page 257)

Externalize Authoring Tool Metadata

To make integration more open and to support plug-and-play functionality, the set of document types that an authoring tool publishes and the set of document types that an authoring tool retrieves can be defined outside the tool adapter. Defining this metadata externally allows users to define new document types that the tool can publish and/or retrieve without triggering the need for software changes as long as the tool adapter already supports publishing or retrieving the types of data in the document types.

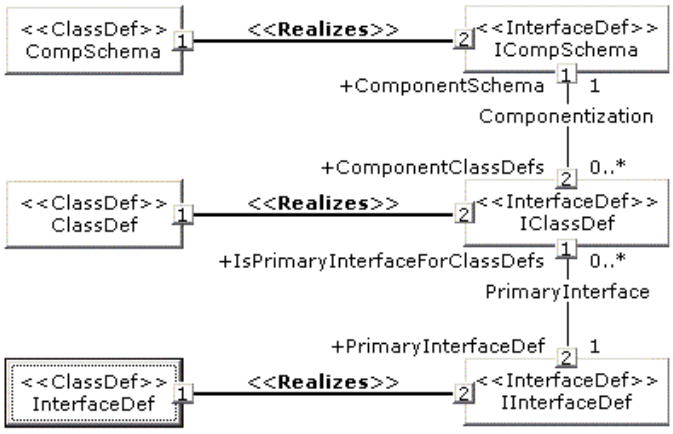
The following UML diagram includes the types of schema objects involved in externalizing this kind of tool metadata. The sections that follow describe these objects.



Class Definitions (ClassDef)

A class definition in the SmartPlant schema is the realization of a primary interface definition within a component schema.

Most applications use concrete class definitions to denote the existence of objects and use interface definitions (or abstract class definitions), if they are used at all, to identify the roles for the objects. The SmartPlant schema is based on a different approach where class definitions indicate that a primary interface definition (which identifies the existence of an object) is used by an authoring tool component (by a component schema).



As part of defining how the primary interface definition is used within a component schema, the class definition identifies the roles supported by a component schema for that object. The identification of the roles is done by the class definition realizing a subset of the interface definitions implied (directly or indirectly) by the primary interface definition. This set of realized interface definitions/roles defines the extent to which that object (primary interface definition) is supported by that software component (component schema).

The same primary interface definition, when used in a different component schema, can have a different class definition that realizes it.

Component Schemas (CompSchema)

A component schema is the subset of the overall SmartPlant schema that is supported by an application component (that is, an application or authoring tool that integrates with SmartPlant). An application that integrates with the framework may choose to have a single component schema that covers the entire set of data that application publishes or it may choose to break the data up into multiple component schemas.

The choice of a single component schema versus multiple component schemas is an application-specific decision. The factors that might lead to one decision versus the other are:

- Whether the data has any seams (if no seams, then probably only one component schema)
- Performance (smaller component schemas should result in better performance)
- Manageability (smaller component schemas are individually more manageable – however, redundancy across component schemas may be management problem)
- Documents produced (since data is published with documents)

After you make the required changes to the SmartPlant schema, you can use the **Generate Component Schemas** command to generate XML files of all component schemas, or use **Generate Component Schema** command to generate an XML file of a single component schema. When you select **Tools > Schema > Generate Component Schemas**, Schema Editor automatically begins generating all component schemas. To generate a single component schema, select **Tools > Schema > Generate Component Schema**, and then select the component schema you want to generate.

DocumentType Class Definition

The class definition associated with a publishable or retrievable document type.

EnumEnum Class Definition

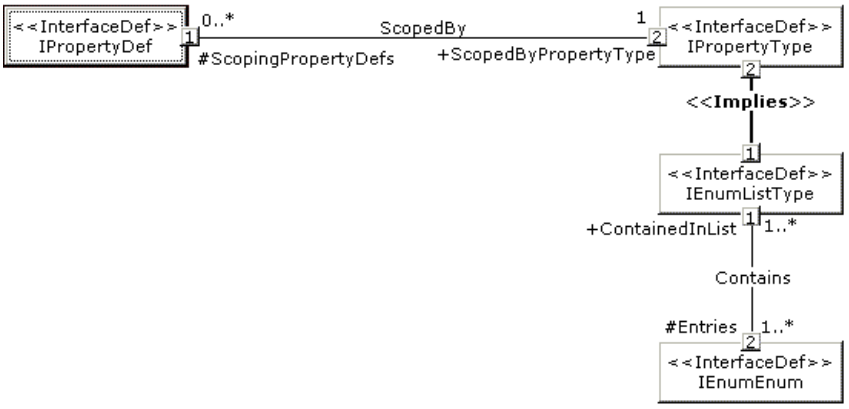
Enumerated entry. Enumerated entries must be contained within an enumerated list and are always leaf nodes (do not contain other enumerations).

Enumerated Lists (EnumListType)

An enumerated list is a predefined list of possible values for a property. A simple list might contain a couple values (like Yes and No) whereas a complex list might have many values and may be part of a multi-level enumeration hierarchy.

Enumerated lists are property types, which means they can be used to scope property definitions. When an enumerated list scopes a property definition, that enumerated list defines the set of acceptable values for instances of that property definition (that is, for properties).

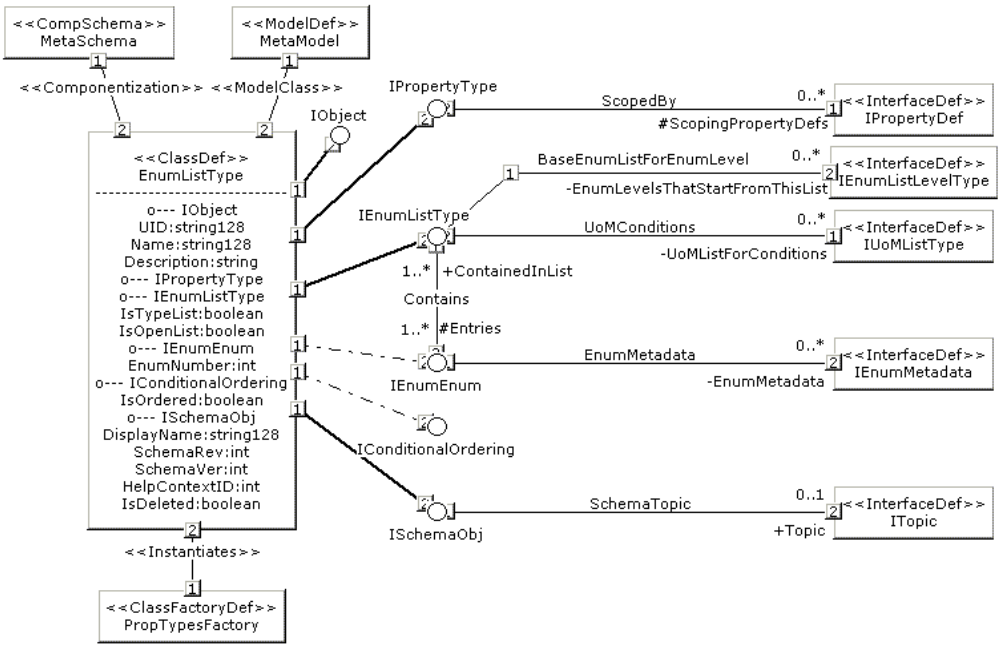
In the Schema Component, enumerated lists are instances of the EnumListType class definition and realize interface definition IEnumListType. Each enumerated list contains one or more enumerations (objects that realize IEnumEnum).



The enumerations for an enumerated list may be either enumerated entries (EnumEnum objects) or other enumerated lists. If enumerated lists are contained within other enumerated lists, the result is an enumeration hierarchy where the enumerated lists are branches in the hierarchy and the enumerated entries are leaf nodes within the hierarchy.

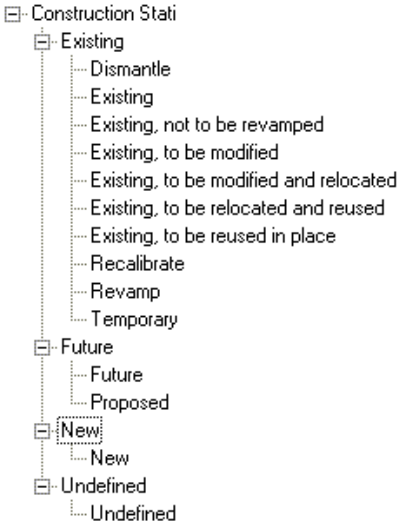
Since an enumerated list may be contained within another enumerated list, the EnumListType class definition not only realizes IEnumListType but also IEnumEnum.

The EnumListType class definition is shown in the diagram below.



Enumeration Hierarchies

When enumerated lists are contained within other enumerated lists, the result is an enumeration hierarchy. A typical enumeration hierarchy is show below.



Enumerated lists that are not part of enumeration hierarchies are considered to be single-level lists. An enumeration hierarchy is considered to be a multi-level list, where the number of levels is defined by the number of enumerated lists in the longest path through the hierarchy. For ConstructionStati, shown in the screen shot above, the number of levels is two (ConstructionStati and either Existing, Future, New or Undecided).

While most enumeration hierarchies in the SmartPlant schema are two or three levels deep, they can go much deeper. The equipment types hierarchy, for example, has seven levels.

The reasons for multi-level enumerated lists are:

1. To account for common semantics associated with branch nodes in the hierarchy
2. To allow for different tools having different levels at which they are able to specify the enumeration hierarchy.

The common semantics are illustrated by the Existing (branch) node in the ConstructionStati enumerated lists. All of the leaf nodes that branch from Existing are simply refinements of the semantics associated with Existing.

One tool may have only a semantic understanding of construction status at the “existing”, “new”, “future” level where another tool may differentiate between the different types of “existing.”

SPMapClassDef

Tool Class Definition

For every tool that integrates with SmartPlant, there should be one Tool object (an instance of the Tool class definition). This object and related objects are used to define the information specific to that tool, including the progID for the tool's adapter, the tool (map) schemas for that tool and the set of document types published and retrieved by that tool.

ToolSchema Class Definition

Every tool (map) schema file used to define mapping from the tool's schema to the SmartPlant schema and/or mapping from the SmartPlant schema to the tool's schema should be defined using an instance of the ToolSchema class definition.

Each ToolSchema object is related to the Tool object for its tool, to the DocumentType objects that are published and/or retrieved using the tool schema, and to each of the map class definitions (SPMapClassDef) for that tool schema.

InterfaceDef Instances

IClassDef Interface Definition

Primary interface definition for class definitions.

ICompSchema Interface Definition

Primary interface definition for component schemas

IDocumentType Interface Definition

For every class definition that corresponds to a type of document (that is published, retrieved and/or used for passing tombstones), there should be at least one DocumentType object. This DocumentType object should realize IDocumentType.

The IDocumentType interface definition is used to relate the document type to the tool schemas that are used for publishing and/or retrieving documents for this document type. While a document type may be related to multiple tool schemas for publishing and/or retrieving, it should be related to no more than one tool schema for a particular tool.

IEnumEnum Interface Definition

Primary interface for enumerated entries. Also realized by enumerated lists. Exposes properties and behavior of enumerations contained within an enumerated list.

IEnumMetadata Interface Definition

Class definitions that can be used to extend the definition of an enumerated entry or enumerated list should realize IEnumMetadata. This interface definition is used to relate the class definition to the enumeration by way of an EnumMetadata relationship.

IMapClassDef Interface Definition

Primary interface definition for tool’s (map) class and interface definitions.

ITool Interface Definition

For every tool that integrates with SmartPlant, there should be a Tool object that realizes the ITool interface definition.

This interface definition is used to:

1. Define the progID for the tool's adapter
2. Identify the tool schemas for that tool
3. Identify the tombstones document type

AdapterProgID Property Definition

The AdapterProgID property definition is used to identify the progID (for example, "MyApplication.MyAdapter") of the adapter for the Tool object. The software identified by this progID should support the required adapter interfaces (described elsewhere).

IToolSchema Interface Definition

For every tool schema that a particular tool supports, there should be one ToolSchema object. Each of these ToolSchema objects should realize IToolSchema.

The IToolSchema interface definition has relationships that identify:

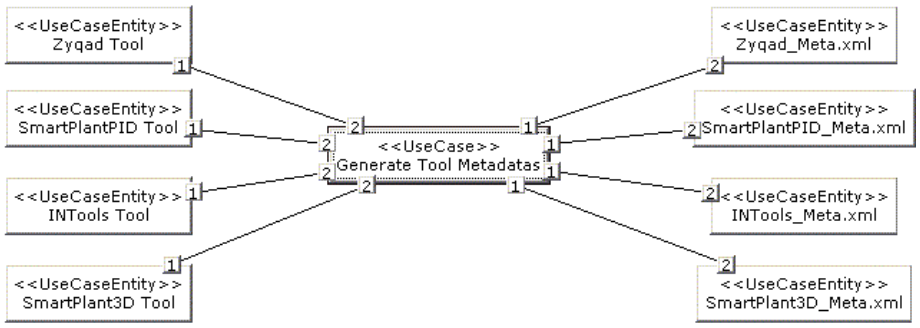
1. The tool for the tool schema (ToolToolSchema)
2. The publishable document types that use the tool schema (PublishableDocTypes)
3. The retrievable document types that use the tool schema (RetrievableDocTypes)
4. The map class definitions for the tool schema (ToolInstantiates)

MenuCmd Instances

Generate Tool MetaDatas Command

The **Generate Tool MetaDatas** command is used to generate the metadata file for each Tool defined in the SmartPlant Schema. For each Tool defined, a file called <Tool>_Meta.xml is created (for example, SmartPlantPID_Meta.xml).

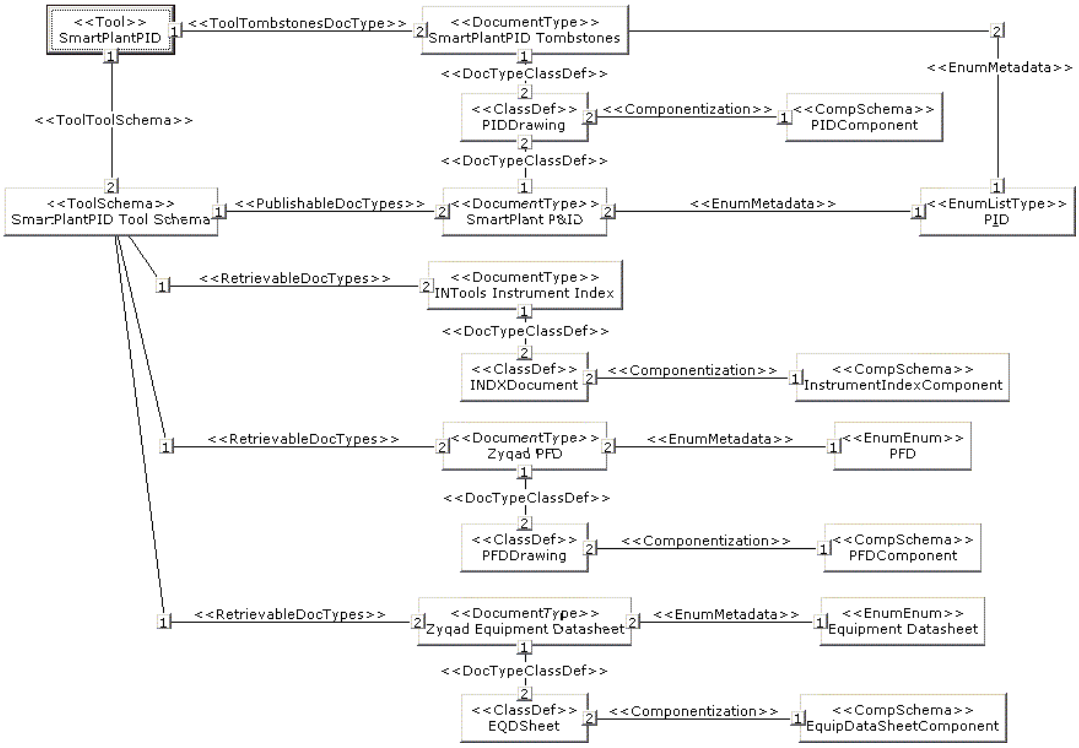
UML Diagram for Generate Tool Metadatas



This file contains the <Tool> object as well as the <DocumentType> objects for the tool (one <DocumentType> object for each document type published and/or retrieved by the tool) and the <ToolSchema> objects for that tool. In addition, the <ClassDef> object for each document type and the <CompSchema> objects for those class definitions are also loaded into the generated file.

All of the relationships between these objects are also loaded into the generated XML file. However, although the EnumMetadata relationships between the <DocumentType> objects and the corresponding SmartPlant enumerations (<EnumEnum> or <EnumListType>) are included, the corresponding enumerations are not included. Likewise, the ToolInstantiates relationships between the <ToolSchema> objects and the <SPMapClassDef> objects are included, but the corresponding <SPMapClassDef> objects are not included.

Tool Metadata for SmartPlantPID



Because the generated tool metadata file contains <ClassDef> and <CompSchema> objects, the tool metadata must be loaded into a container in a unique container composition and this container composition must be parented by the meta-schema container composition and not the schema container composition (in order to avoid UID conflict on the ClassDef and CompSchema objects - since those objects are loaded as part of the component schema).

Any IObject pointers to the <ClassDef> or <CompSchema> objects in the tool metadata container should not be passed to outside applications since these pointers are not the same as the corresponding objects in the schema container. For example, the <ClassDef> for PIDDrawing from the loaded SmartPlantPID_Meta.xml file will not have the same IObject pointer as the <ClassDef> for PIDDrawing from PIDComponent.xml.

The <ClassDef> and <CompSchema> objects are included in the Tool metadata file so that the SmartPlant Client can determine the component schema to load for a particular document type. The <EnumEnum>, <EnumListType> and <SPMapClassDef> objects are not included since the resolution of the relationships to these objects can be done after the component schema and tool schema have been loaded.

Menu Instances

Tools > Schema Menu

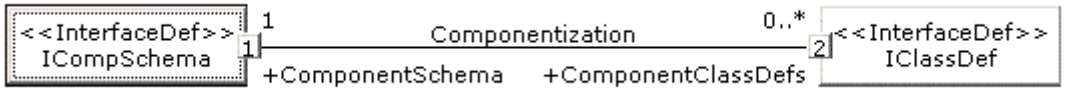
The **Tools > Schema** menu exposes the tool commands that apply to the Schema objects.

Tools Menu

The **Tools** menu exposes the various tool commands for each type of data supported by the Schema Editor (that is, MetaSchema, Schema, and so forth).

RelDef Instances

Componentization Relationship Definition



The Componentization relationship definition is used to tie a class definition (realizes IClassDef) to its component schema (realizes ICompSchema). Every class definition belongs to one and only one component schema. A component schema can contain an arbitrary number of class definitions.

Since a class definition can belong to only one component schema, the existence of a class definition depends on the component schema. If a new component schema is introduced, then every class definition for that component schema must be new and unique to that component schema.

Every class definition has one and only one primary interface (an interface definition, on the other hand, can be the primary interface for multiple class definitions). The many-to-one relationship definition between a class definition and a component schema and between a class definition and a component schema means that the class definition can be considered to be an intersection object between a primary interface and a component schema.

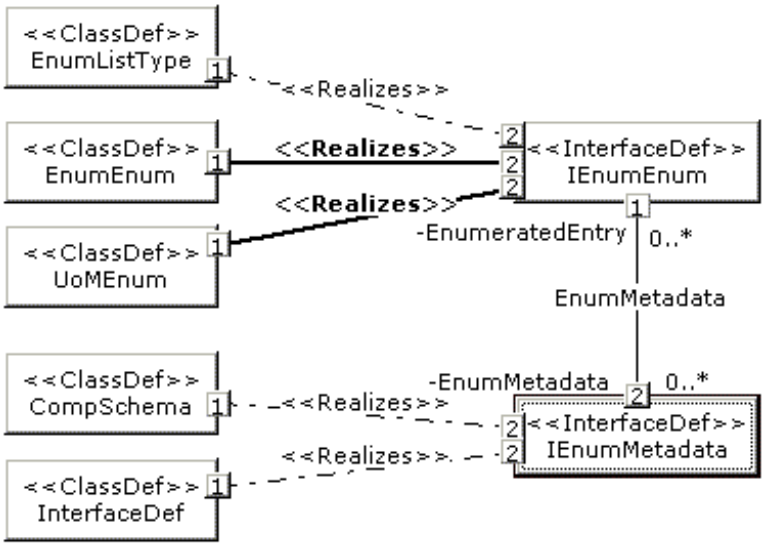
Although the cardinalities on the individual relationship definitions would allow an interface definition to be a primary interface for more than one class definition for a component schema, this is not allowed. Therefore, there can be only one class definition for a component schema that uses a particular interface definition as its primary interface.

DocTypeClassDef Relationship Definition

The DocTypeClassDef relationship definition relates a document type to the class definition used for documents of that document type. Each document type should be related to one and only one class definition and a particular class definition should not be related to more than one document type.

Document types are related to their enumerated entries using EnumMetadata relationships. A particular enumerated entry for a type of document can be related to multiple document types (for example, a P&ID document type enumeration can be related to multiple DocumentType objects). Therefore, the existence of a DocumentType object is driven by the existence of a class definition that is published/retrieved (that is, a DocTypeClassDef relationship), not by the existence of a document type enumerated entry.

EnumMetadata Relationship Definition



Enumerations (objects that realize IEnumEnum) can be described by other schema objects that realize IEnumMetadata.

An enumeration may be multiply-described by other schema objects or not described at all. Likewise, other schema objects may describe multiple enumerations or no enumerations.

The EnumMetadata relationship is used to extend the definition of an enumeration beyond the limited set of properties associated with the various enumeration property types. For example, this relationship, when it ends at an interface definition, defines the semantics that apply to the enumeration (for example, the enumeration Pump relates to IPump, which defines the property definitions and relationship definitions that make up a pump).

PublishableDocTypes Relationship Definition

The PublishableDocTypes relationship definition relates a tool schema to the document types that are published using that tool schema. A given tool schema can be used for publishing any number of document types, and documents of a given document type may be published using any of a number of different tool schemas (that is, the relationship definition is many-to-many).

While a document type may be related to multiple tool schemas for publishing, it should be related to no more than one tool schema for a particular tool.

RetrievableDocTypes Relationship Definition

The RetrievableDocTypes relationship definition relates a tool schema to the document types that are retrievable using that tool schema. A given tool schema can be used for retrieving any number of document types and documents of a given document type may be retrieved using any of a number of different tool schemas (that is, the relationship definition is many-to-many).

While a document type may be related to multiple tool schemas for retrieval, it should be related to no more than one tool schema for a particular tool.

ToolInstantiates Relationship Definition

The ToolInstantiates relationship definition is used to tie the map class definitions (SPMapClassDef) to their tool schema (ToolSchema). Every map class definition should belong to one and only one tool schema. A tool schema can be related to multiple map class definitions.

ToolTombstonesDocType Relationship Definition

When the command is executed that checks whether objects previously identified by a given tool as moved have been subsequently deleted, all objects and relationships that no longer exist result in tombstone objects being created. These tombstone objects are placed in a document of the type identified by the ToolTombstonesDocType relationship for that tool.

Every tool should have one and only one document type that is used for these moved and subsequently deleted objects and relationships. Likewise, a document type identified as being one tool's tombstone document type should not be used as another tool's tombstone document type (that is, each tool should have its own unique tombstone document type object).

Since multiple document type objects can be related to the same enumeration (by EnumMetadata relationships), multiple document type objects used as tombstone document types can be related to the same enumeration.

ToolToolSchema Relationship Definition

The ToolToolSchema relationship definition relates a tool to each of the tool schemas that it supports. A given tool schema belongs to one and only one tool, whereas a tool can support any number of tool schemas.

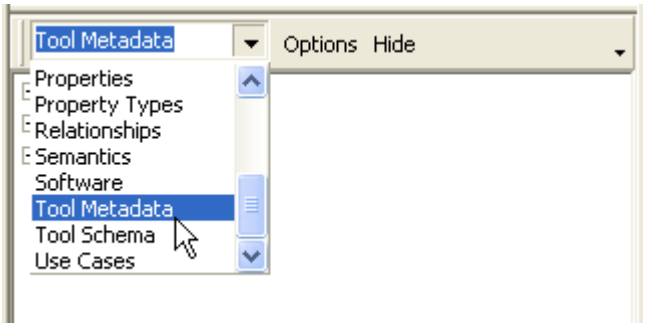
See Also

Publish and Retrieve File Types for Tools (on page 239)
Create a New Document Type that Existing Tools Can Publish and Retrieve (on page 251)
Update Document Types that Existing Tools Can Publish and Retrieve (on page 255)
Generate Authoring Tool Metadata Files (on page 257)

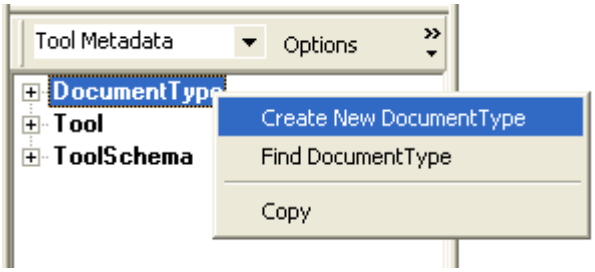
Create a New Document Type that Existing Tools Can Publish and Retrieve

1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**.

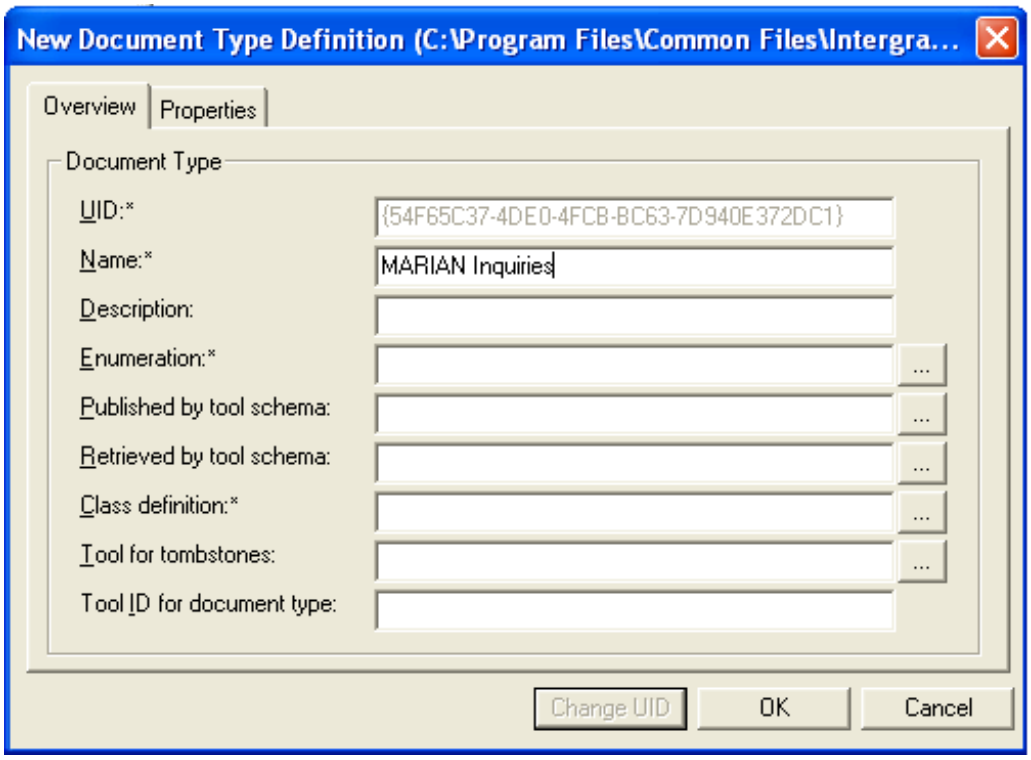
   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Click **Editor** in the **Form** list instead.
4. Click **OK**.
5. In the bottom tree view, click **Tool Metadata** in the **Select other classe**s list box to display SmartPlant schema objects for tool metadata, including tool objects, tool schema objects, and document types.



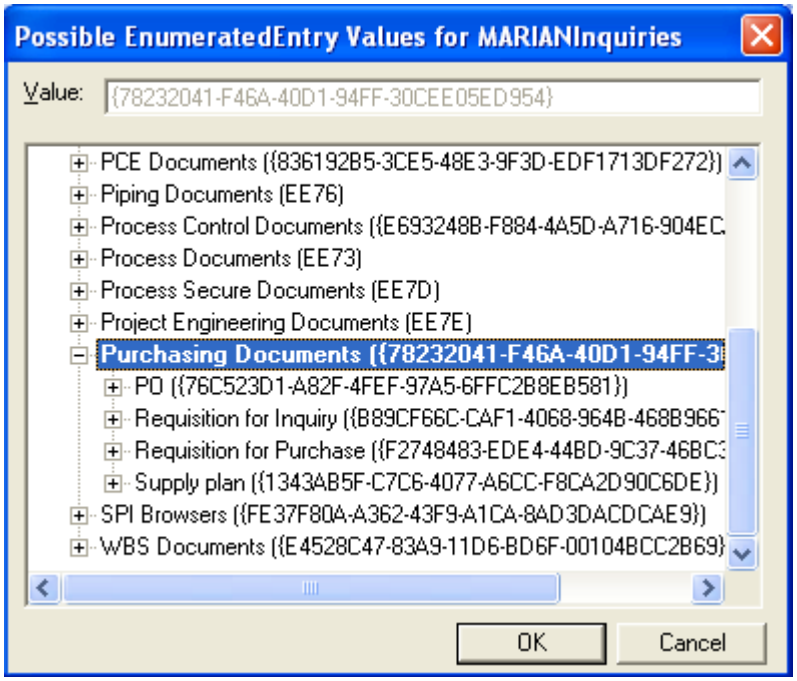
6. In the bottom tree view, right-click **DocumentType**, and then click **Create New DocumentType**.



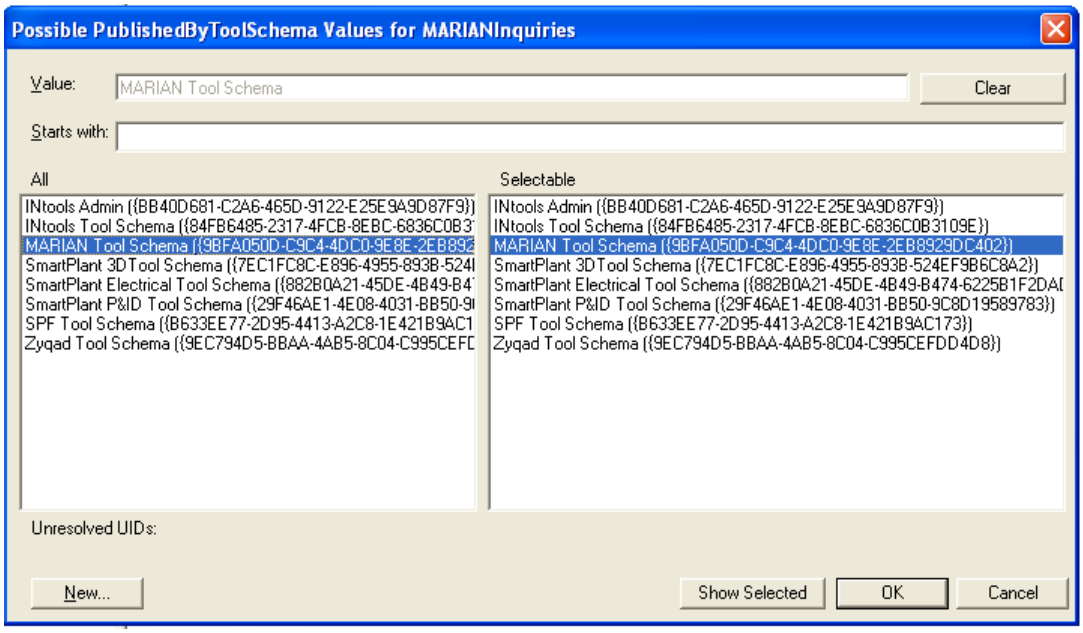
7. In the **New Document Type Definition** dialog box, type a name and description for the new document type.



8. Beside the **Enumeration** box, click ... and select the enumerated list for the new document type. Because the document type list is a hierarchical enumerated list, you can expand existing nodes to select a child list for the new document type.

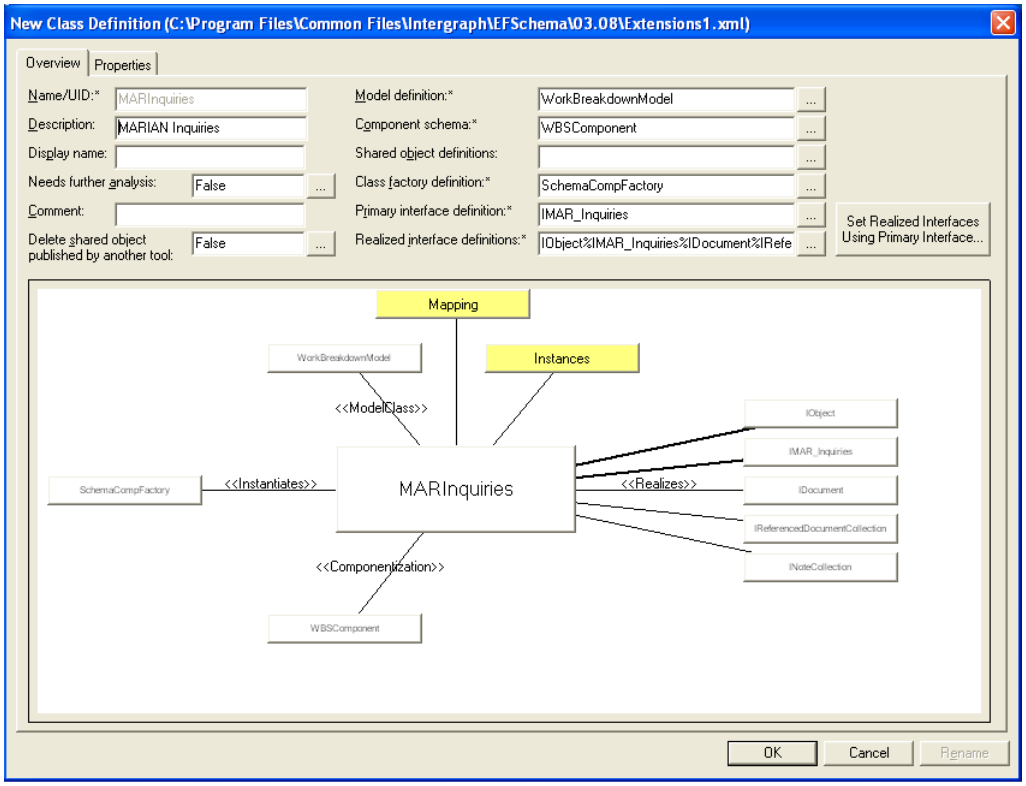


9. To define authoring tools that can publish the new document type, click ... beside the **Published by tool schema** box, and select the tool schemas for the publishing tool or tools.

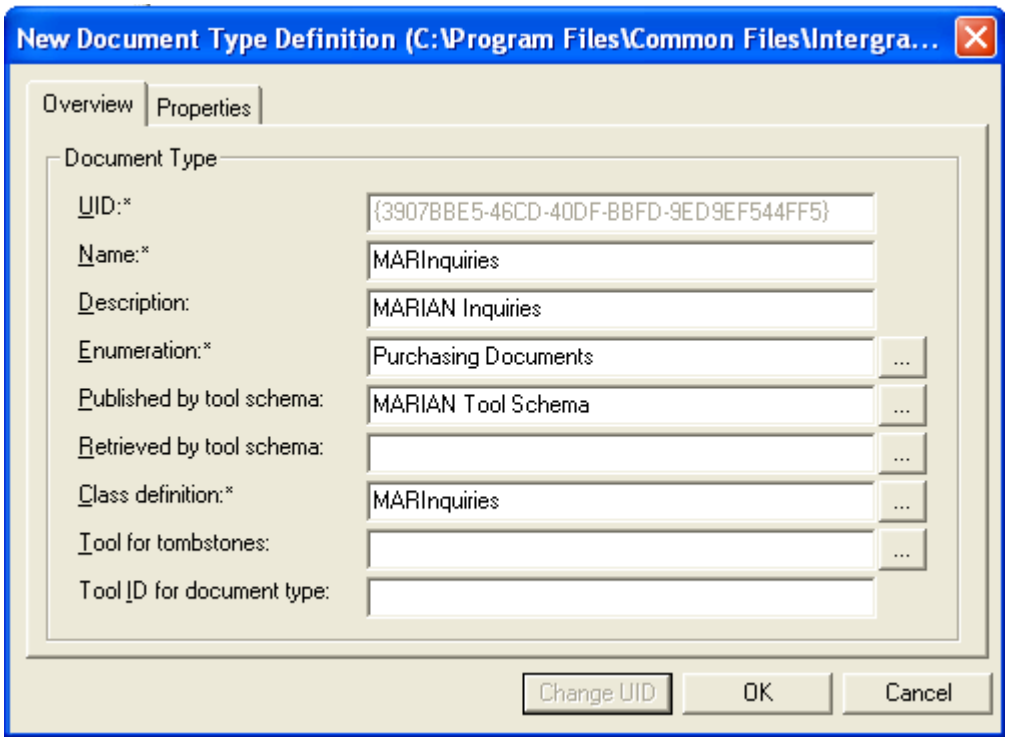


10. To define authoring tools that can retrieve the new document type, click ... beside the **Retrieved by tool schema** box, and select the tool schemas for the retrieving tool or tools.

TIP If a class definition for the document type does not already exist, click **New** at the bottom of the **Possible ClassDef Values for <DocumentType>** dialog box, and create a new class definition with a relationship to the appropriate component schema.



11. Click **OK** to save the new document type and its relationships with the tool schemas that publish and retrieve it.



See Also

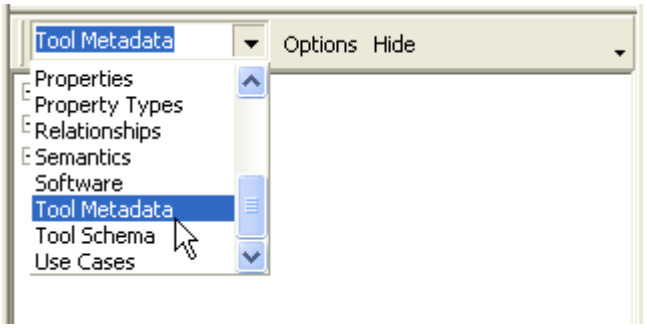
Publish and Retrieve File Types for Tools (on page 239)
Externalize Authoring Tool Metadata (on page 240)
Generate Authoring Tool Metadata Files (on page 257)

Update Document Types that Existing Tools Can Publish and Retrieve

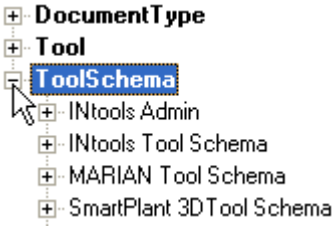
1. After you open schemas, tool schemas, or data files, click the type of file that you want to view on the **View** menu. For example, to view the schema, click **View > Schema (All)**.
2. Edit the EFSchma.cfg file to make EFSchema.xml editable. For more information, see *Modify a File Configuration* (on page 94).

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Click **Editor** in the **Form** list instead.
3. Click **OK**.

4. In the bottom tree view, click **Tool Metadata** in the **Select other classe**s list box to display SmartPlant schema objects for tool metadata, including tool objects, tool schema objects, and document types.

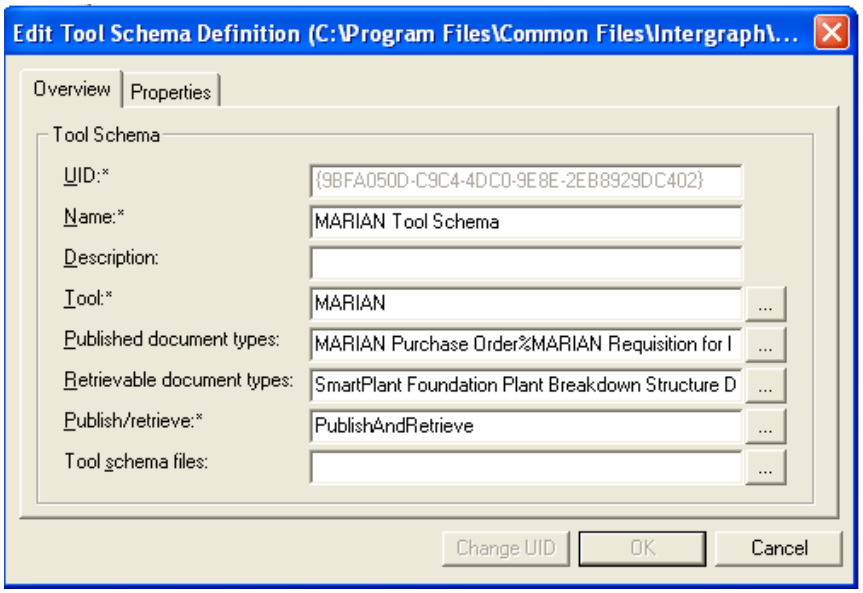


5. In the bottom tree view, expand the **ToolSchema** node.

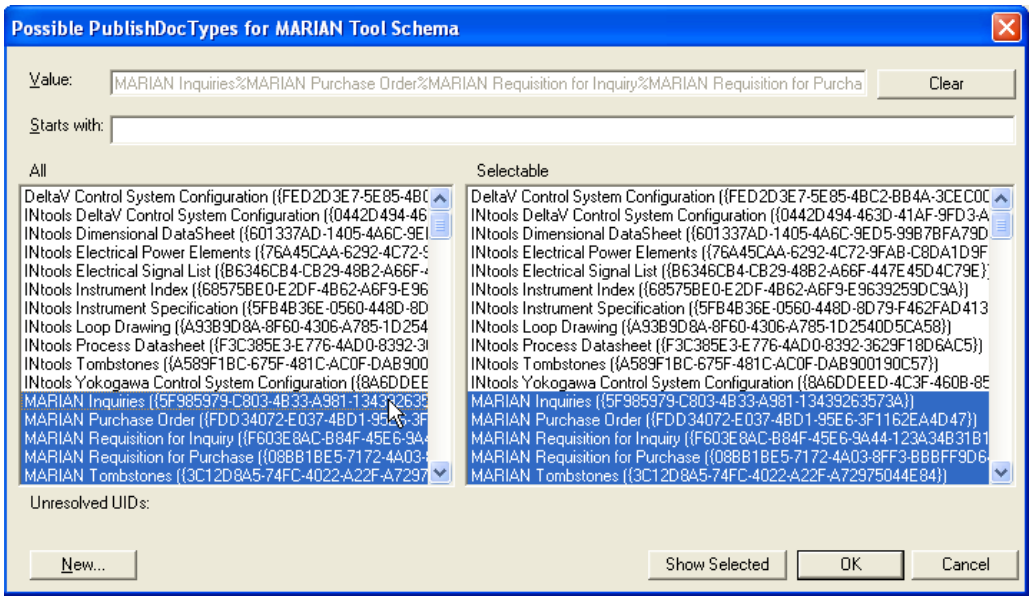


6. Under **ToolSchema**, double-click the tool schema to which you want to add publish and retrieve file types. The tool schema defines publishable and retrievable file types for an authoring tool.

7. In the **Edit Tool Schema Definition** dialog box, click ... beside **Published document types** to add document types that this tool can publish.



8. In the **PublishableDoc Types for <Tool Schema>** dialog box, select the existing document types that you want the authoring tool to publish, and click **OK**.



9. To add document types that this tool can publish, click ... beside **Retrieved document types**, and select the existing document types to retrieve.
10. Click **OK** in the **Edit Tool Schema Definition** dialog box.
11. Click **File > Save Schema File** to save the changes in the SmartPlant schema.
12. Edit the EFSchma.cfg file to make EFSchema.xml read-only again. For more information, see *Modify a File Configuration* (on page 94).

See Also

Publish and Retrieve File Types for Tools (on page 239)
Externalize Authoring Tool Metadata (on page 240)

Generate Authoring Tool Metadata Files

1. Open the SmartPlant schema configuration file (EFSchema.cfg). For more information, see *Open a File Configuration* (on page 94).

   TIP The SmartPlant schema configuration file references the SmartPlant schema file (read-only), other schema files (read-only), and your schema extensions file (editable). The default schema extensions file is Extensions1.xml, and when you open EFSchema.cfg, your changes are saved in Extensions1.xml by default. However, you can specify additional extensions files to store your schema modifications. For more information, see *Modify a File Configuration* (on page 94).

2. Click **Tools > Schema > Generate Tool Metadatas**.

   TIP The software generates metadata XML files for each authoring tool that has metadata defined in the SmartPlant schema. The XML file names are in the following format: <ToolID>_meta.xml, where ToolID is the authoring tool's ID defined in the SmartPlant schema. The tool metadata files are placed in the same location as the SmartPlant schema file you opened as part of the SmartPlant schema configuration file.

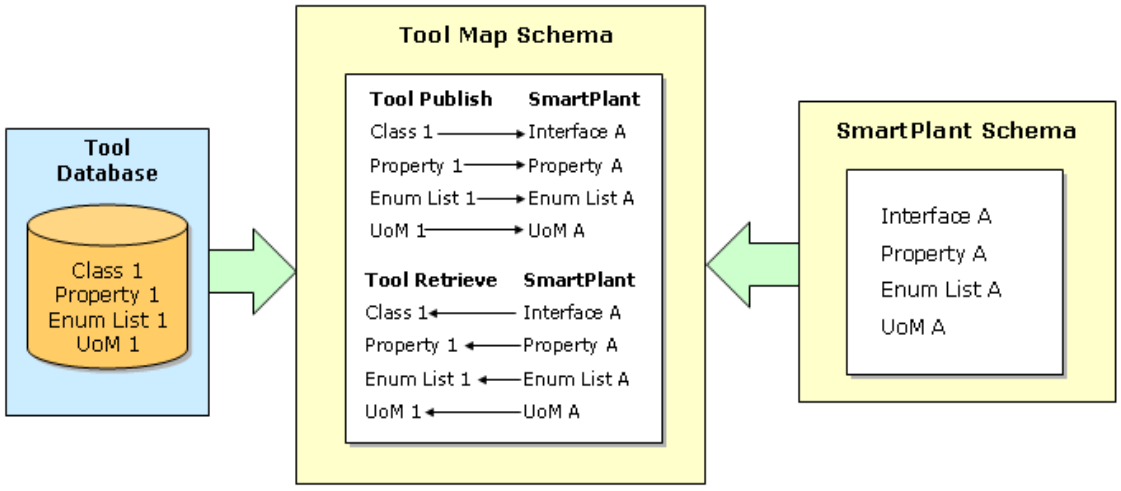
SECTION 12

SmartPlant Mapping Overview

The SmartPlant integration approach is based on a publish/retrieve paradigm. Applications publish their own data and retrieve data published by other applications. The publish and retrieve processes are done using an adapter, which is a piece of software that implements specific interfaces used to support SmartPlant integration.

The adapter must bridge the gap between the application's representation of its data and the SmartPlant representation of equivalent data. This bridge process involves defining a mapping between the application's representation (its schema) and the SmartPlant representation (its schema). SmartPlant mapping is the bridge that is used to integrate SmartPlant applications.

The following graphic shows a simple example of how the authoring tool map schema defines the mapping between the tool data model (schema) and the SmartPlant schema:



The mapping process involves two basic steps:

1. Defining the tool map schema objects that reflect the application's schema
2. Mapping those tool schema objects into (publish) and out of (retrieve) the SmartPlant schema

NOTE You can find individual mapping examples for authoring tools in the *Authoring Tool Mapping Examples* section of the *SmartPlant Mapping User's Guide*.

See Also

Work Processes for Mapping for Tools with Metadata Adapters (on page 327)

Tool Map Schemas

Tool map schemas describe the mapping between objects in the tool database structure (tool metadata) and the SmartPlant schema. For example, the Drawing class in SmartPlant P&ID maps to the PIDDrawing class in the SmartPlant schema. This mapping is used during publish and retrieve to convert tool data to SmartPlant data and vice versa. Every integrated authoring tool has a tool map schema.

When a document is published, the authoring tool adapter uses the tool map schema to convert the data used by the tool to a format that adheres to the SmartPlant schema. When a data file is retrieved into an authoring tool, the authoring tool adapter uses the tool map schema to convert the SmartPlant XML to the format used by the tool.

Tool map schemas can be edited directly in the Schema Editor, or they can be synchronized with the tool metadata using the **Edit Engineering Tool Options** command for authoring tools that have a metadata adapter. The **Edit Engineering Tool Options** is only available after a SmartPlant Foundation connection exists. You can establish this connection by checking the SmartPlant schema file out of the SmartPlant Foundation Desktop Client and launching the Schema Editor from the client. For more information about metadata adapters and tool map schemas, see *Mapping with Metadata Adapters* (on page 291).

For more information about the delivered mapping between the SmartPlant authoring tools, see the SPECPMappingReports.zip, located at [*drive*]:Program Files (x86)\SmartPlant\Foundation\< *version_number* >\Server\Help. These reports, generated by SmartPlant Enterprise Control Panel, include:

- The mapping between classes, properties, and enumerations in the map file
- The properties that are mapped out-of-the-box for publishing and retrieving information
- The properties that are unmapped

The reports also show an example of a comparison between properties from two authoring tools.

See Also

Cardinalities for Mapping (on page 267)
Comparing Tool Map Schema with the SmartPlant Schema (on page 304)
Connecting to a SmartPlant Foundation Plant (on page 296)
Mapping Directly Against the SmartPlant Schema (on page 277)
Map Enumerated List and Map Unit of Measure List Mapping (on page 283)
Types of Mapping for Map Class Definitions (on page 271)
Mapping Relationship Definitions (on page 266)
Tool Objects Required for Metadata Adapters (on page 297)
Synchronizing Tool Map Schemas with Tool Metadata (on page 303)
Tool Schema Models (on page 260)
Tool Map Schema Objects (on page 261)
Using View Definitions to Define Mapping (on page 275)

Tool Schema Models

The tool schema mapping supported by the Schema Editor/Schema Component allows a tool to describe its schema using any of the following underlying models:

- Simple class model
- Class inheritance model
- Class/interface model

These three models should be adequate to describe even complex application schema models.

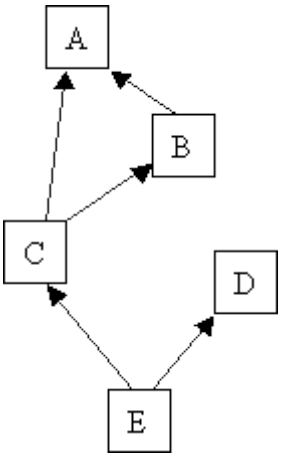
Simple Class Model

The simple class model is a model in which there is no inheritance, and there are no interfaces. This will obviously only occur for very simple and/or flat tool models.

Class Inheritance Model

With the class inheritance model, classes may inherit from other classes. In this model, the classes that inherit are the specialized classes, and the classes that are inherited from are the generalized classes. A class can both inherit from other classes and be inherited by other classes (multiple levels of inheritance). A class may also inherit from multiple classes (multiple inheritance).

The following diagram illustrates a class inheritance model with both multiple levels of inheritance and multiple inheritance at one level. Arrows are used to show the inheritance. Therefore, class C inherits from class A and class B. Class E inherits from class C and class D.

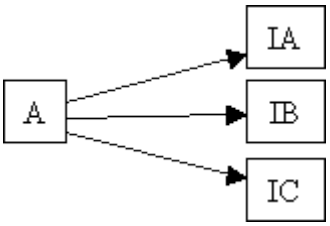


Class/Interface Model

A class/interface model is similar to a class inheritance model with the following differences:

- A class/interface model only has two levels (class and interface), but a class inheritance model can have multiple levels.
- The classes (actually interfaces) inherited from have no implementation in a class/interface model, but classes inherited from in a class inheritance model can have implementation (software methods).

A simple class inheritance model is shown below. In this model, class A inherits from (realizes) interfaces IA, IB, and IC.



See Also

Tool Map Schemas (on page 259)

Tool Map Schema Objects

The following tool map schema objects are used to define the mapping between the authoring tool and the SmartPlant schema:

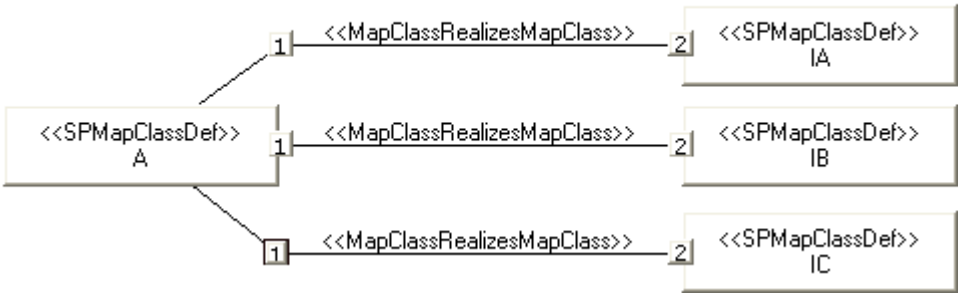
- Map class definitions (instances of SPMapClassDef)
- Map property definitions (instances of SPMapPropertyDef)
- Map relationship definitions (instances of SPMapRelDef)
- Map enumerated list definitions (instances of SPMapEnumListDef)
- Map enumerations (instances of SPMapEnumDef)
- Map unit of measure list definitions (instances of SPMapUoMListDef)
- Map units of measure definitions (instances of SPMapUoMDef)
- Map software definition (maximum of one instance of SPMapSoftware)
- Relationships between map objects and other map objects (map schema) and between map objects and SmartPlant objects (mapping)

Map Class Definitions

Map class definitions are used to represent both application classes and application interfaces. Therefore, all of the classes and interfaces from the simple class model, the class inheritance model, and the class/interface model are modeled using map class definitions (instances of SPMapClassDef). The following UML diagram represents the modeling of a class inheritance model using a tool map schema:



The following UML diagram represents the modeling of a class/interface model using a tool map schema:

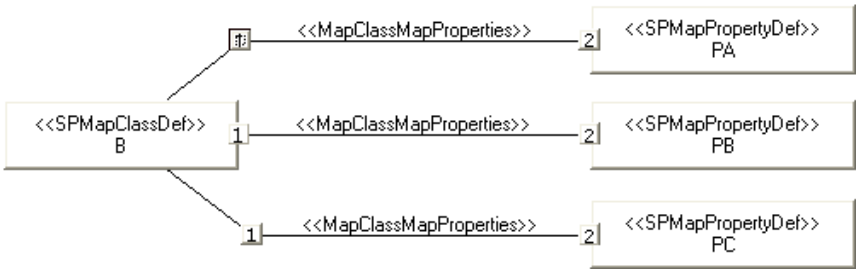


Notice that inheritance in the class inheritance model uses MapClassMapClass relationships, but the relationships between the class and interface in the class/interface model are instances of the MapClassRealizesMapClass relationship definition.

Map Property Definitions

In representing a simple class model or a class inheritance model, all of the classes in the model can have (expose) properties. In a class/interface model, only the interfaces can have properties.

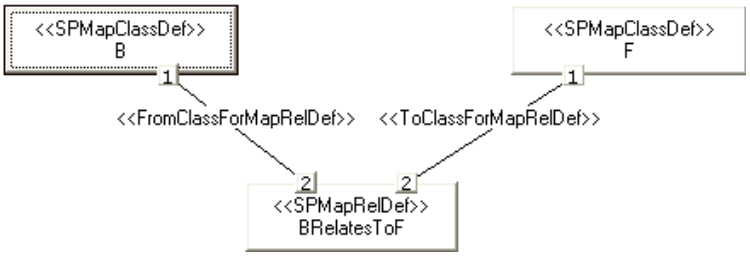
Application properties, when modeled using a tool (map) schema, are instances of map property definitions (SPMapPropertyDef). These map property definitions are tied to their map class definitions using instances of the MapClassMapProperties relationship definition. Therefore, if map class B has properties PA, PB, and PC, they would be modeled in the tool schema as shown in the following diagram:



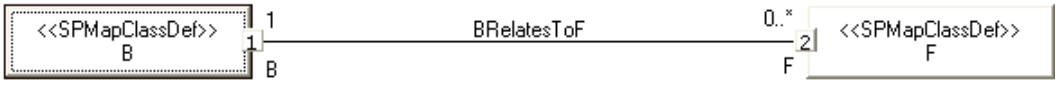
Map Relationship Definitions

All but the simplest of applications have relationship definitions between their classes or interfaces that identify the types of relationships that can exist in the application's data. These relationship definitions are modeled using map relationship definitions (instances of SPMapRelDef).

Therefore, if instances of class B can be related to instances of class F, the relationship definition BRelatesToF would be modeled as:



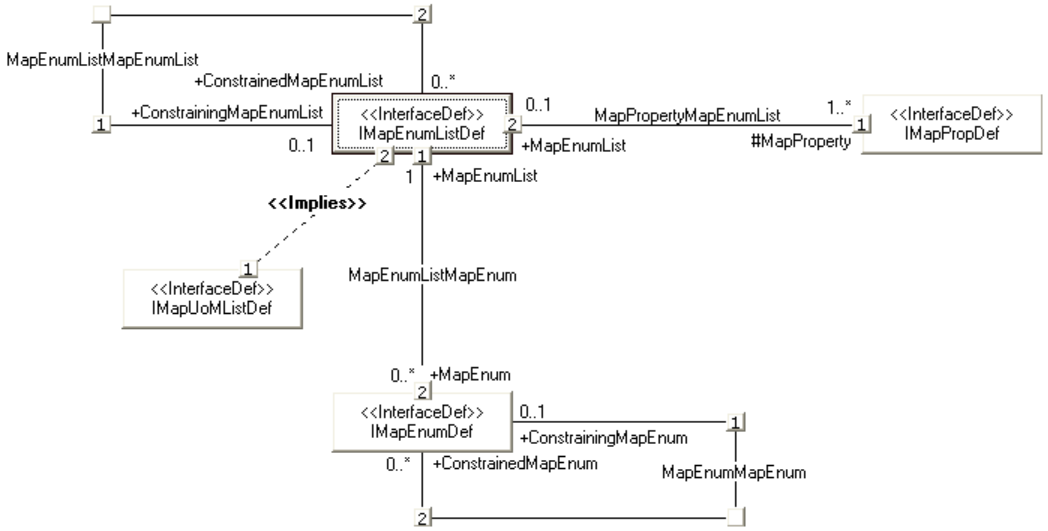
OR



A map relationship definition (SPMapRelDef) has two different relationship definitions (FromClassForMapRelDef and ToClassForMapRelDef) to the map class definitions at its "from" and "to" ends.

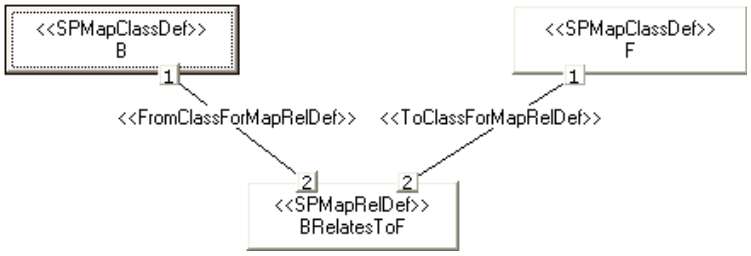
Map Enumerated List and Map Enumeration Definitions

The following interface model shows how map enumerated lists and map unit of measure lists are modeled within the map schema.

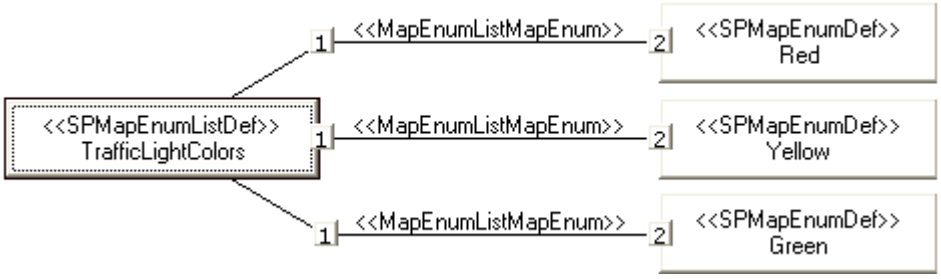


A map unit of measure list is a specialization of a map enumerated list as indicated by the Implies relationship between IMapUoMListDef and IMapEnumListDef. Therefore, the same relationship types are used for both map enumerated lists and map unit of measure lists.

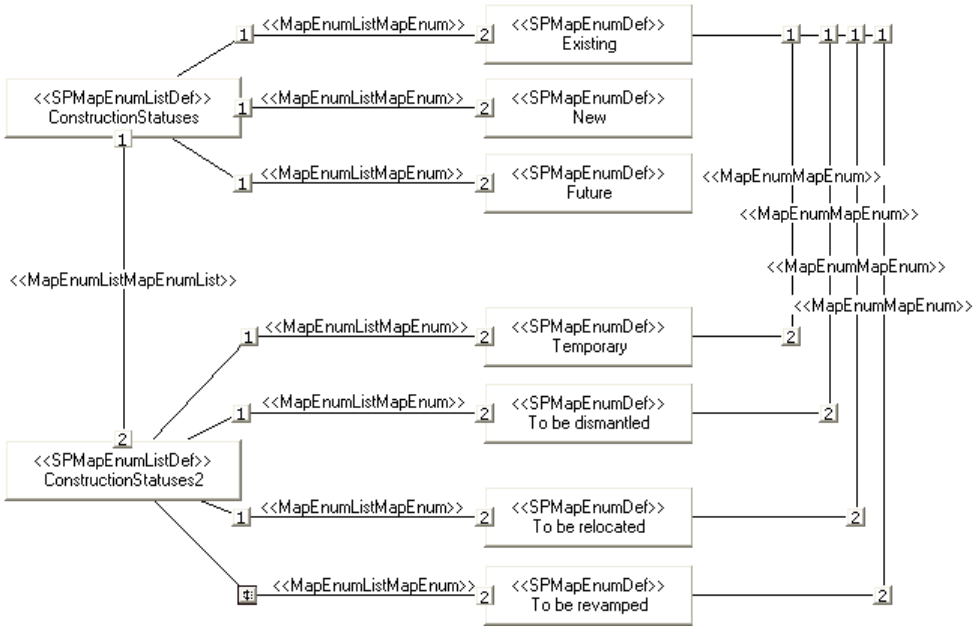
A map property that is scoped by a map enumerated list (or a map unit of measure list) should have a MapPropertyMapEnumList from the map property to the map enumerated list (or map unit of measure list).



Application enumerated lists are described using map enumerated lists (instances of SPMapEnumListDef) and map enumerations (instances of SPMapEnumDef). Using these objects, an application enumerated list for TrafficLightColors would be modeled as:

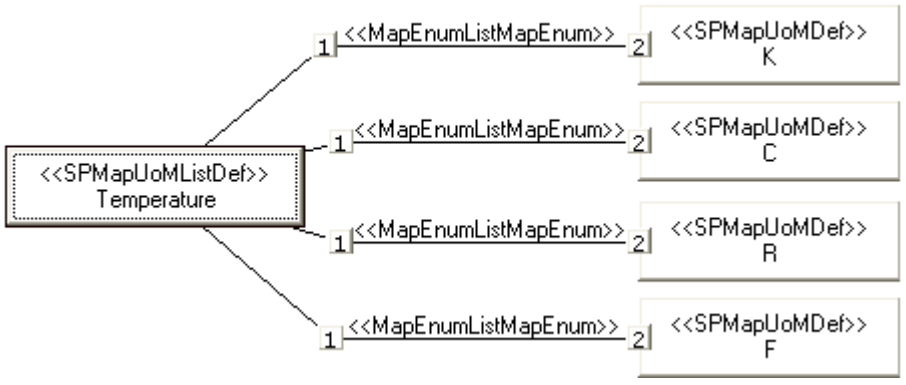


If an enumerated list contains more than one level, then it is modeled using a map enumerated list definition for each level. The map enumerated list definitions are related as are the individual map enumerations. For example, the enumerated list for ConstructionStatuses would look like the following:



Map Unit of Measure List and Map Unit of Measure Definitions

Application units of measure are modeled very similarly to enumerated lists with the exception that the list for the unit of measure is modeled as a unit of measure list (instance of SPMapUoMListDef) and the individual units of measure are modeled as instances of SPMapUoMDef. With this approach, a temperature unit of measure list would be modeled as:



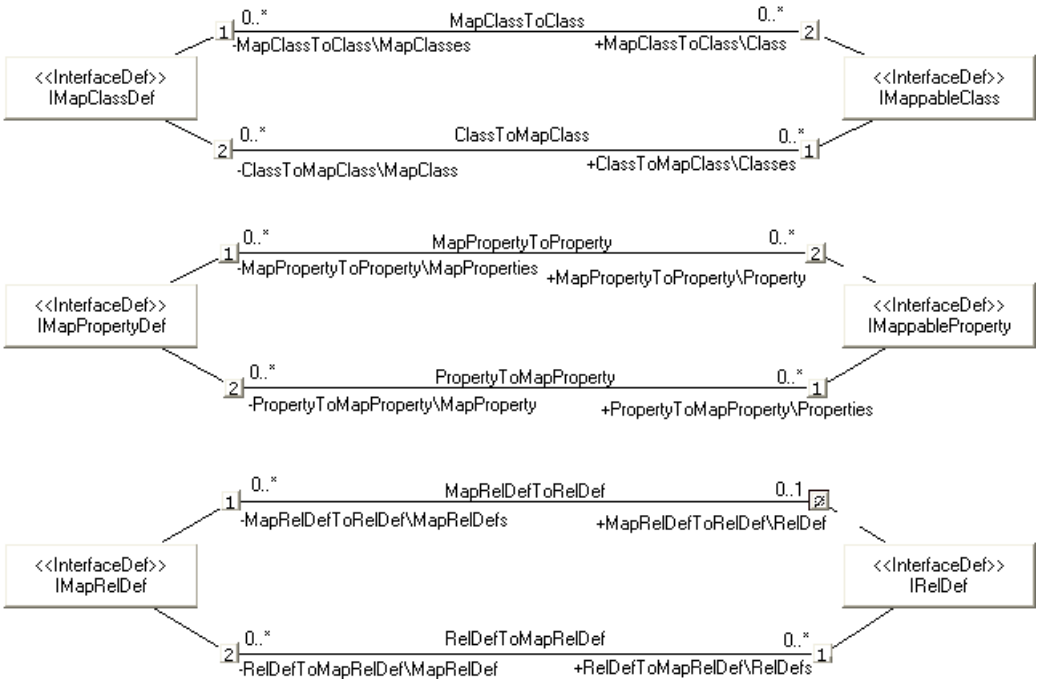
The same relationship types (MapEnumListMapEnum) are used for unit of measure lists and enumerated lists.

See Also

Tool Map Schemas (on page 259)

Mapping Relationship Definitions

The following UML diagrams illustrate the various relationship definitions that are used to define the mapping between the tool map schema and the SmartPlant schema:



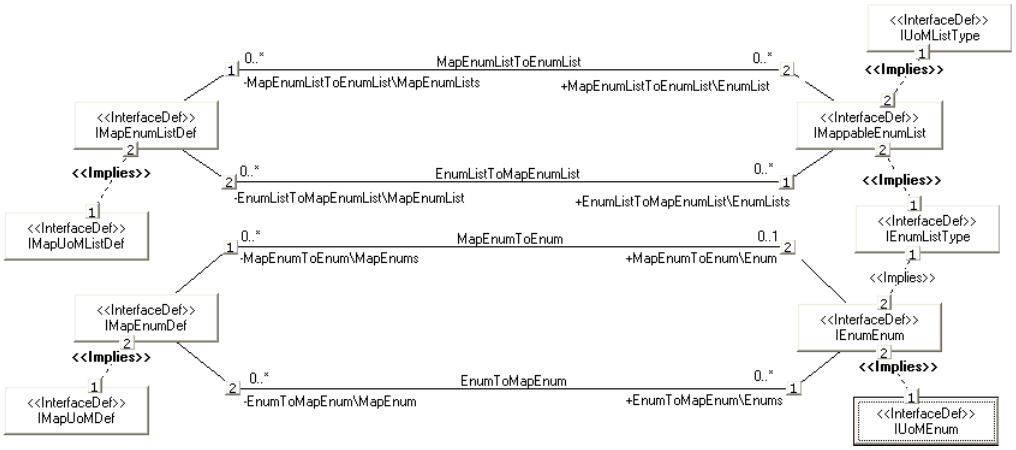
For each of the various map objects, there are two relationship definitions that are used for mapping:

- Map<*Object*>To<*Object*>, where <*Object*> is the type of map object, is used for defining mapping from the tool schema into the SmartPlant schema for publish.
- <*Object*>ToMap<*Object*> is used for defining mapping from the SmartPlant schema to the tool schema for retrieve.

IMappableClass is realized by both ClassDef and InterfaceDef. Therefore, map class definitions can be mapped to either class definitions or interface definitions.

IMappableProperty is realized by PropertyDef and EdgeDef. Therefore, a map property definition can be mapped to either property definitions or edge definitions.

The following UML diagram shows the relationship definitions used to define mapping for tool enumerated lists and entries and unit of measure lists and entries:



Like the other map objects, map enumerated lists, map unit of measure lists, map enumerations, and map units of measure have two map relationships to SmartPlant schema objects, Map<*Object*>To<*Object*> for publish and <*Object*>ToMap<*Object*> for retrieve.

Because IMapUoMListDef implies IMapEnumListDef, map unit of measure lists and map enumerated lists both map to IMappableEnumList, an interface common to both SmartPlant unit of measure lists and SmartPlant enumerated lists. Although in theory this allows a map unit of measure list to be mapped to a SmartPlant enumerated list and vice versa, in actuality map enumerated lists map to SmartPlant enumerated lists, and map unit of measure lists map to SmartPlant unit of measure lists.

Map enumerations can map to either SmartPlant enumerations or SmartPlant enumerated lists. Map units of measure can map to SmartPlant units of measure. Because IMapUoMDef implies IMapEnumDef, the mapping relationships used for map enumerations and map units of measure are the same. As for the unit of measure lists and enumerated lists, this would appear to make it possible for map enumerations to map to SmartPlant units of measure and vice versa, but in actuality map enumerations map to SmartPlant enumerations, and map units of measure map to SmartPlant units of measure.

See Also

Tool Map Schemas (on page 259)

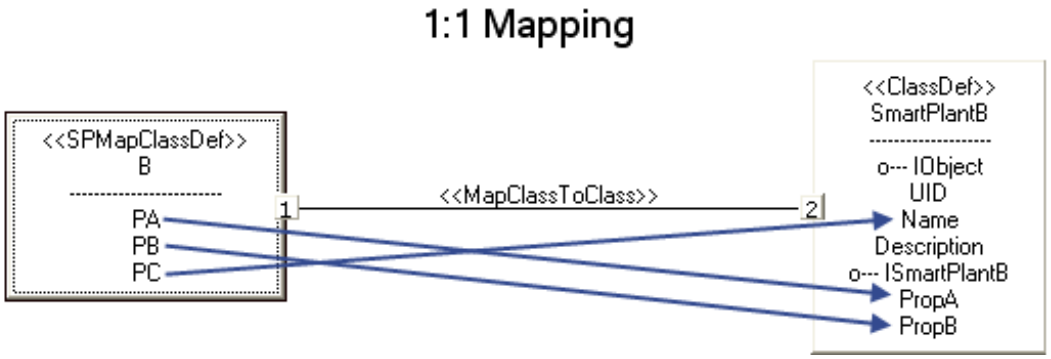
Cardinalities for Mapping

The MapClassToClass and ClassToMapClass relationship definitions are used to define the mapping between map class definitions and SmartPlant class and/or interface definitions. For publishing, a map class definition can be mapped to either a class definition or the primary interface definition for the class definition. For retrieval, the map class definition can be mapped from class and/or interface definitions. However, Intergraph strongly recommends that you define retrieve mapping using SmartPlant interface definitions rather than class definitions because doing so enables plug-and-play behavior.

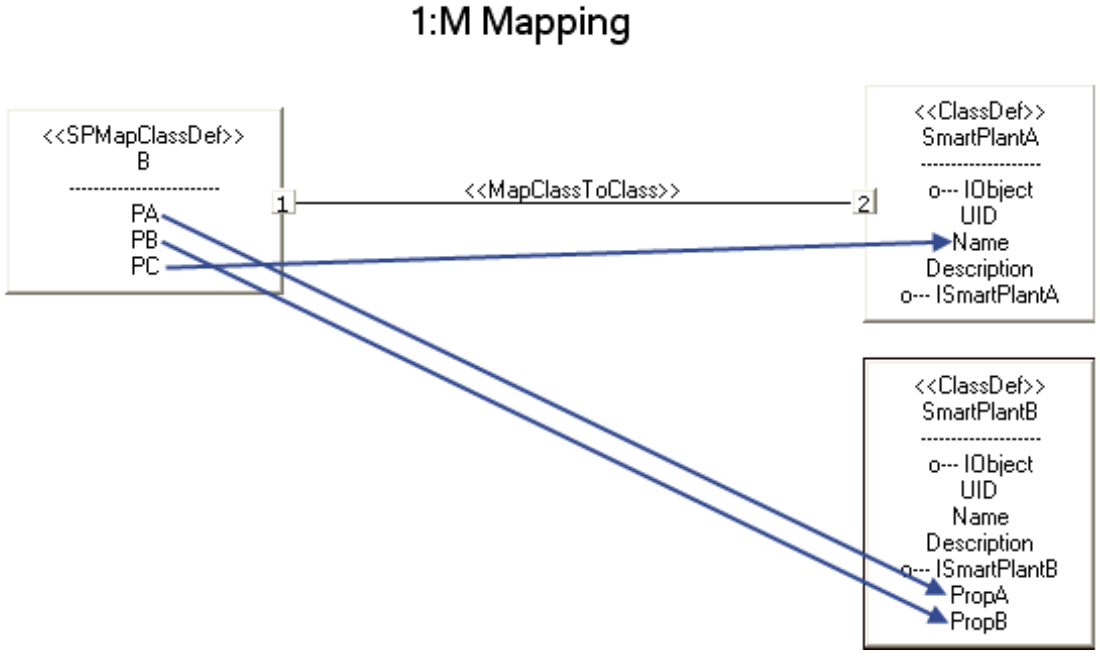
The definition of the mapping for a class definition involves the mapping of its properties. When the map properties for a map class definition are included, a number of possible map cardinalities can occur: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), and

many-to-many (M:N). The following diagrams describe these map cardinalities using a simple class model with no inheritance and no interfaces.

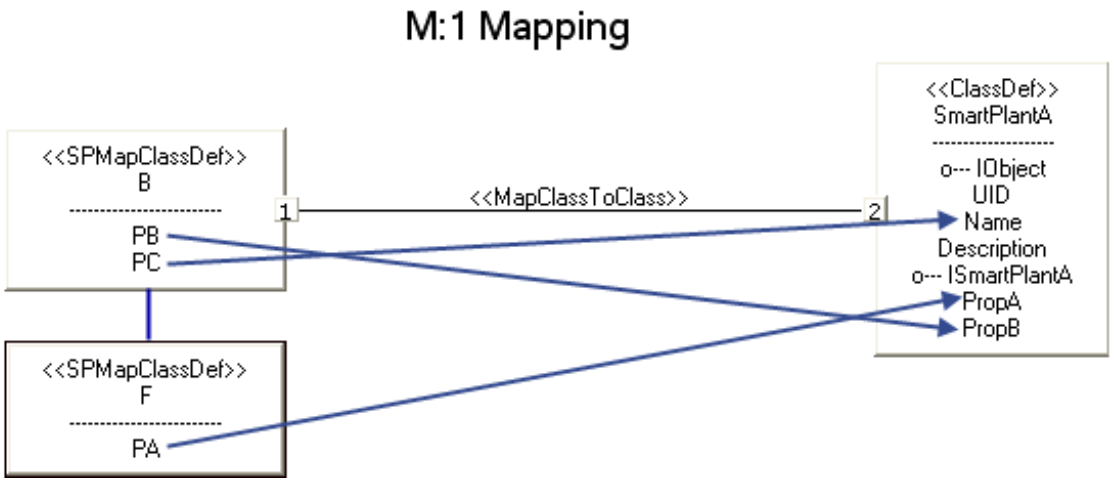
In one-to-one (1:1) mapping, the properties for one map class definition map to one SmartPlant class (or primary interface) definition.



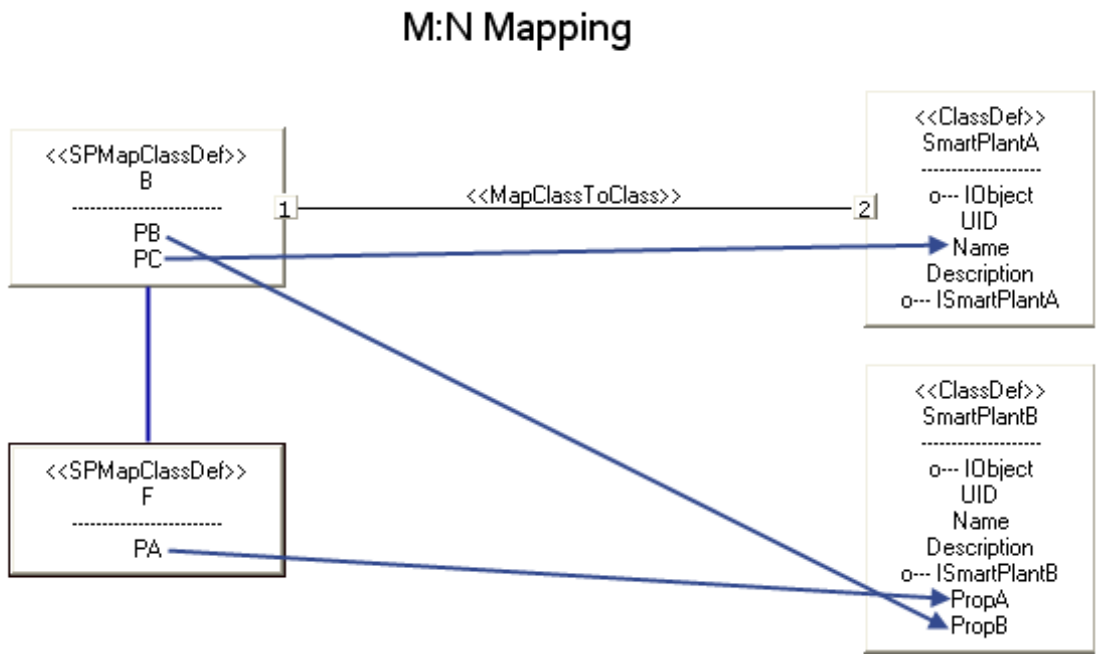
In one-to-many (1:M) mapping, map properties for one map class definition map to properties on more than one SmartPlant object. In the following example, map properties for map class definition B map to properties on SmartPlant class definitions SmartPlantA and SmartPlantB. The line connecting SmartPlantA and SmartPlantB in this diagram is an edge definition that goes between interface definitions realized by the two class definitions.



In many-to-one (M:1) mapping, map properties on related map class definitions map to one SmartPlant class or interface definition. In the following diagram, the line connecting map class B with map class F is a map edge definition.



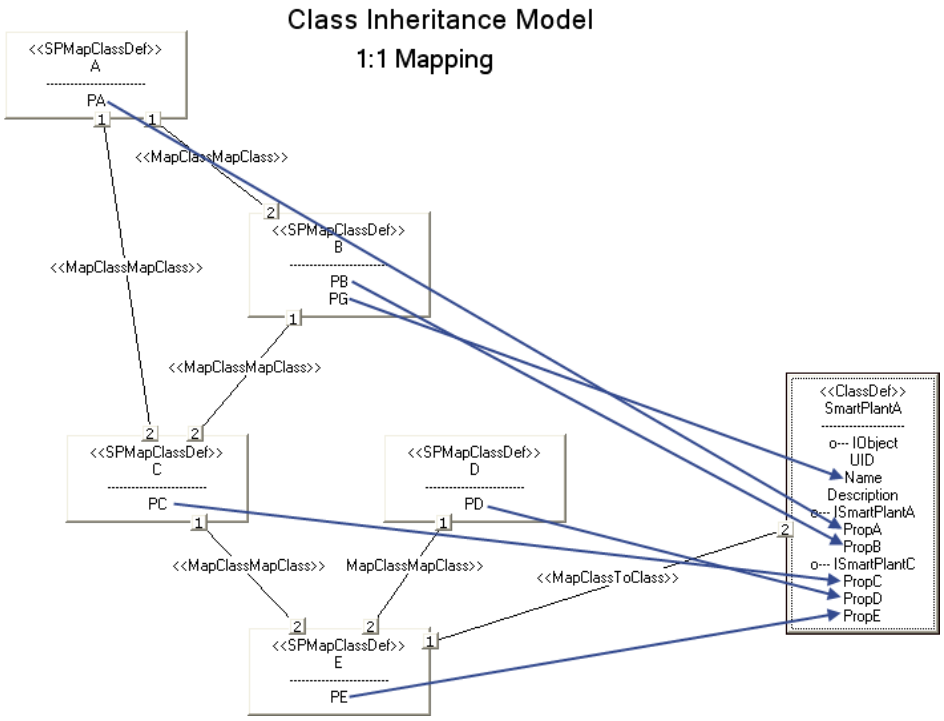
In many-to-many (M:N) mapping, map properties on related map class definitions map to property definitions on more than one class and/or interface definition.



Cardinalities for Class Inheritance and Class/Interface Models

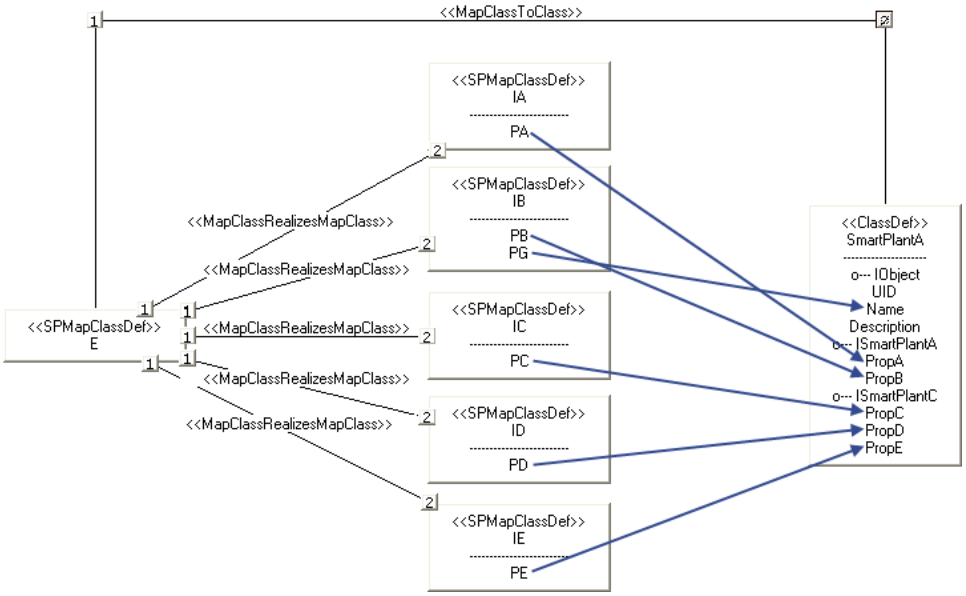
The same cardinalities of mapping can be used for class inheritance models and class/interface models. When a class inheritance model is used, an application object is described by a map class definition and all of the map class definitions from which it directly or indirectly inherits. If map class definition E inherits directly or indirectly from map class definitions A, B, C, and D,

one-to-one mapping can involve map properties from any of these map class definitions as shown in the following diagram:



Likewise, when a class/interface model is used, an application object is described by a map class definition for the class and a separate map class definition for each interface definition. For a class/interface model, the property mapping is done using map properties on the realized map interfaces (map class definitions with IsInterface=True).

Class/Interface Model
1:1 Mapping



See Also

Tool Map Schemas (on page 259)

Types of Mapping for Map Class Definitions

Just as the tool map schema supports simple class models, class inheritance models, and class/interface models, there are three types of mapping that closely correspond to these models:

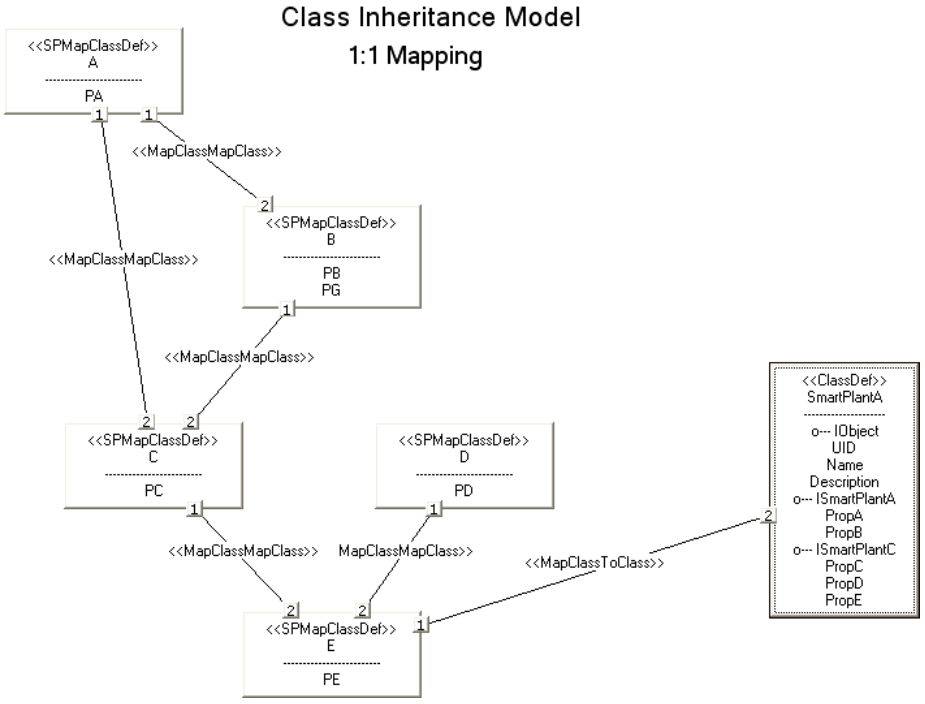
- Direct mapping
- Mapping including generalized classes in map
- Mapping including realized classes in map

The type of mapping that is used has no impact on the cardinalities of mapping. All three types of mapping can be used in one-to-one, one-to-many, many-to-one, and many-to-many mapping cases.

Direct Mapping

The simplest mapping type is direct mapping. In direct mapping, only map properties for the map class being mapped are used. Any generalized or realized map classes for the map class are not used in the mapping.

For example, if direct mapping is used for map class E, shown in the following diagram, then only property PE will be mappable. Properties on generalized map classes A, B, C, and D will not be included in the mapping.

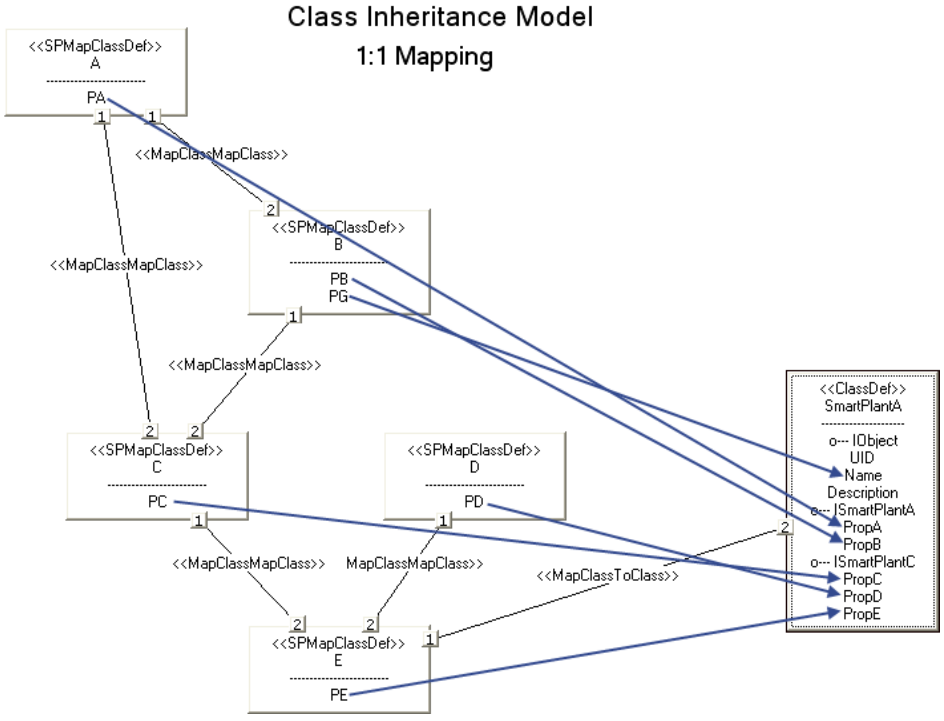


Direct mapping is useful in two cases:

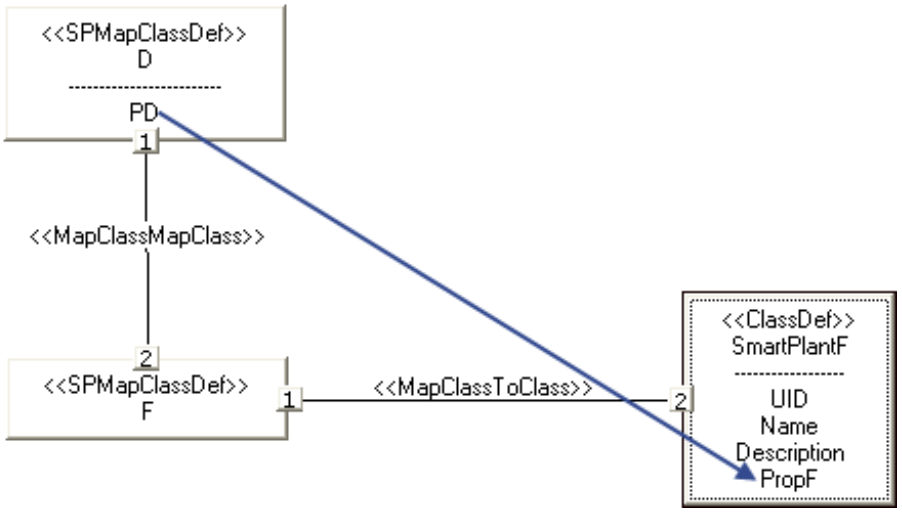
- When the model is a simple class model with no inheritance or interfaces
- When the properties for a generalized map class or a map interface always map the same regardless of the specialized class (for generalized classes) or the realizing map class (for map interfaces). For example in the previous UML diagram, if property PD on map class D always maps to PropD on interface ISmartPlantC, then the mapping for map class D can be defined to be direct.

Mapping Including Generalized Map Classes

When the tool map schema is used to represent a class inheritance model, the mapping type is usually mapping including generalized map classes. With this mapping type, the definition of the map property mapping for a map class will include properties from direct and indirect generalizations of the map class. For the previous example, the mapping of map class E includes the map properties from generalized map classes A, B, C, and D as shown in the following diagram:

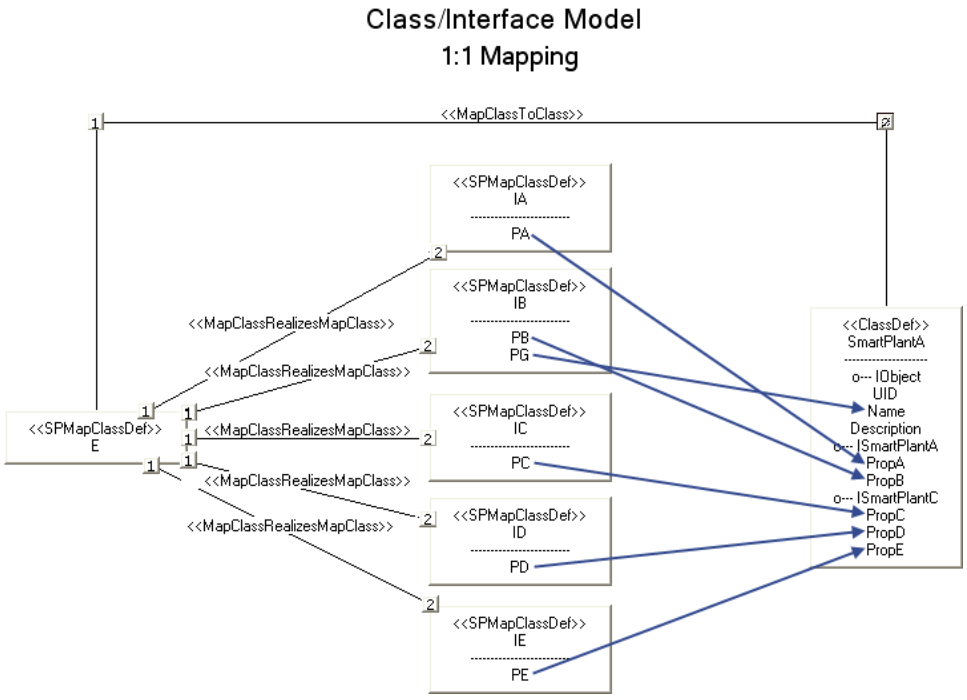


A characteristic of mapping using generalized map classes is that a property on a generalized class might map differently depending on the specialized class being mapped. For example, map property PD on map class D maps to property PropD for specialized map class E in the previous diagram, but the same property for specialized map class F maps to property PF in the following diagram.



Mapping Using Realized Map Classes

Just as mapping using generalized map classes is intended for use with class inheritance models, mapping using realized map classes is intended for use with class/interface models. With mapping using realized map classes, mapping for a map class includes mapping the map properties for each map interface realized by that map class. For example, if map class E realizes map interfaces IA, IB, IC, ID, and IE, then mapping E will include mapping the properties for IA, IB, IC, ID, and IE as shown below.



Like mapping using generalized map classes, mapping involving realized map classes can result in a map property for a map interface being mapped differently depending on the realizing map class.

See Also

Tool Map Schemas (on page 259)

Using View Definitions to Define Mapping

Mapping directly against the SmartPlant schema requires a detailed understanding of that model. For many users, gaining this understanding can be an intimidating undertaking. Defining mapping using view definitions can offer a simpler alternative that can help bypass the need to have a detailed understanding of the SmartPlant schema model.

Mapping using view definitions is especially helpful because, if view definitions are configured, users are not required to search through interface definitions that a class definition realizes to find the properties that they want to map. Instead, the view definition defines pre-selected properties for an object, bypassing the need to understand what interface definition exposes which property.

The **Publish** and **Retrieve** tabs on the **Edit** window for map class definitions use view definitions for viewing and defining the mapping. The **Overview** tab on this **Edit** window allows the user to select the class or interface for publish and retrieve or the explicit view definition to use for publish and retrieve. If the user selects a class definition or interface definition for the mapping, the software determines if a view definition has been defined for mapping to this class or interface definition. If an existing view definition is found that is suitable for defining mapping, then that view definition will be used. If no view definition is found, then a view definition will be dynamically generated.

NOTE The SmartPlant schema contains a small set of predefined view definitions that you can use for mapping. However, not all objects for which you may want to define mapping have view definitions defined in the schema. Dynamically generated view definitions may not be as helpful as predefined view definitions for simplifying mapping.

The general approach for defining publish mapping using a view definition includes the following steps:

- For each application map property, find a corresponding view definition property and map the application property to that view definition property.
- When all application properties for which a corresponding view definition property was found have been mapped, evaluate the remaining application properties and determine which, of any, of those properties should still be mapped. For each application not mapped that should be mapped, either create a new property definition and add it to the view definition or find an existing property definition not in the view definition and add it to the view definition. In either case, after adding the property to the view definition, map the application property to the added view property.
- After all map properties that should be mapped have been mapped, use the **Set Application Properties Not to be Mapped** button  to mark the rest of the application properties as "not to be mapped" for publish and retrieve.
- Evaluate the remaining view definition properties and determine if there are any view definition properties for which an application map property should be created. Create a new application property for each of these view properties (in some cases, the properties may exist on a related application class, and mapping across a map edge definition may be used instead of creating a new application property) and map that new application property to the view property.
- After all view properties that should be mapped have been mapped, use the **Set SmartPlant Properties Not to be Mapped** button  to mark the rest of the SmartPlant view definition properties as "not to be mapped" for publish and retrieve.

See Also

Tool Map Schemas (on page 259)

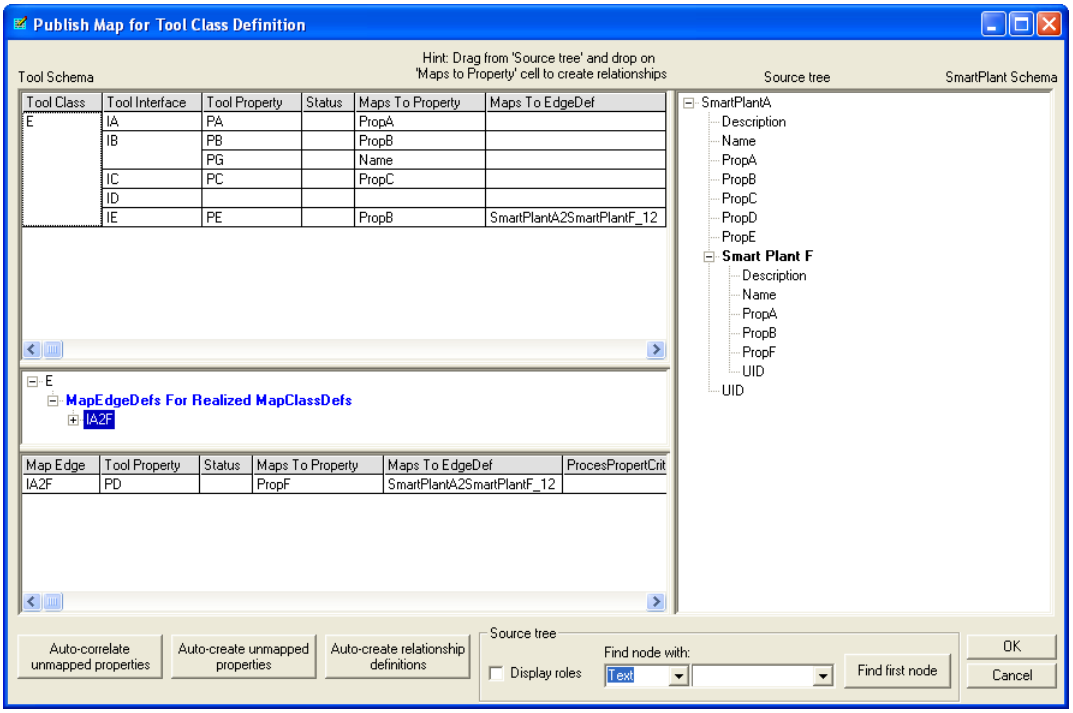
Mapping Directly Against the SmartPlant Schema

While mapping using view definitions is generally easier for users, in some cases mapping directly against the SmartPlant schema may be required. In particular, mapping against the SmartPlant schema is required for the following cases:

- For to-many mapping where a specifically defined view definition has not been created because the automatically generated view definition only handles to-one mapping. An alternative to mapping directly would be to create an explicit view definition for this specific case, but this is probably more complicated than just mapping directly against the SmartPlant schema.
- For from-many mapping. The mapping that can be done using the **Publish** and **Retrieve** tabs is currently limited to from-one mapping.

To map directly against the SmartPlant schema, an understanding of at least the portion of the SmartPlant schema that will be involved in the mapping is required.

The **Publish Map** and **Retrieve Map** buttons available from the **Overview** tab on the **Edit** windows for map class definitions can be used to map directly against the SmartPlant schema. For example, selecting the **Publish Map** button for a map class definition displays the following window to allow mapping directly against the SmartPlant schema:



A similar window is available when you click the **Retrieve Map** button on the **Edit** window for a map class definition.

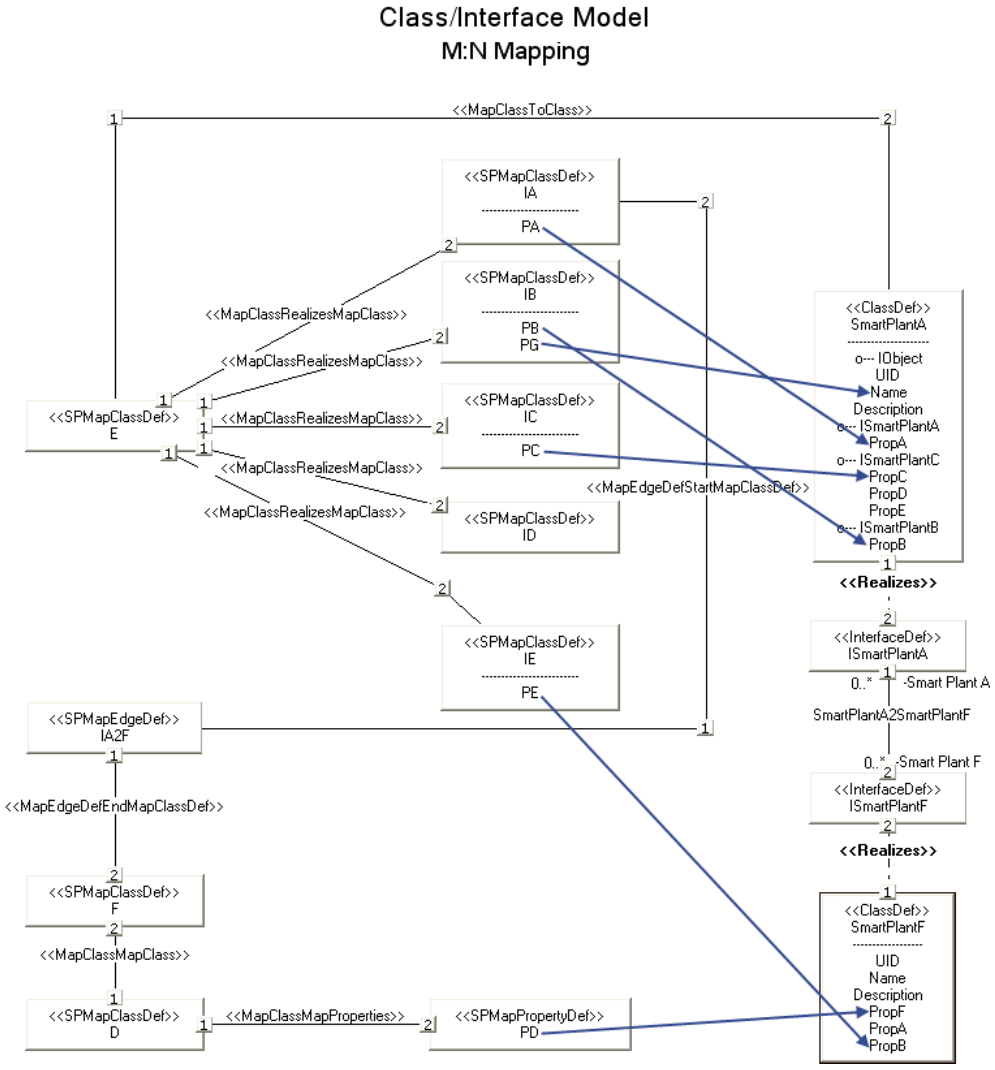
One-to-One and One-to-Many Mapping

To define one-to-one mapping using the **Publish Map for Tool Class Definition** window, you can drag a property definition from the tree control and drop it in a cell for the **Maps to Property**

column in the table on the upper left of the window. If only property definitions are selected from the mapped-to class or interface definition, dragging and dropping establishes a one-to-one mapping relationship between the dragged SmartPlant property definition and the map property for the dropped-in row. In this case the **Maps to EdgeDef** column will be blank for all rows in the table.

Because the **Source tree** on the right side of the window displays both properties and edge definitions for the mapped-to class or interface definition, you can also select a property definition across an edge definition (bolded text in the tree) to define mapping. If you select a property definition across an edge definition, then the software establishes a relationship between the map property for the dropped-in row with the property definition across the edge definition. In this case the **Maps to EdgeDef** column reflects the edge definition across which the property definition was selected.

When property definitions across edge definitions are included this way, then the mapping, instead of being one-to-one, becomes one-to-many. For example, the mapping between property PE and PropB across an edge definition from the previous screenshot appears in the UML diagram below. Notice the arrow going from map property PE on SPMapClassDef IE (realized by SPMapClassDef E) to PropB on SmartPlantF. In this case, the one instance of E maps to both SmartPlantA and SmartPlantF (that is, one application object maps to two SmartPlant objects).

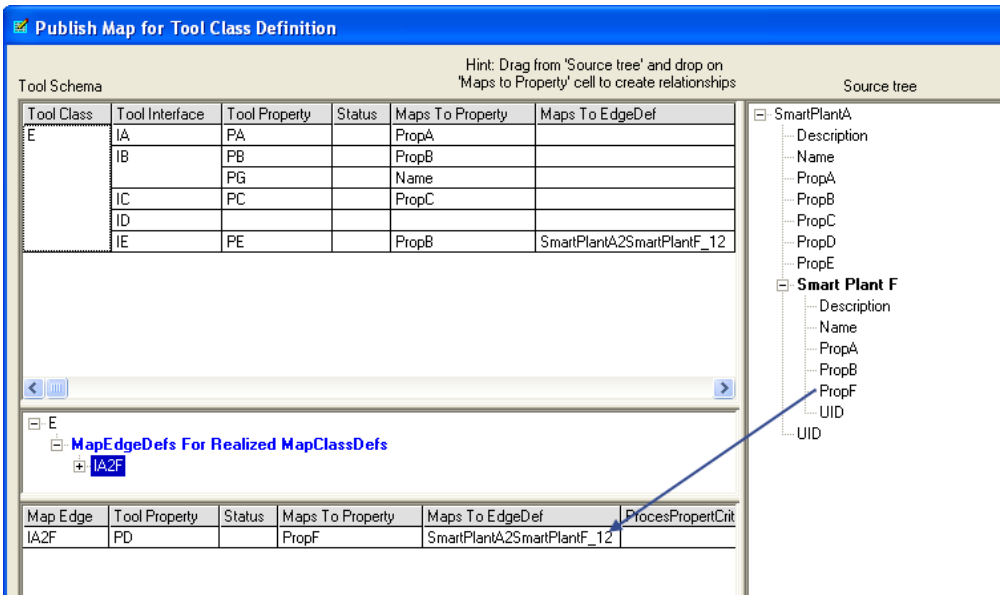


Many-to-One and Many-to-Many Mapping

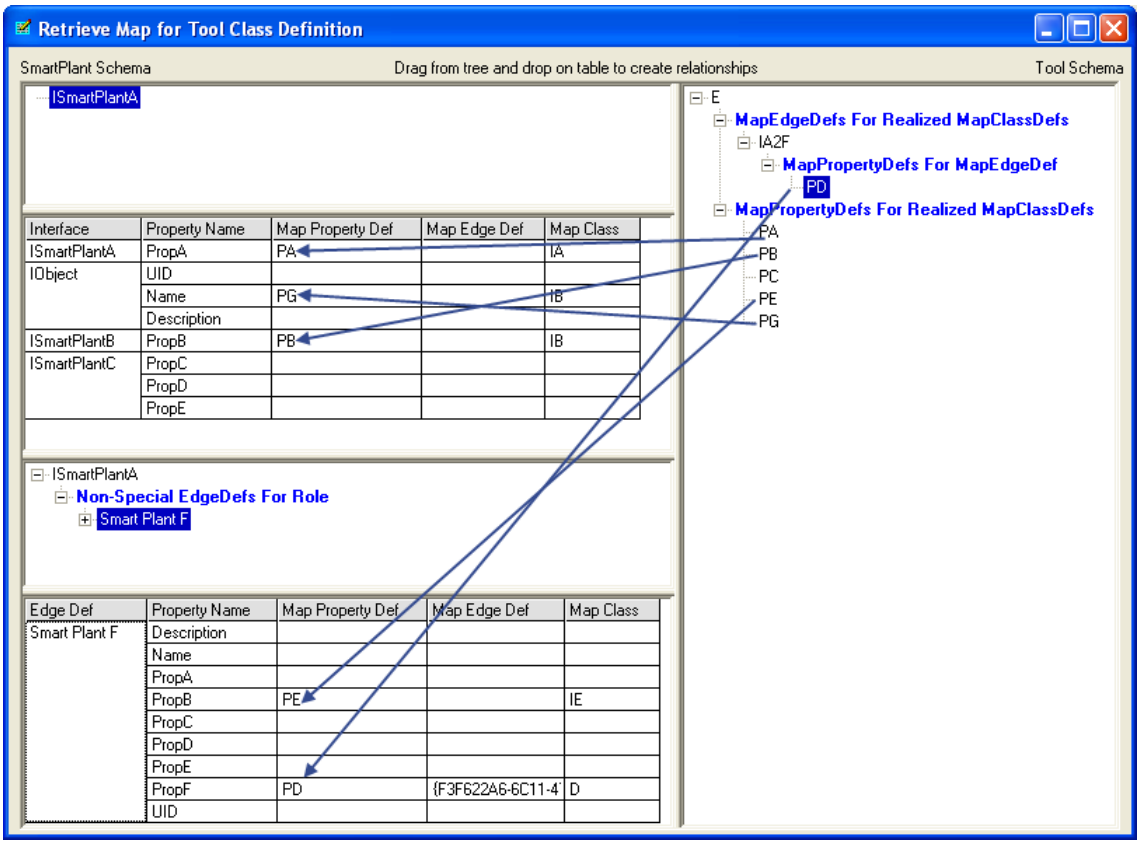
To define many-to-one mapping in the **Publish Map for Tool Class Definition** window, you can drag and drop properties from the mapped-to class or interface definition in the tree view onto the **Maps to Property** column for a map edge definition in the table on the lower left of the window. When you create a many-to-one mapping, the **Maps to EdgeDef** column is blank.

To define many-to-many mapping, you can drag and drop properties that are not from the mapped-to class or interface definition but from a related interface definition onto the **Maps to EdgeDef** column.

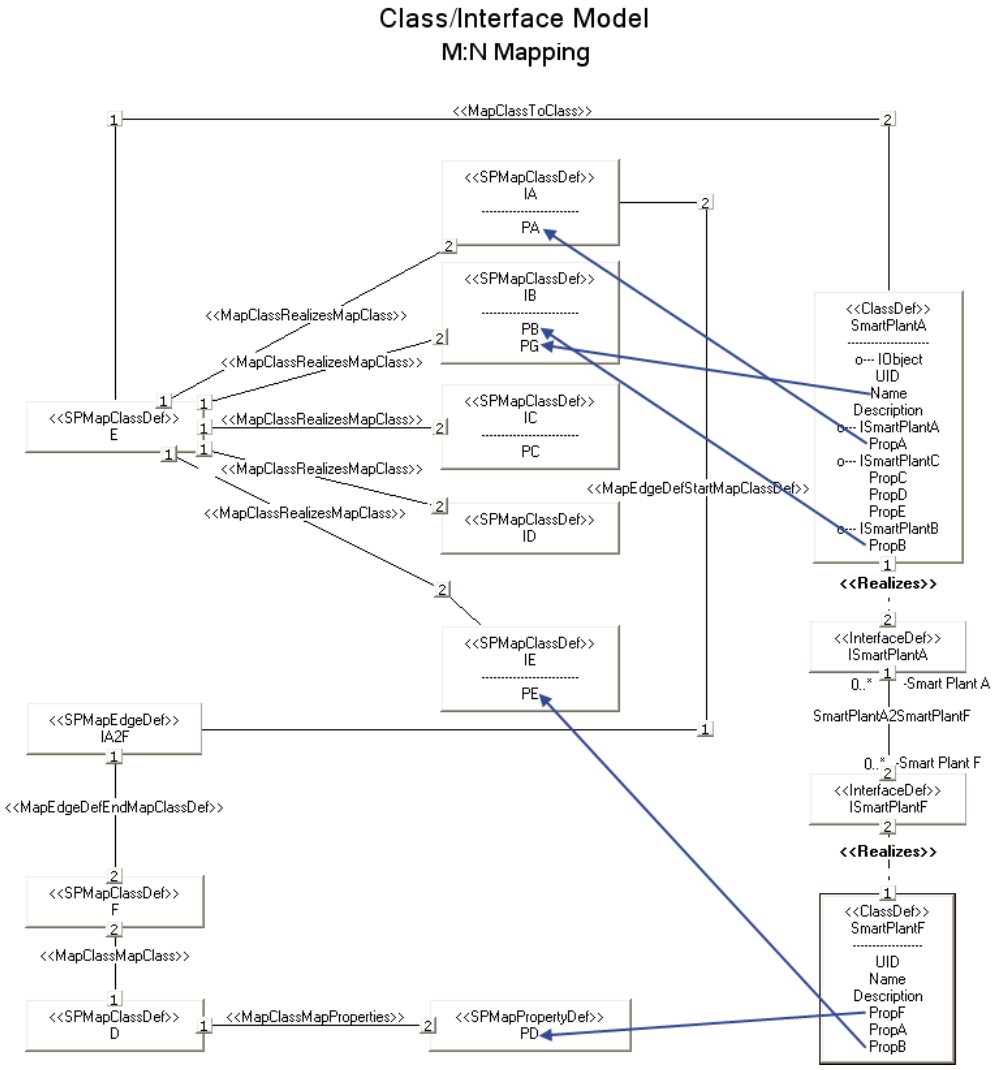
For example, in the following graphic, you can see that tool map property PD for map class definition D across map edge definition IA2F is mapped to PropF for SmartPlantF across edge definition Smart Plant F (for relationship definition SmartPlantA2SmartPlantF). In this case two related map objects (E and F) are mapped to two related SmartPlant objects (SmartPlantA and SmartPlantF). This is an example of a many-to-many mapping.



The **Retrieve Map for Tool Class Definition** window works similarly to the **Publish Map for Tool Class Definition** window. The following example shows retrieve mapping for retrieving from interface definition ISmartPlantA into map class definition E. The arrows represent drag-and-drop operations that were performed to produce the displayed mapping. Note that the **Map Edge Def** column for map properties PA, PB, PE, and PG is blank (because these are map properties for the E map class), but the **Map Edge Def** column for PD shows the UID for map edge definition IA2F.



The UML diagram below illustrates the many-to-many mapping shown in the previous retrieve graphic:



See Also

Tool Map Schemas (on page 259)

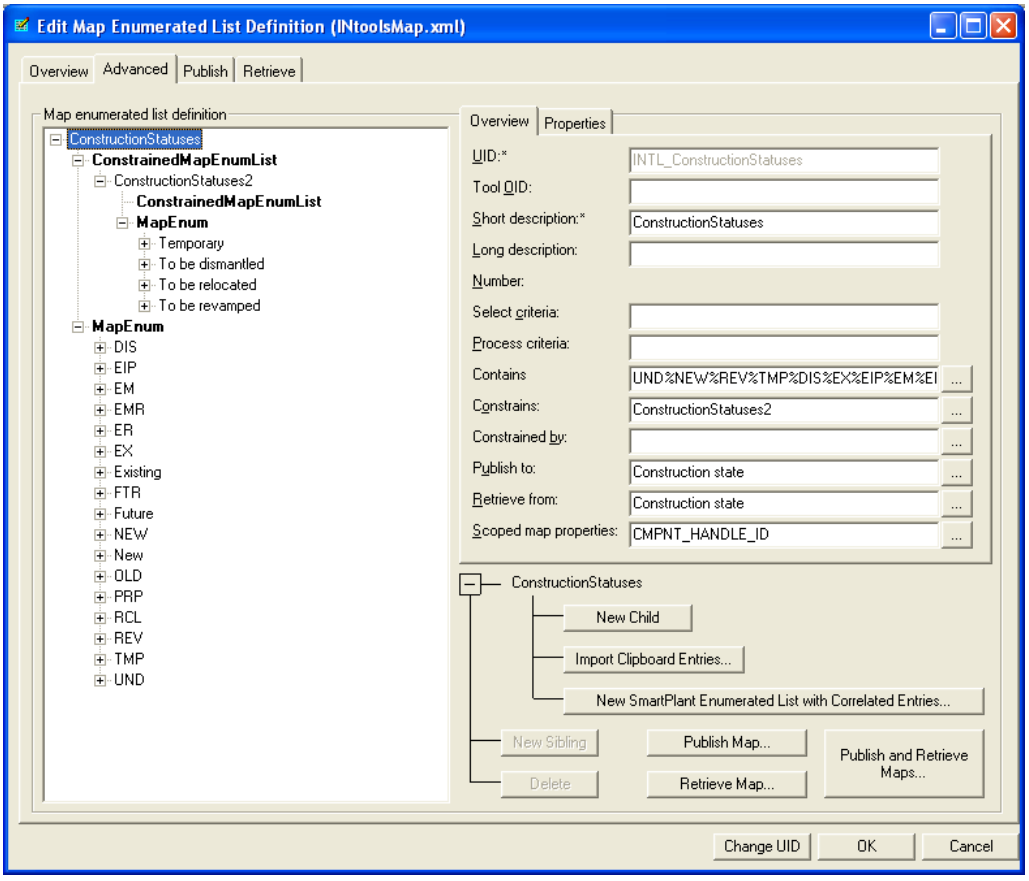
Map Enumerated List and Map Unit of Measure List Mapping

Tool map enumerated lists and map unit of measure lists need to be mapped into (for publish) and out of (for retrieve) the SmartPlant schema. This is necessary for tool map properties that are scoped by an enumerated list or a unit of measure list to properly exchange their values with other applications.

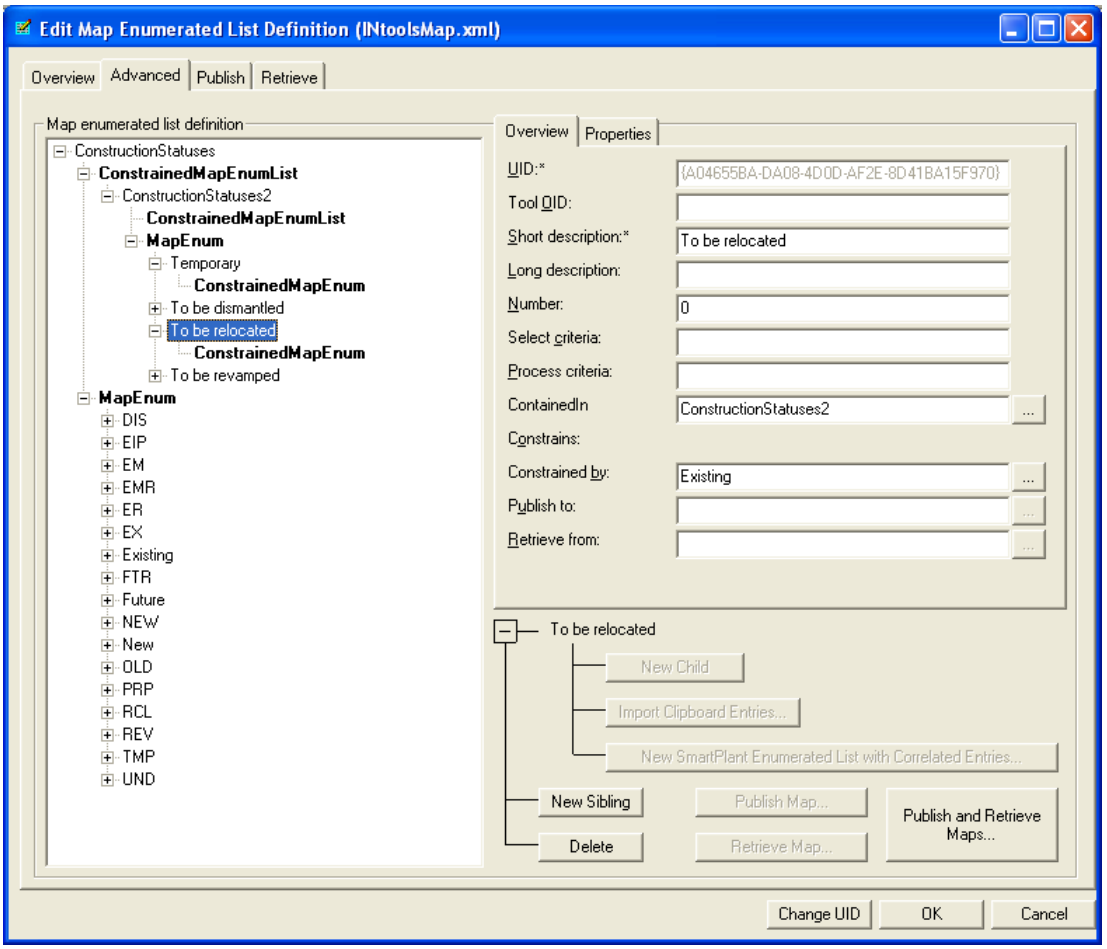
Like map classes and their map properties, there are also two methods by which map enumerated lists and their map enumerations or map unit of measure lists and their map units of measure can be mapped. One method is to define the mapping using separate forms activated from the **Overview** tab, and the other method is to use the **Publish** and **Retrieve** tabs. Before you begin mapping map enumerated lists and map unit of measure lists, you need to understand the map schema model that is used for modeling map enumerated lists and map enumerations.

Defining Map Enumerations and Their Relationships

Using the **Advanced** tab of the **Edit** window for map enumerated lists (SPMapEnumListDef), you can view and modify the complete definition of the map enumerated list. The **Advanced** tab for the ConstructionStatuses enumerated list appears in the following graphic. The MapEnumListMapEnum relationships between the map enumerated list ConstructionStatuses and its map enumerations are depicted in the **Contains** box, and the MapEnumListMapEnumList relationship between ConstructionStatuses and ConstructionStatuses2 is shown in the **Constrains** box.



If you select a node from the tree view on the left, the options on the right update accordingly. For example, selecting the **To be relocated** node from the tree view looks like the following. The MapEnumMapEnum relationship between **To be relocated** and **Existing** appears in the **Constrained by** box.



Several buttons in this window can be used to create new map enumerations, including **New Child**, **New Sibling**, and **Import Clipboard Entries**. If you create a new second level entry, such as **To be relocated**, it should be related to the appropriate first level entry using the **Constrained by** field.

NOTE This same dialog box is used for map unit of measure lists. However, all map unit of measure lists are one level. Therefore, the **Constrains** and **Constrained by** options should never be used for map units of measure.

Mapping Using the Separate Mapping Forms

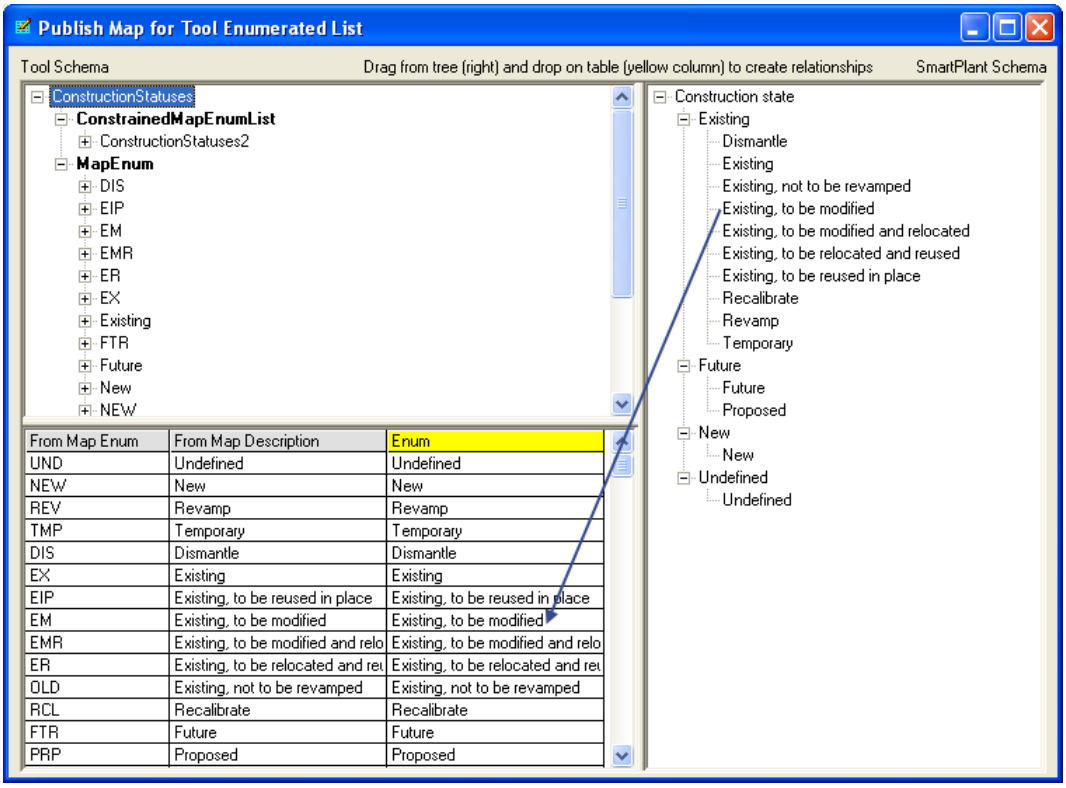
On the **Advanced** tab, there are several buttons that can be used to define the mapping between map enumerated lists or unit of measure lists and the corresponding SmartPlant enumerated lists or unit of measure lists.

You can use the **New SmartPlant Enumerated List with Correlated Entries** button to create a new SmartPlant enumerated list of the same name and with the same entries and structure as exists for the map enumerated list. At the same time, the map enumerated list is mapped to the

new SmartPlant enumerated list, and each of the map enumerations gets mapped to the corresponding SmartPlant enumerations. While this command makes the whole mapping process very simple, you should use it only when there is no existing list in the SmartPlant schema that can be used instead. Creating a new SmartPlant enumerated list when a suitable one already exists introduces ambiguity into the SmartPlant schema.

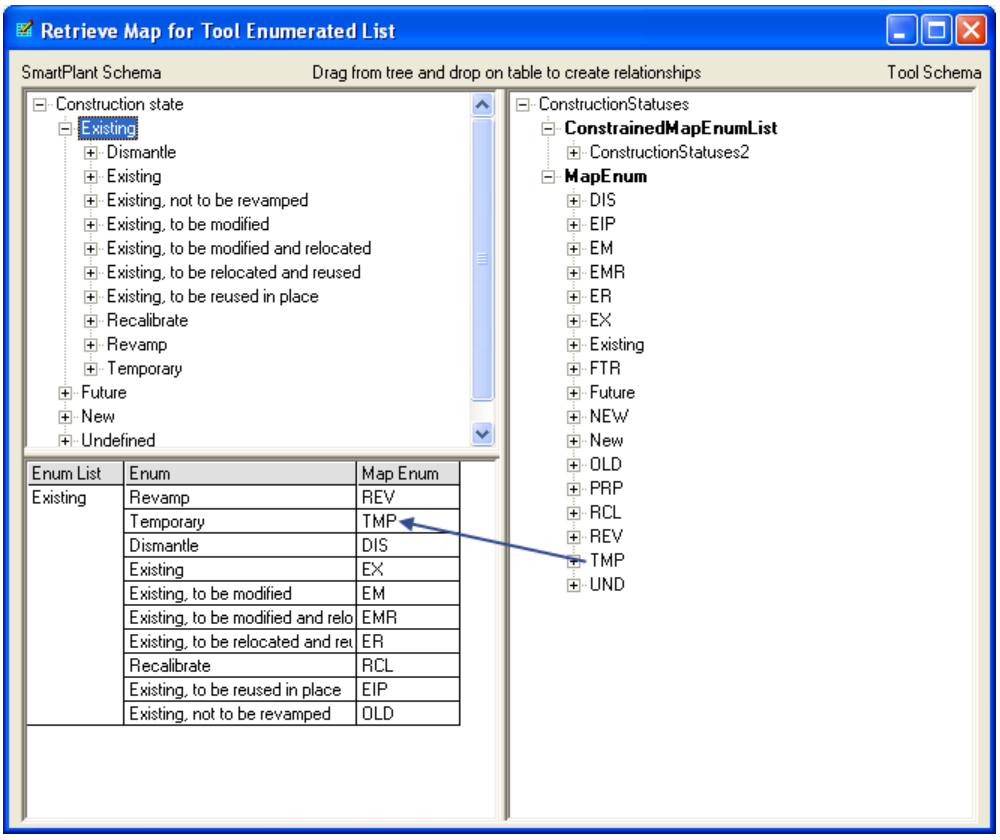
The **Publish Map**, **Retrieve Map**, and **Publish and Retrieve Maps** buttons display separate windows that can be used to define the mapping. All of these windows work by dragging a node from a tree view on one side of the window and dropping it in the appropriate column of the table on the other side of the window.

To establish a map relationship using the **Publish Map for Tool Enumerated List** window, you can drag an enumeration from the SmartPlant schema tree control on the right and drop it in the **Enum** column in the table for the appropriate map enumeration.

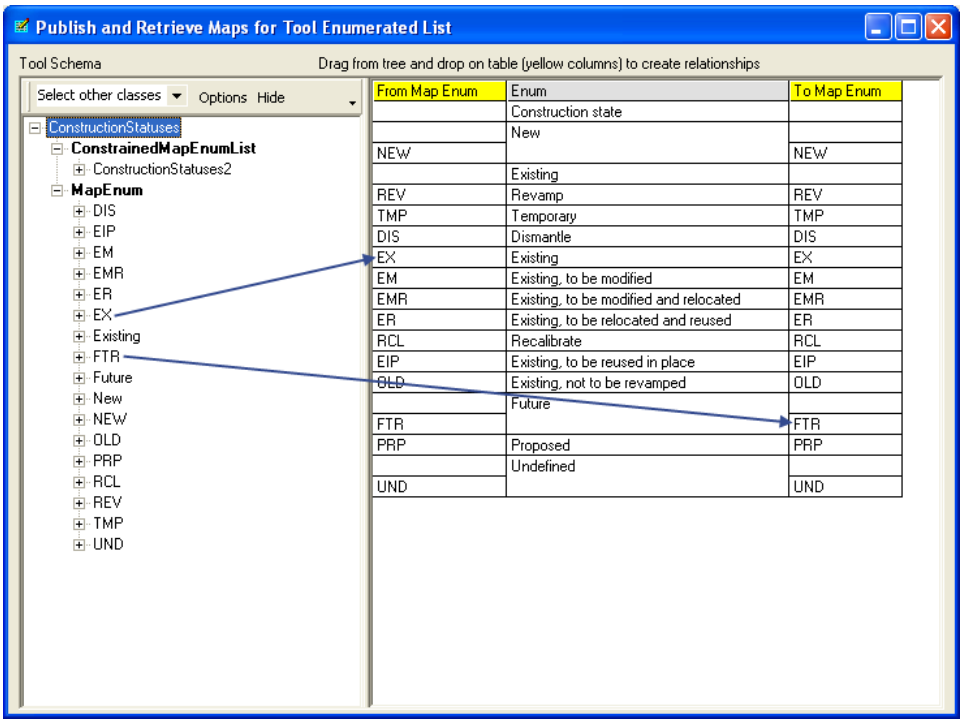


From Map Enum	From Map Description	Enum
UND	Undefined	Undefined
NEW	New	New
REV	Revamp	Revamp
TMP	Temporary	Temporary
DIS	Dismantle	Dismantle
EX	Existing	Existing
EIP	Existing, to be reused in place	Existing, to be reused in place
EM	Existing, to be modified	Existing, to be modified
EMR	Existing, to be modified and relo	Existing, to be modified and relo
ER	Existing, to be relocated and reu	Existing, to be relocated and reu
OLD	Existing, not to be revamped	Existing, not to be revamped
RCL	Recalibrate	Recalibrate
FTR	Future	Future
PRP	Proposed	Proposed

To create a map relationship using the **Retrieve Map for Tool Enumerated List** window, drag a map enumeration from the tool schema tree on the right and drop it in the **Map Enum** column in the table for the appropriate enumeration.



To create a publish map relationship using the **Publish and Retrieve Maps for Tool Enumerated List** window, drag a map enumeration from the tool schema tree and drop it in the **From Map Enum** column in the table. To create a retrieve map relationship, drag a map enumeration from the tool schema tree and drop it in the **To Map Enum** column.



Mapping Using the Publish and Retrieve Tabs

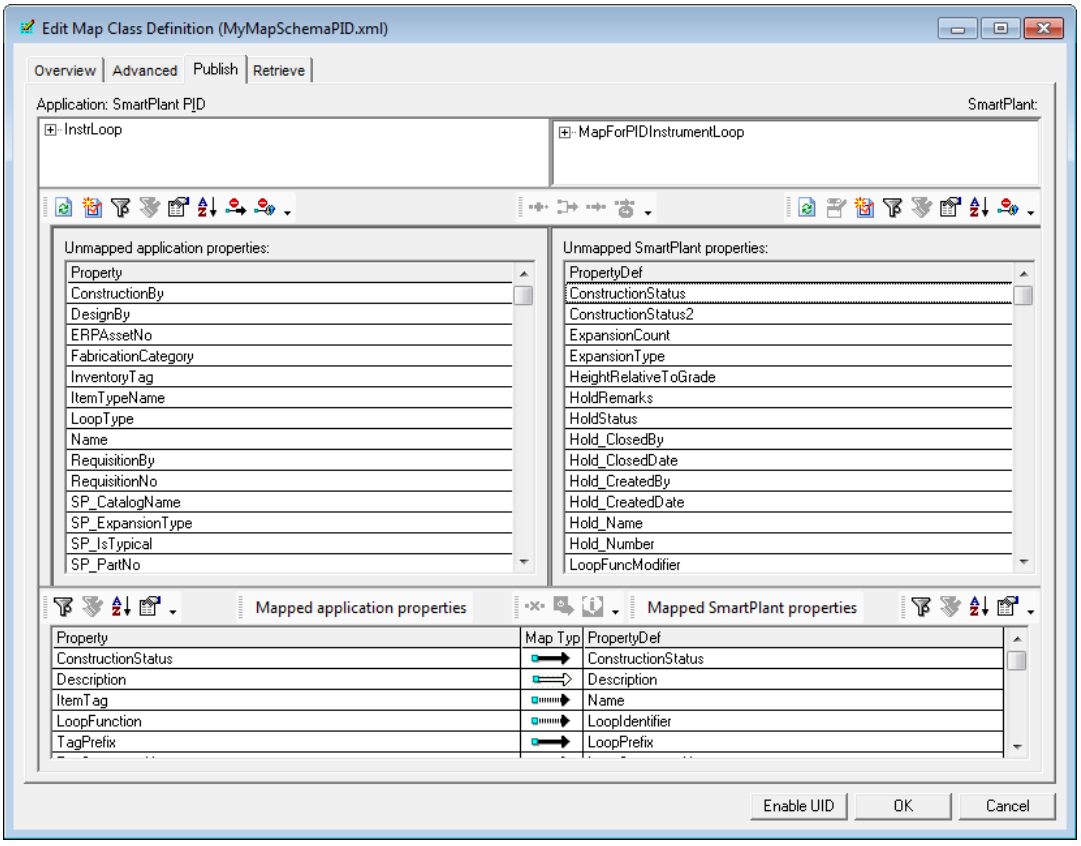
The **Publish** and **Retrieve** tabs on the **Edit** window for map enumerated lists and map unit of measure lists can also be used to define the mapping between the map enumerated lists or map unit of measure lists and the corresponding SmartPlant enumerated lists or unit of measure lists. These tabs are similar to the corresponding tabs for defining the mapping for a map class and its map properties.

The tree views at the top of the window display the map enumerated list and the corresponding enumerated list. Clicking a node in one of the tree views either filters the content displayed in the lower tables or selects a row in one of these tables. The toolbars on the left and right sides control the display of the table beneath them.

Selecting a row from the **Unmapped application enumerations** table and a row from the **Unmapped SmartPlant enumerations** table, and then clicking **Map** establishes a map relationship between the selected enumerations.

Selecting one or more rows from the bottom table and then clicking **Unmap** results in the deletion of those map relationships.

Selecting one or more rows from the bottom table and then clicking **Enable/Disable** disables map relationships that are enabled, or enables map relationships that are disabled.



See Also

Tool Map Schemas (on page 259)

Defining Authoring Tool Mapping

Default publish and retrieve mapping is delivered with the integrated Intergraph SmartPlant Enterprise and AspenTech products, including Smart 3D, SmartPlant Electrical, SmartPlant Engineering Manager, SmartPlant Instrumentation, SmartPlant P&ID, and Aspen Basic Engineering. However, as you configure your integrated environment, you may need to modify the delivered authoring tool mapping to meet your integration requirements.

Reasons to Modify Mapping

Changes to the authoring tool mapping for a particular application may be required for a variety of reasons:

- A class, property, enumerated list, enumerated list entry, unit of measure list, or unit of measure that exists in both the authoring tool map schema and the SmartPlant schema is not mapped.
- A class, property, enumerated list, enumerated list entry, unit of measure list, or unit of measure that exists in the authoring tool map schema does not exist in the SmartPlant schema and therefore is not mapped.

- A class, property, enumerated list, enumerated list entry, unit of measure list, or unit of measure that exists in the SmartPlant schema is not mapped in the tool map schema.
- A class, property, enumerated list, enumerated list entry, unit of measure list, or unit of measure that exists in the authoring tool is not mapped correctly for your needs.
- A property, enumerated list, enumerated list entry, unit of measure list, or unit of measure that your company needs to publish or retrieve does not exist in the authoring tool, the tool map schema, or the SmartPlant schema.

You may also need to create the mapping for a new authoring tool that you want to integrate. All authoring tools must have an authoring tool adapter, which is a small piece of authoring tool software, to be able to integrate.

Mapping Process Overview

Once the objects that you want to map exist in both the authoring tool map schema and the SmartPlant schema (whether they are created manually or automatically by synchronize), you are ready to define the publish and retrieve mapping for those objects. In the Schema Editor, you can map tool class definitions and their properties, enumerated lists, and unit of measure (UoM) lists.

Mapping for tool properties is performed by opening the map class definition that expose those map properties. The Schema Editor supports one-to-one, one-to-many, many-to- one, and many-to-many mapping for property definitions between tool objects (described by the tool map schema) and SmartPlant objects (described by the SmartPlant schema).

Mapping that involves many tool objects on one end of the relationship is handled by defining map edge definitions in the tool map schema. Map edge definitions define navigation across relationship edges in the tool map schema and are used to get from one tool object to related specific tool objects.

Mapping that involves multiple SmartPlant objects is done using edge definitions in the SmartPlant schema. Edge definitions define navigation across relationship edges in the SmartPlant schema and are used to get from one SmartPlant object to related SmartPlant objects.

Mapping for map enumerated lists or map UoM lists is defined from the map enumerated list or map UoM list. All mapping for enumerated lists and UoM lists is one-to-one.

NOTES

- The authoring tool adapter determines the objects that you can add to your authoring tool map schema and then map for publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.
- You can find individual mapping examples for authoring tools in the *Authoring Tool Mapping Examples* section of the *SmartPlant Mapping User's Guide*.

Common Mapping Scenarios

The most common types of authoring tool mapping scenarios, and those that are covered in this document, include the following:

- Extending an existing enumerated list in the SmartPlant schema and the authoring tool and mapping
- Adding a new string property to the SmartPlant schema and the authoring tool and mapping

- Adding a new property with an enumerated list to the SmartPlant schema and the authoring tool and mapping

See Also

SmartPlant Mapping Overview (on page 258)
Map Publish and Retrieve Properties in a Graphical View (on page 414)
Work Process for Mapping for Tools without Metadata Adapters (on page 427)
Add Objects to a Tool Map Schema (on page 429)
Before You Begin Mapping for Tools with Metadata Adapters (on page 328)
Compare a Tool Map Schema to the SmartPlant Schema (on page 385)
Synchronize Tool Metadata with the Tool Map Schema (on page 328)
Mapping with Metadata Adapters (on page 291)

Mapping with Metadata Adapters

Prior to version 2007 (3.8) of SmartPlant Enterprise, users who wanted to add an object in the authoring tool and then map that object to a SmartPlant schema object had to manually add the object to the tool's metadata and then manually add the object to the corresponding tool map schema, while adhering to the specific rules for the application. In previous versions, the user was also required to explicitly identify the locations for the schema and tool schema files.

With version 2007, metadata adapters were added for the SmartPlant applications. The addition of these metadata adapters removes the need for the user to know and adhere to the specific rules for each application for a change to successfully work. Each application's metadata adapter hides this complexity from the user, making application changes much simpler. The metadata adapters also manage both the application's metadata and the map schema file. Changes that are made are applied to both, thereby eliminating an extra step in the process.

In order to simplify the process of managing authoring tool map schemas, metadata adapters have been developed for the following SmartPlant applications:

- Smart 3D
- SmartPlant Instrumentation
- SmartPlant Electrical
- SmartPlant P&ID

Metadata adapters are authoring tool software that make two-way communication between the Schema Editor and the authoring tools possible for mapping. Metadata adapters allow the Schema Editor to connect to the tool database and synchronize the tool map schema with the tool metadata by updating one or the other, depending on user selections. Metadata adapters can be developed for third-party applications. For more information, see the *SmartPlant Metadata Adapter Reference Guide*, delivered with the SmartPlant Foundation programming documentation.

In previous releases, all authoring tools were required to add properties, enumerated lists, and enumerated entries in the tool database, add them to the tool map schema and the SmartPlant schema in the Schema Editor, and then map, each in separate steps. Metadata adapters streamline the tool map schema extension and mapping work process.

For properties, enumerated lists, and enumerated entries, metadata adapters allow you to enter the object in the authoring tool and have it automatically added to the tool map schema during synchronization in the Schema Editor. You can also enter enumerated lists and their entries to the tool map schema in the Schema Editor, and the metadata adapter automatically adds those

same objects to the metadata in the tool database. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

Metadata adapters can simplify the process of editing tool map schemas by automatically handling special requirements of the corresponding authoring tool adapter. For example, the SmartPlant P&ID adapter requires that new enumerated list definitions have a UID of SP_<Codelist_Index>, for example, SP_12001. The metadata adapter can supply this UID automatically so that the user is not required to look up the codelist index in the Enumerations database table for the SmartPlant P&ID Data Dictionary user in the SmartPlant P&ID database.

Metadata adapters also automatically locate the tool map schema on the file system, eliminating the need for the user to specific the location.

Authoring Metadata Adapter Mapping Terminology

Outside the Schema Editor, SmartPlant Enterprise authoring tools use various internal applications to edit their metadata. Each tool also has one or more tool map schemas that contain the mapping between the authoring tool metadata and the SmartPlant schema. Before you begin mapping, you need to understand the meaning of metadata in each tool, how that metadata is edited internally for each tool, and the name of the tool map schema for each tool.

Authoring Tool	**Tool Metadata Terminology**	**Method to Define Metadata**	**Tool Map Schema**
Smart 3D	Catalog Schema database	Bulkload into Catalog database or edit in the Catalog task	DesignBasis_map.xml (retrieve) SP3DPublishMap.xml (publish) SP3DToEFWClassMap.xml (design basis to 3D objects)
SmartPlant Electrical	data dictionary	Edit in Data Dictionary Manager	SPELMap.xml
SmartPlant Instrumentation	tables, User-Defined Fields (UDFs), User-Defined Tables (UDTs)	Add entries to select lists using the **Tables** menu in all modules. Associate new properties with UDFs in the Administration module.	INtoolsMap.xml
SmartPlant P&ID	data dictionary	Edit in Data Dictionary Manager	SPPIDDataMap.xml

For more information about authoring tool mapping concepts, see the *SmartPlant Mapping User's Guide*.

Rules for Modifying and Saving Metadata with Metadata Adapters

Each tool's metadata adapter has specific rules that determine what types of objects you can add and update in the Schema Editor. These same rules typically apply to the native tools that authoring tool users use, such as Data Dictionary Manager for SmartPlant P&ID, to edit the tool metadata. For more information about mapping rules for each authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.

When you edit tool metadata using the Schema Editor, changes to the tool metadata is saved at different times depending on the authoring tool that you are using. Most authoring tools, except SmartPlant Instrumentation, save the changes to the tool metadata when you save the tool map schema file using the **File > Save > Tool Schema File** command in the Schema Editor. This allows you to cancel your changes when you are adding new tool properties, enumerated lists, or enumerated entries before you save the tool map schema.

SmartPlant Instrumentation, however, automatically saves your metadata changes to the SmartPlant Instrumentation database as you make your changes in the Schema Editor. Therefore, you should be very careful to make sure that you know what changes you need to make before you begin.

For specific details about what types of objects that you can edit in the authoring tool via the metadata adapter and how save works for the tool's metadata adapter, select a tool in the **Engineering Tool Options** dialog box and then click the **Details** button.

Smart 3D Metadata Adapter

The following section describes the rules and behavior for the Smart 3D metadata adapter.

Metadata Save Behavior

Metadata is saved when the tool schema is saved.

Rules for Application Classes

For this release, Smart 3D does not support adding new classes to tool metadata using the metadata adapter. Adding new SPMapClassDefs for Classes only creates a map object in the tool map schema. However, you can create new interfaces (IsInterface = True) for an existing class using the metadata adapter.

Rules for Application Properties

Smart 3D does not support adding properties to system-defined interfaces (also referred to as packages). However, users can add new properties on existing interfaces in the user-defined package.

Rules for Application Enumerations

- Adding new enumerated lists (code lists) is supported in the tool.
- Publish mapping is not required for enumerated lists (code lists) because the Smart 3D adapter maps automatically based on the enumerated list entry number. However, the number must match in the SmartPlant schema, the Smart 3D tool map schema, and the tool metadata.

Rules for Application UoMs

Changes to SPMapUoMListDefs will only be made in the tool map schema (SP3DPublishMap.xml). The tool database will not be updated with these changes.

Rules for Application Relationships

Adding and deleting SPMapRelDefs is supported in the tool. However, adding and deleting SPMapRelDefs to certain objects defined in the system package are not allowed in the tool.

Rules for Application Edges

Any changes to SPMapEdgeDefs will only be made in the tool map schema (SP3DPublishMap.xml). Tool database will not be updated with these changes.

SmartPlant Electrical Metadata Adapter

The following section describes the rules and behavior for the SmartPlant Electrical metadata adapter.

Metadata Save Behavior

Metadata is saved when the tool schema is saved.

Rules for Application Classes

- Updating existing classes is allowed.
- Creating new classes is not allowed.
- Deleting is not allowed for classes.

Rules for Application Properties

- Updating existing properties is allowed.
- Creating new properties is allowed.
- Deleting new properties before they are saved is allowed. Deleting properties after they are saved is not allowed.

Rules for Application Enumerations

- Updating existing lists is allowed.
- Creating new lists is allowed.
- Deleting new lists before they are saved is allowed.
- Adding new entries to a list is allowed.
- Edit entries is allowed.
- Deleting entries is allowed. Deleted entries become unavailable but are not physically removed from the database.
- Deleting a saved list is not allowed.

Rules for Application UoMs

UoMs are not currently synchronized.

Rules for Application Relationships

Relationships do not exist in the metadata.

Rules for Application Edges

Edges do not exist in the metadata.

SmartPlant Instrumentation Metadata Adapter

The following section describes the rules and behavior for the SmartPlant Instrumentation metadata adapter.

Metadata Save Behavior

Saves changes immediately to the tool database.

Rules for Application Properties

- Creating properties inside the authoring tool is only allowed when the properties are created as user-defined fields (UDFs).
- Deleting a property that you have already synchronized between the tool metadata and the map schema file is not allowed because the software cannot determine whether data values exist for that property.

Rules for Application Enumerations

Defining a new enumerated list for a property is not allowed. Properties can be associated with existing enumerated lists.

SmartPlant P&ID Metadata Adapter

The following section describes the rules and behavior for the SmartPlant P&ID metadata adapter.

Metadata Save Behavior

Metadata is saved when the tool schema is saved

Rules for Application Classes

- Updating existing classes is allowed.
- Creating new classes is not allowed.
- Deleting is not allowed for classes.

Rules for Application Properties

- Updating existing properties is allowed.
- Creating new properties is allowed.
- Deleting new properties before they are saved is allowed. Deleting properties after they are saved is not allowed.

Rules for Application Enumerations

- Updating existing lists is allowed.
- Creating new lists is allowed.
- Deleting new lists before they are saved is allowed.
- Adding new entries to a list is allowed.
- Editing entries is allowed.
- Deleting entries is allowed. Deleted entries become unavailable but are not physically removed from the database.
- Deleting a saved list is not allowed.

Rules for Application UoMs

UoMs are not currently synchronized.

Rules for Application Relationships

Relationships do not exist in the metadata.

Rules for Application Edges

Relationships do not exist in the metadata.

Tools without Metadata Adapters

Some authoring tools, such as Aspen Basic Engineering, SmartPlant Engineering Manager, and SmartPlant Materials, do not deliver metadata adapters. For these tools, you can still add data to the tool's database, tool map schema, and the SmartPlant schema and then define the mapping. However, each step must be performed manually. For more information, see *Work Process for Mapping for Tools without Metadata Adapters* (on page 427).

See Also

Before You Begin Mapping for Tools with Metadata Adapters (on page 328)

Connecting to a SmartPlant Foundation Plant

Modifying the SmartPlant schema for integration, including performing mapping for tools with metadata adapters, requires the following:

- Check out the SmartPlant schema CMF file from the SmartPlant Foundation Desktop Client.
- Make sure your scope is set to the plant, right-click the new revision of the CMF file in SmartPlant Foundation, and click **Launch Schema Editor**. This provides the Schema Editor with the appropriate plant and SmartPlant Foundation site for connection.
- Select the schema version or versions that you want to view in the **Specify configurations to display** window of the **Active Configurations** dialog box.
- Click the **Advanced** button on the **Active Configurations** dialog box to view the **Set Configurations for new objects** window, where you can specify what version or versions of the schema changes should be written to. For example, to modify the 2016 schema, select the 2016 configuration for new objects.

  NOTE By default, all configurations are checked to allow forward and backward compatibility between schema versions when creating new objects or modifying existing objects.
- Click **File > SmartPlant > Edit Engineering Tool Options** to activate the authoring tool metadata adapters.

For every application registered against that plant, the software checks to see if a Tool object exists for that application in this schema file. If not, an information message indicates the need for the Tool object. If the software finds a Tool object, it attempts to launch the metadata adapter specified in that Tool object (if the metadata adapter doesn't get activated, then an information message indicates this failure).

For every ToolSchema object that traces back to a Tool object in the schema and for which the metadata adapter for that Tool was successfully activated, the software adds an entry for that authoring tool to the list on the **Engineering Applications** dialog box. Selecting an authoring tool entry in the list enables the check boxes under **Startup options** for that application/map schema combination.

Selecting the **Load map schema** check box indicates that the software should open and parse the map schema file. Selecting **Connect to application schema** indicates that software should extract the application's metadata, compare it with that for the loaded map schema, and perform the synchronization process. By default, this synchronization process displays a window that shows the differences resulting from the comparison of the application's metadata with the map schema file. Selecting the **Automatically synchronize schemas** check box bypasses this user interface and uses the default (software selected) choices for synchronizing the schema.

Although **Load map schema** and **Connect to application schema** are separate check boxes, the current tool metadata adapters require that either both or neither of these check boxes are selected. Therefore, selecting **Load map schema** also selects **Connect to application schema**. This is referred to as connected mode, which is currently required by all tool metadata adapters.

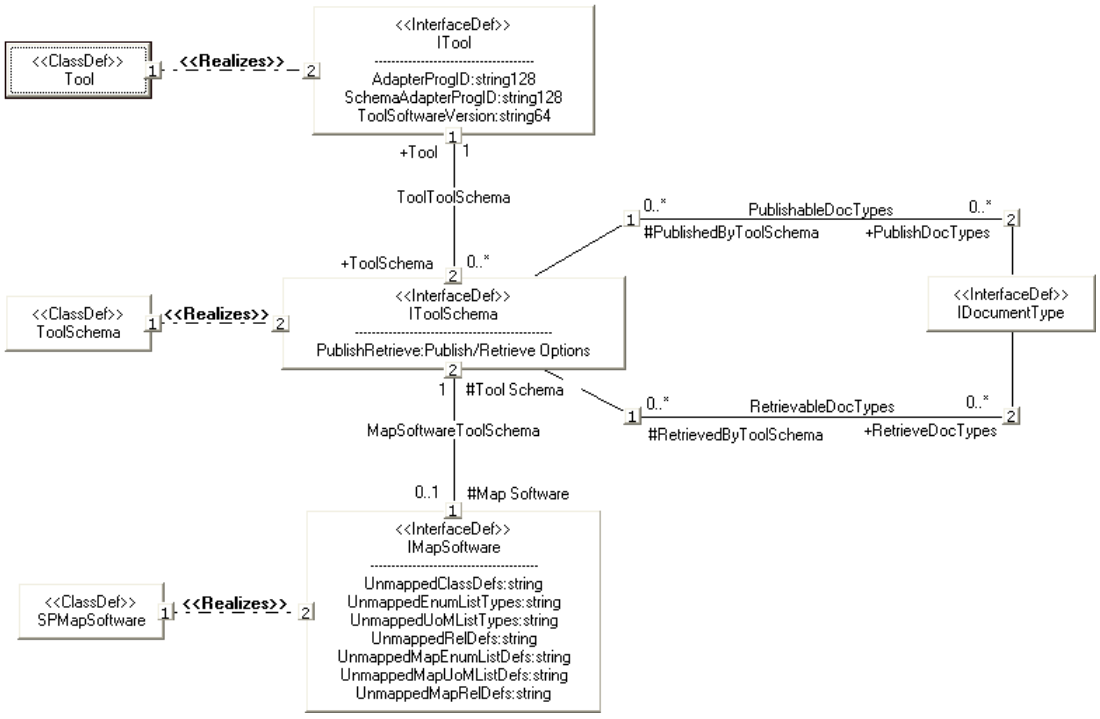
For more information about connecting to SmartPlant Foundation and synchronizing tool map schemas with tool metadata, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

Tool Objects Required for Metadata Adapters

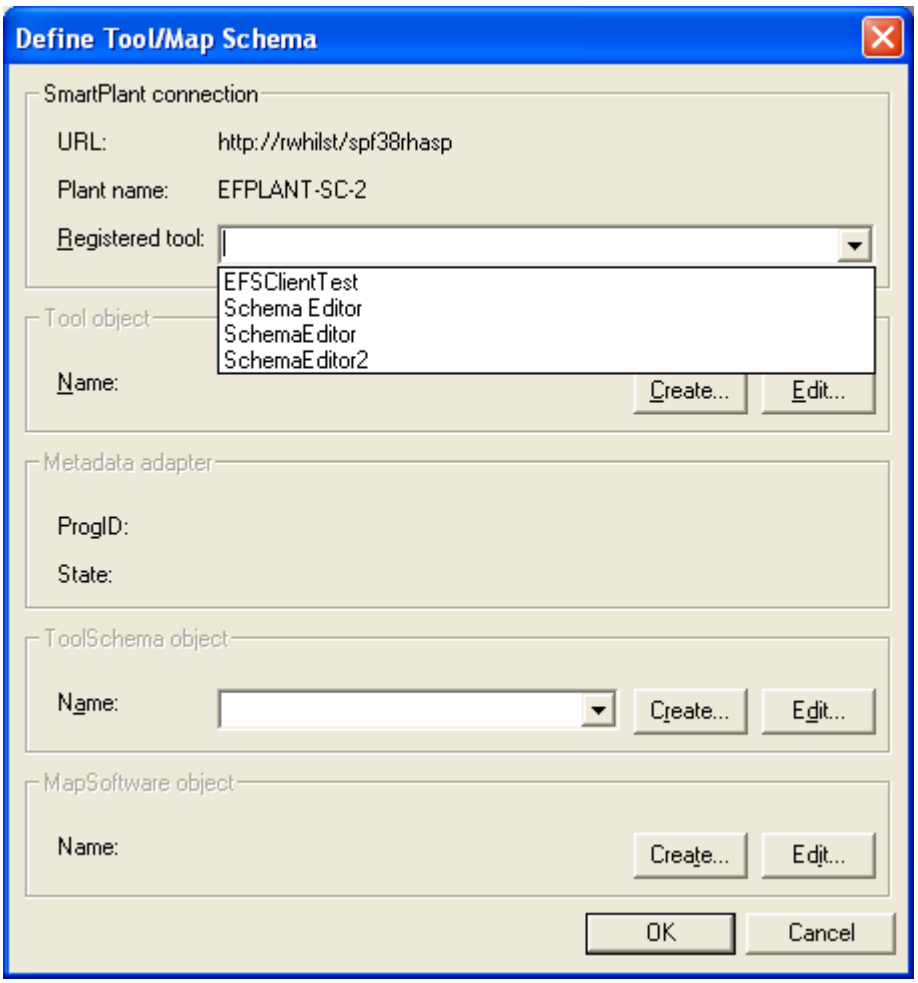
For connecting to SmartPlant Foundation to work for a particular application, the following objects must exist in the SmartPlant schema or the tool map schema:

- Tool (SmartPlant schema object) - Defines the existence of the application in the context of the SmartPlant schema. Sets certain properties, such as the metadata adapter progID, needed for proper functional behavior.
- ToolSchema (SmartPlant schema object) - Defines the existence of a tool map schema in the context of the schema.
- SPMapSoftware (tool map schema object) - Ties the tool map schema to the corresponding ToolSchema object in the SmartPlant schema.

The following UML diagram illustrates the relationships among these objects and the properties exposed by the interfaces for these objects:



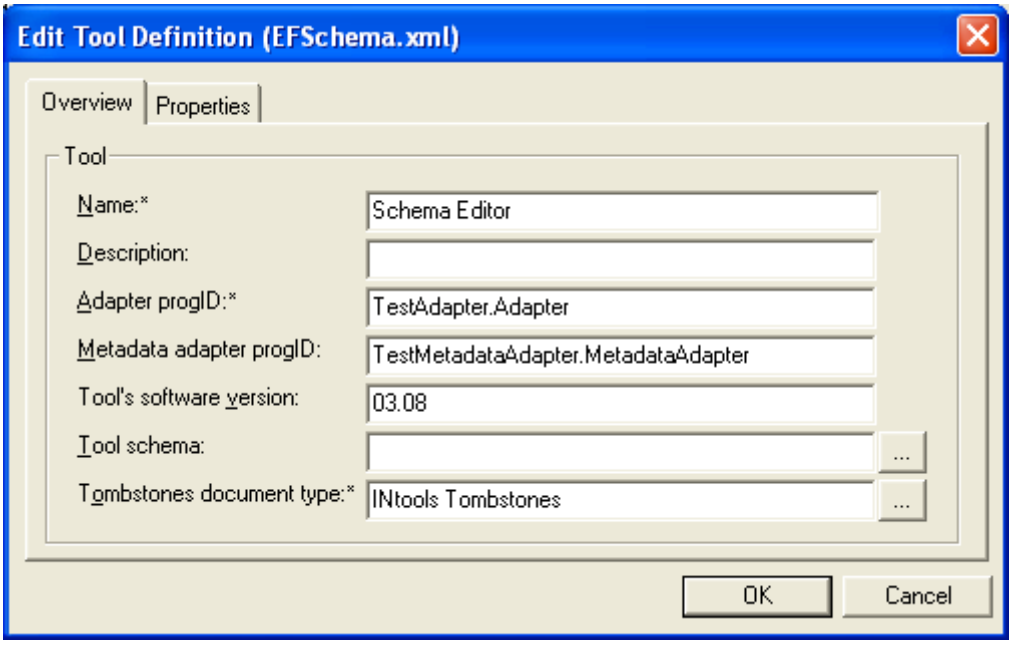
If the Tool, ToolSchema, or SPMapSoftware objects or the corresponding relationships are missing for a particular application, then that application is not available via the **Edit Engineering Tool Options** command. If an application is not available due to one or more of these objects being missing, use the **Add Application/Map Schema** button on the **Application Metadata** tab of the **Workflows** dialog box to define the objects and relationships in the **Define Tool/Map Schema** dialog box.



Selecting a tool from the **Registered tool** list displays whatever information has been defined for that application.

Tool Object

If a Tool object exists for the application, then the **Name** for that Tool object appears under **Tool object**. If the Tool object does not exist, you can create one by clicking **Create** button under **Tool object** to display the **Edit Tool Definition** dialog box.



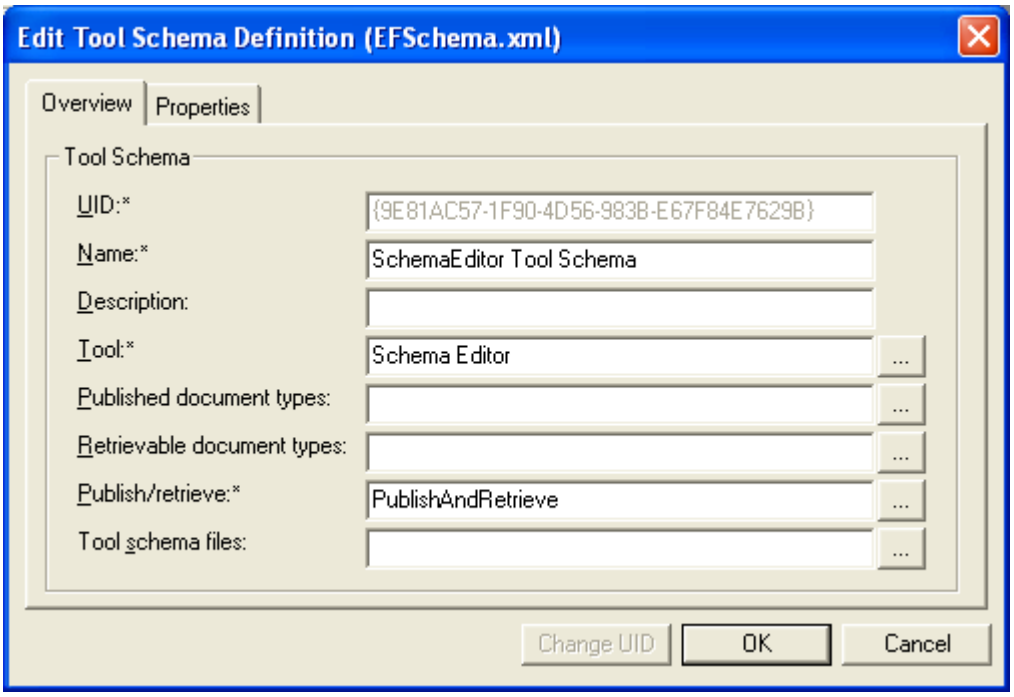
If the Tool object already exists for the specified application, the progID specified for the metadata adapter is used to activate the specified DLL/software class. Alternatively, if the user creates a Tool object for the specified application, the software attempts to activate the metadata adapter when the user clicks **OK**.

If the metadata adapter is successfully activated, the **Status** text under **Metadata adapter** in the **Define Tool/Map Schema** dialog box reflects that the metadata adapter has been activated. If the specified metadata adapter cannot be activated, then an error message appears, and the **Status** text reflects this failure.

ToolSchema Object

If the Tool object is related to one or more ToolSchema objects, then the **Name** list under **ToolSchema** in the **Define Tool/Map Schema** dialog box lists these ToolSchema objects. Selecting one of these objects activates the **Edit** button, and, if the map schema for that application has been loaded, the software attempts to locate the SPMapSoftware object that is related to the ToolSchema object.

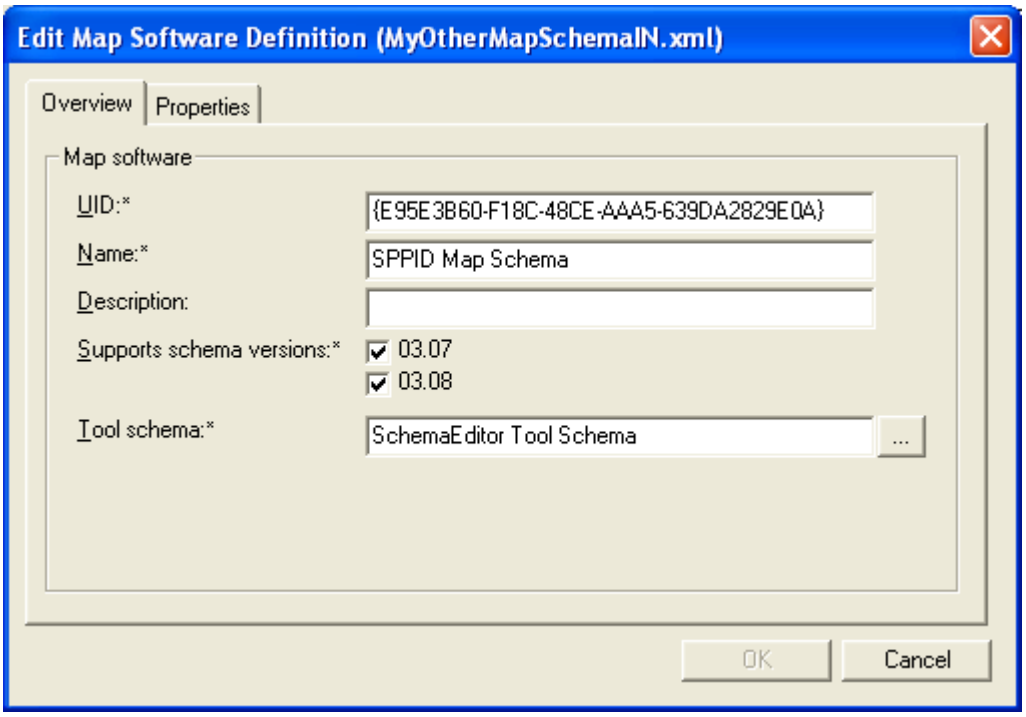
If no ToolSchema object exists, then the **Create** button can be used to create the ToolSchema object in the **Edit Tool Schema Definition** dialog box.



SPMapSoftware Object

To determine if the SPMapSoftware object exists, you can use the **File > SmartPlant > Edit Engineering Tool Options** command and select the desired application. When you click **Finish** after synchronization is completed, **Map Environment** that appears should include the selected application in the tree view. If expanding this application in the tree view in the **Map Environment** fails to show the data for that application, then the SPMapSoftware object is missing and needs to be created.

To create the missing SPMapSoftware object, click **Create** under **MapSoftware** in the **Define Tool/Map Schema** dialog box. The dialog box for creating/editing the SPMapSoftware object is shown in the following graphic:



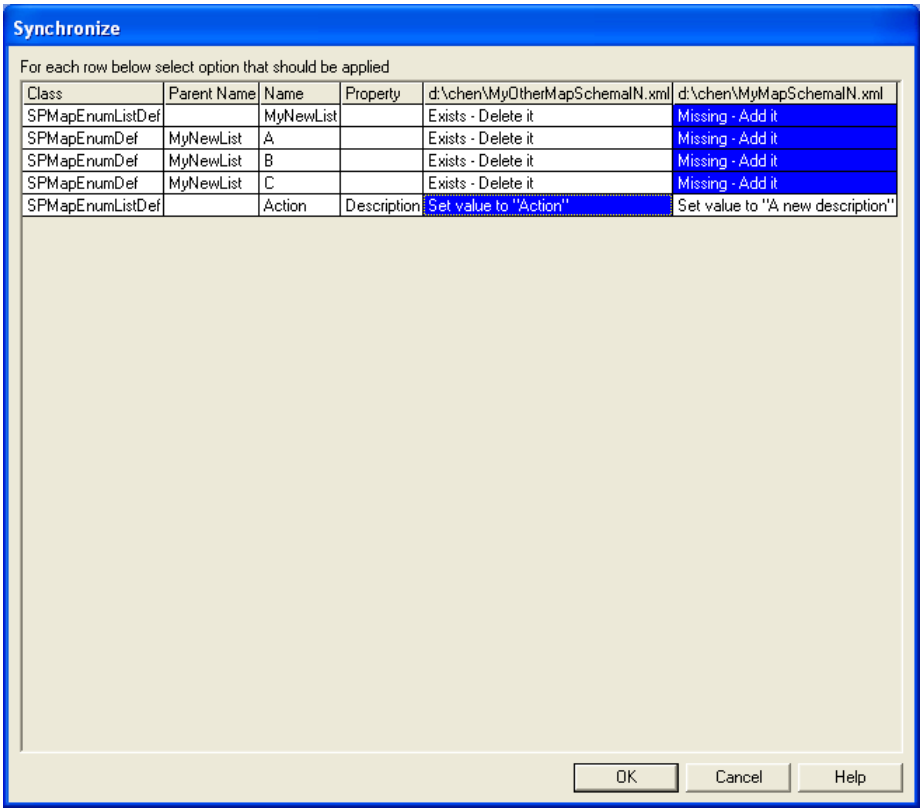
The Schema Editor, when working with application metadata adapters, relies on the version supported information in the SPMapSoftware object. If the SPMapSoftware object for a given application doesn't indicate support for the version of the schema being used, then a message will be displayed to the user warning him about proceeding.

See Also

Tool Map Schemas (on page 259)

Synchronizing Tool Map Schemas with Tool Metadata

After you click **OK** in the **Engineering Options** dialog box in the Schema Editor, the software goes through a process of synchronizing the application's metadata with the tool map schema file for each application/map schema you selected. The synchronization process involves doing an object by object comparison between the application metadata's objects and the tool map schema's objects. The software displays the results of this comparison in the **Synchronize** dialog box.



For every difference detected between the application's metadata and the tool map schema file, a row is displayed in this dialog box. The **Class** column shows the class for the difference. The **Parent Name** column identifies the parent object's name. For map enumerations, the parent object is the map enumerated list. For map units of measure, the parent object is the map unit of measure list, and for map properties, the parent object is the map class. For all other map objects, this column is blank.

The **Name** column shows the name of the object with the difference. If the difference is in a particular property value for the object, then the **Property** column shows the particular property that is different.

The **<Tool> Database** and **<Tool Map Schema Path>** columns show the particular actions that can be taken to eliminate the difference. Eliminating all differences results in the application's metadata and tool schema file being "synchronized". When the **Synchronize** dialog box appears, one of the two columns for each row is automatically selected. This is the default action that the software takes to eliminate the difference. The application's metadata adapter specifies the default actions for that application. Most of the time the software automatically

selects the option in the **<Tool Map Schema Path>** column to eliminate the difference. You can change the action to be taken to eliminate the difference by selecting the other column for a particular row. However, changes that are not allowed by the metadata adapter result in an error message and are prevented by the software.

When the appropriate changes are selected to eliminate the differences, you can click **OK**. The software performs the various actions that have been selected. You are not currently allowed to cancel synchronization.

For more information about synchronizing tool map schemas with tool metadata, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

Comparing Tool Map Schema with the SmartPlant Schema

Tracking the progress of mapping can be a challenge. Frequently, the mapping for a particular map class does not involve mapping all of its properties. Users very rarely use all of the available SmartPlant view definition properties for mapping when view definitions are used. Therefore, viewing the mapping for a map class definition shows map properties that have not been and should not be mapped, as well as view properties that have not been mapped to. With this visual feedback, it can be difficult to know when mapping is complete.

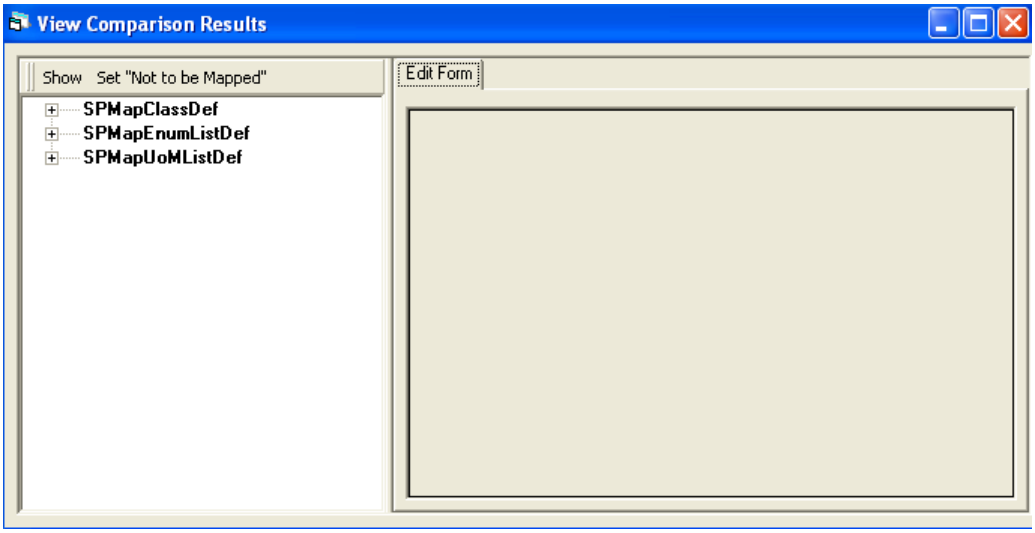
To facilitate in tracking the progress of mapping, a number of commands can be used. The **Set Application Properties Not to Be Mapped** button  can be used after all map property mapping for a particular map class is complete to remove properties from the unmapped control in the middle of the **Publish** and **Retrieve** tabs.

Likewise, when the map property mapping for a map class is complete, the **Set SmartPlant Properties Not to Be Mapped** button  can be used to indicate that the remaining unmapped-to SmartPlant properties are not to be mapped. This will remove them from the unmapped control in the middle of the **Publish** and **Retrieve** tabs.

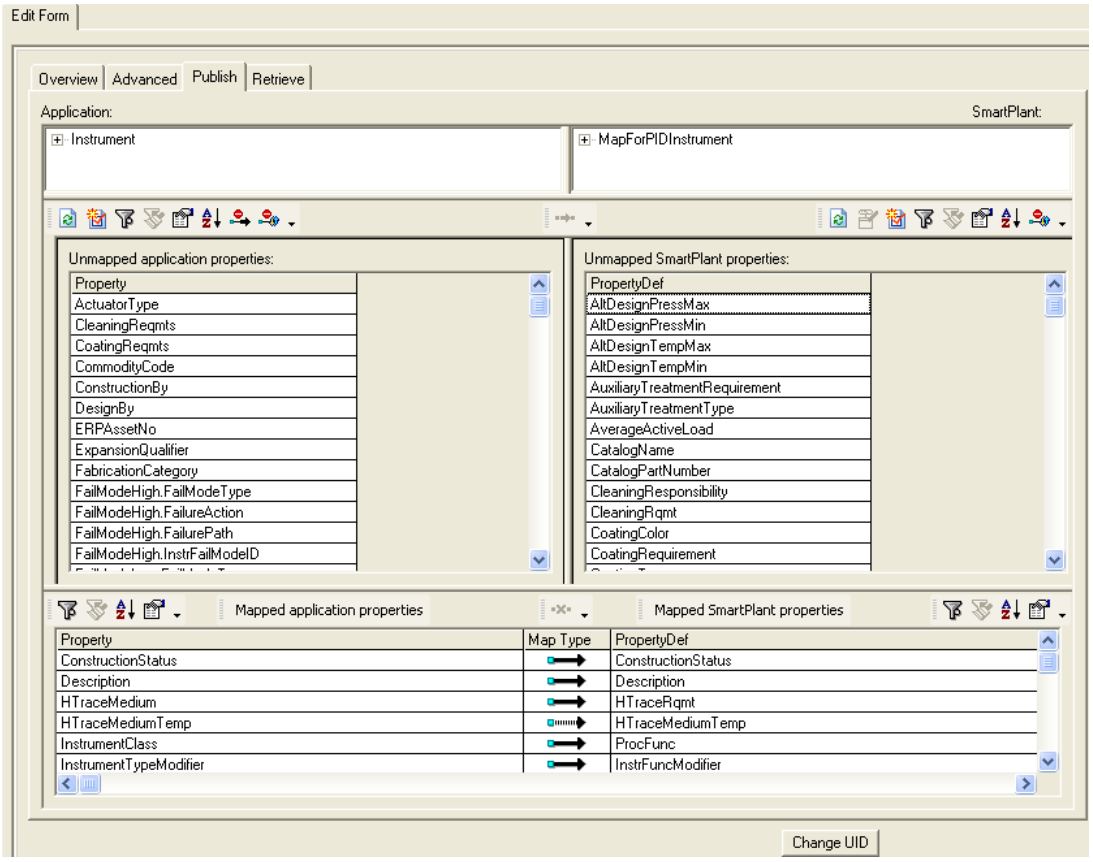
If there are no unmapped map properties except those flagged as not to be mapped and there are no unmapped-to-SmartPlant properties except those flagged as not to be mapped, then the map class is considered to be "consistent" with the SmartPlant schema. This consistency between the map schema and the schema is important because a comparison of the map schema with the schema treats "consistent" map classes as being the same as the corresponding SmartPlant object (view definition); therefore, these are not report as a difference for the comparison.

Likewise, map enumerated lists and map unit of measure lists that have either been fully mapped or for which all unmapped entries have been flagged as not to be mapped are reported as differences during comparison.

The **File > SmartPlant > Compare (Active Tool Schema) With Schema** command determines the map classes, map enumerated lists, and map unit of measure lists that have not been fully mapped (those that still have one or more unmapped objects that haven't been flagged as not to be mapped). These incompletely mapped objects appear in the **View Comparison Results** window.



When you expand one of the nodes in the **View Comparison Results** tree view, the objects of the selected type that have not been mapped at all appear in red, and those objects that have been partially mapped appear as blue. Selecting one of these objects display its **Edit Form** tab on the right side of the **View Comparison Results** window. You can use this tab to complete the mapping for that object.



When an object is completely mapped, the software removes it from the tree view in the **View Comparison Results** window. When all map objects have been fully mapped, the comparison should result in an empty tree view. Therefore, using comparison is a good way to track the progress of the publish mapping process.

After all the mapping has been completed for an application, only changes made in the application or changes made by another application that impact the application should impact the mapping. Changes made in this application can readily be resolved by connecting to SmartPlant Foundation and synchronizing the tool map schema with the tool metadata database.

For more information about comparing tool map schemas with the SmartPlant schema, see *Compare a Tool Map Schema to the SmartPlant Schema* (on page 385).

See Also

Tool Map Schemas (on page 259)

Viewing Authoring Tool Mapping

In the Schema Editor, you can see how the classes, properties, enumerated lists, and units of measure in a tool map schema map to the SmartPlant schema for publish and retrieve. By opening multiple tool map schemas, you can also see how data maps across authoring tools.

You can view mapping in a variety of ways in the Schema Editor:

- Tool-to-SmartPlant mapping in a UML view
- Point-to-point mapping in a drag-and-drop UML view
- Tool-to-tool mapping for classes and properties in a tabular view
- Tool-to-tool mapping for enumerated lists in a tabular view
- Tool-to-tool mapping for units of measurement in a tabular view

When you view mapping across authoring tools in a tabular view, you can export Microsoft® Excel spreadsheets containing the mapping. These spreadsheets allow you to sort and view the mapping using Excel's built-in tools.

NOTES

- Not all authoring tools externalize all of their mapping in tool map schemas or map files. Some mapping may be defined in the authoring tool adapter code. This kind of mapping does not appear in mapping spreadsheets generated in the Schema Editor.
- Spreadsheets containing the out-of-the-box mapping for the current releases of the authoring tools are delivered with the SmartPlant Foundation software. You can open the spreadsheets from SmartPlant Foundation by clicking **Help > Printable Guides** in the SmartPlant Foundation client or administration software, and then clicking the **SmartPlant Enterprise Mapping Spreadsheets** link to download the .zip file containing the Excel spreadsheets.
- You can also download these spreadsheets from Intergraph Smart Support. To download the spreadsheets, browse to the following web address, and type your Intergraph Smart Support user name and password: *http://smartsupport.intergraph.com* (*https://smartsupport.intergraph.com*). Click **View Downloads**, select **SmartPlant Foundation**, and then click **Technical Notes and Whitepapers**. Click the link to the mapping spreadsheets to download them to your computer.

See Also

View Tool-to-SmartPlant Mapping in a Viewable UML View (on page 308)
View Tool-to-Tool Mapping in a Editable UML View (on page 313)
View Tool-to-Tool Mapping for Enumerated Lists (on page 319)
View Tool-to-Tool Mapping for Classes and Properties (on page 316)
View Tool-to-Tool Mapping for Units of Measure (on page 322)

View Tool-to-SmartPlant Mapping in a Viewable UML View

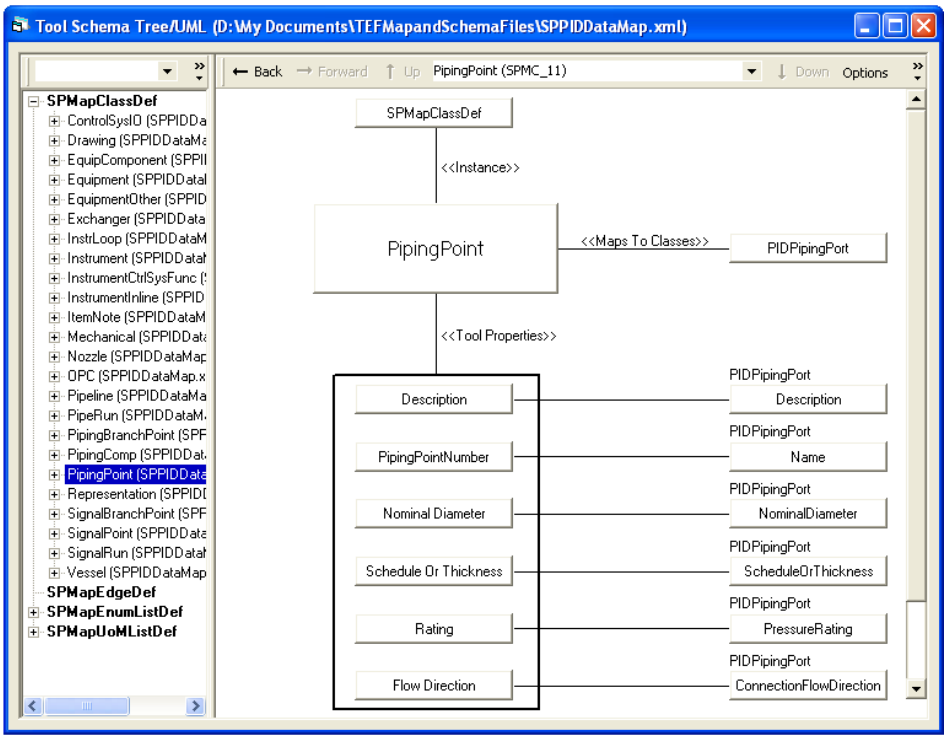
1. Open the two tool map schemas for which you want to view mapping. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.
2. Click **View > Tool Schema**.
3. Click **Options** on the **View Schema** dialog box.
4. On the **View** tab, click **Tree/UML**, and click **OK**.

   TIP To view the mapping for a tool class, expand the **SPMapClassDef** node in the tree view, and then click the class for which you want to view mapping.

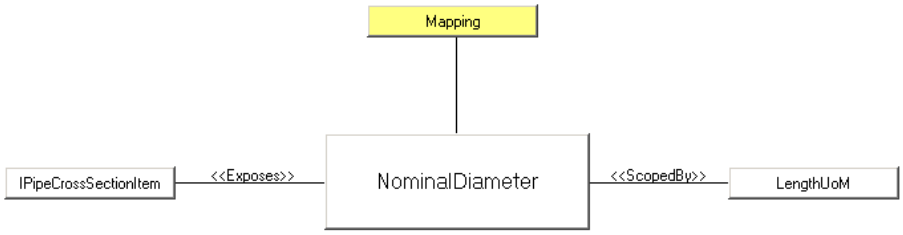


TIPS

- Under the class in the UML view, you can see a list of the tool map properties for the class.
- To the right of the map properties, you can see the classes and properties in the SmartPlant schema to which those properties are mapped for publish.
- If the tool map schema contains retrieve mapping for the map properties, the SmartPlant schema properties that map to the tool properties on retrieve appear on the left side of the UML view.

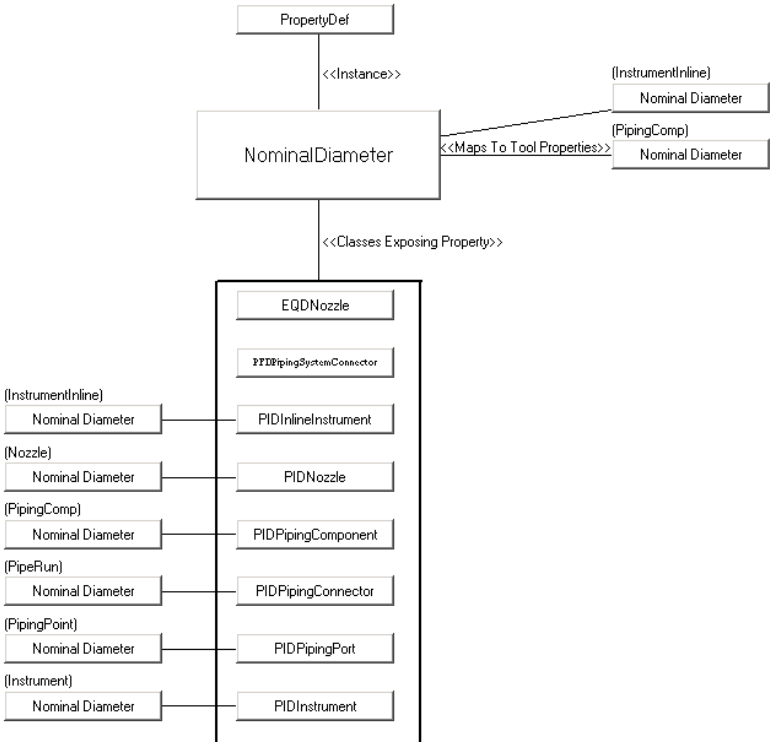
5. To see how a particular SmartPlant schema property maps to and from the various tools, you can click the property in the UML view.

   For example, clicking the NominalDiameter property in the previous example displays the following UML:



6. To see the complete mapping for the SmartPlant property in another UML view, click the **Mapping** button.

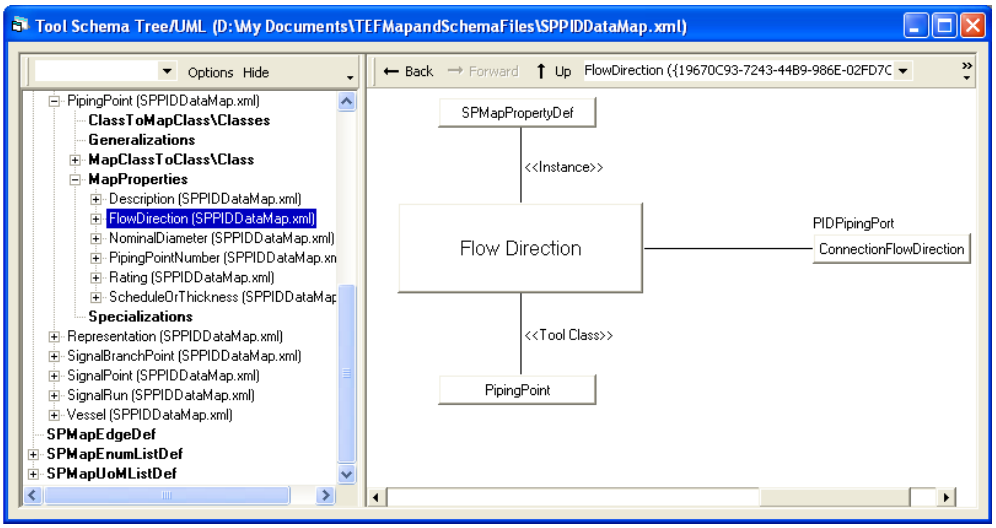
For example, the following property map shows the mapping associated with the NominalDiameter property in the SmartPlant P&ID map file.



TIPS

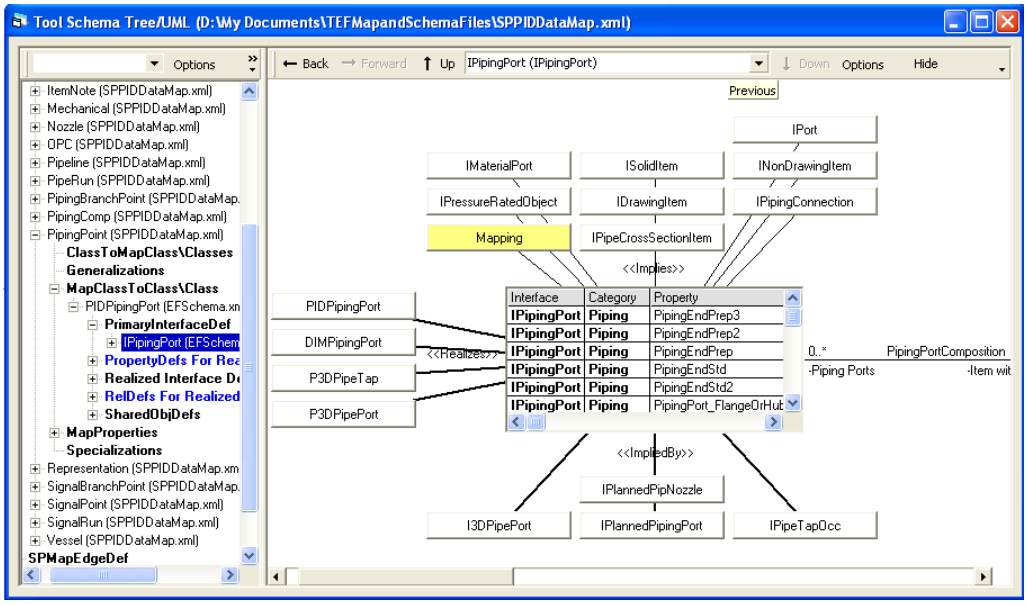
- For each tool property that maps to or is mapped to by the selected SmartPlant property, the name of the tool property appears on a button on the left side of the UML view.
- The ID for the tool class, if applicable, appears above the button. For example, the NominalDiameter property in the SmartPlant schema maps to Nominal Diameter from a variety of SmartPlant P&ID classes.
- If multiple tool map schemas are open in the Schema Editor, you can also see how the property is mapped to and maps from other authoring tools.

7. To see the mapping for an individual tool property, expand the **MapProperties** node under the appropriate class definition, and then click the property in the tree view.

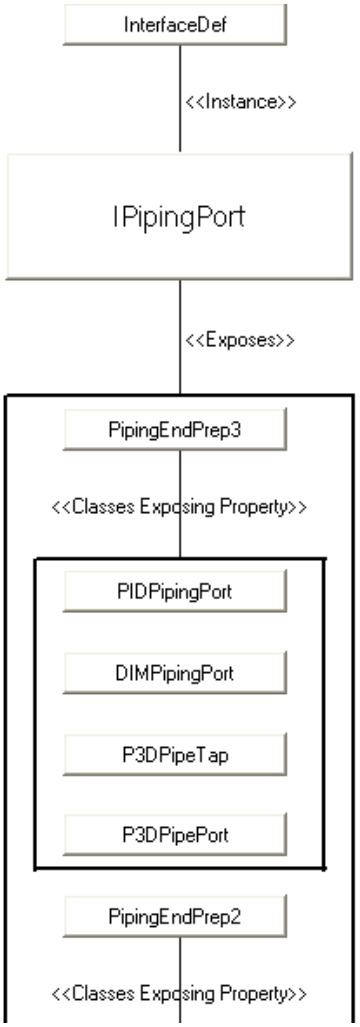


TIP The UML view that you see when you click a property may be different depending on the type of mapping that was used (for example, direct mapping, mapping using realized classes, or mapping using specialization). However, for each type of mapping, the UML view shows the property for each tool class, where applicable, and how the property is mapped to from the SmartPlant schema and how the property maps to the SmartPlant schema.

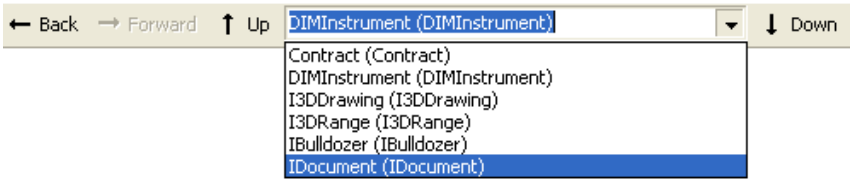
8. To view the mapping for a SmartPlant interface or class, click the interface or class in the tree or UML view, and then click **Mapping**.



9. To see the full mapping for the class or interface, click the **Mapping** button in the UML view.



10. To go back to the last UML view, click **Back** on the toolbar.
11. If you browse back to a previous view, click **Next** to see the next UML view.
12. To select a particular UML view that you have recently viewed, click the name of the view that you want to see in the view list.



13. To move through the view list, use the **Up** and **Down** buttons on the toolbar.

See Also

View Tool-to-Tool Mapping in a Editable UML View (on page 313)
View Tool-to-Tool Mapping for Enumerated Lists (on page 319)
View Tool-to-Tool Mapping for Classes and Properties (on page 316)

View Tool-to-Tool Mapping for Units of Measure (on page 322)
Viewing Authoring Tool Mapping (on page 307)

View Tool-to-Tool Mapping in a Editable UML View

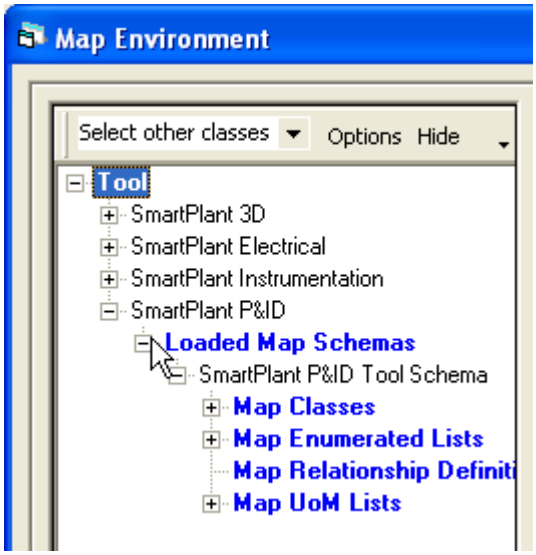
1. Open the two tool map schemas for which you want to view mapping. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

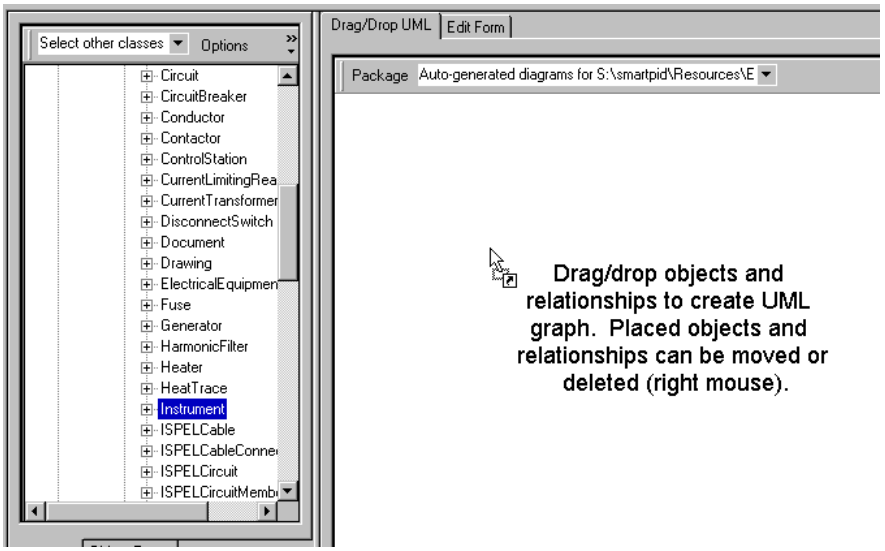
     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. In the **Map Environment**, expand the **Tool** node, and then expand the node for the authoring tool for which you want to view mapping.
3. Under the authoring tool node for the second authoring tool in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.



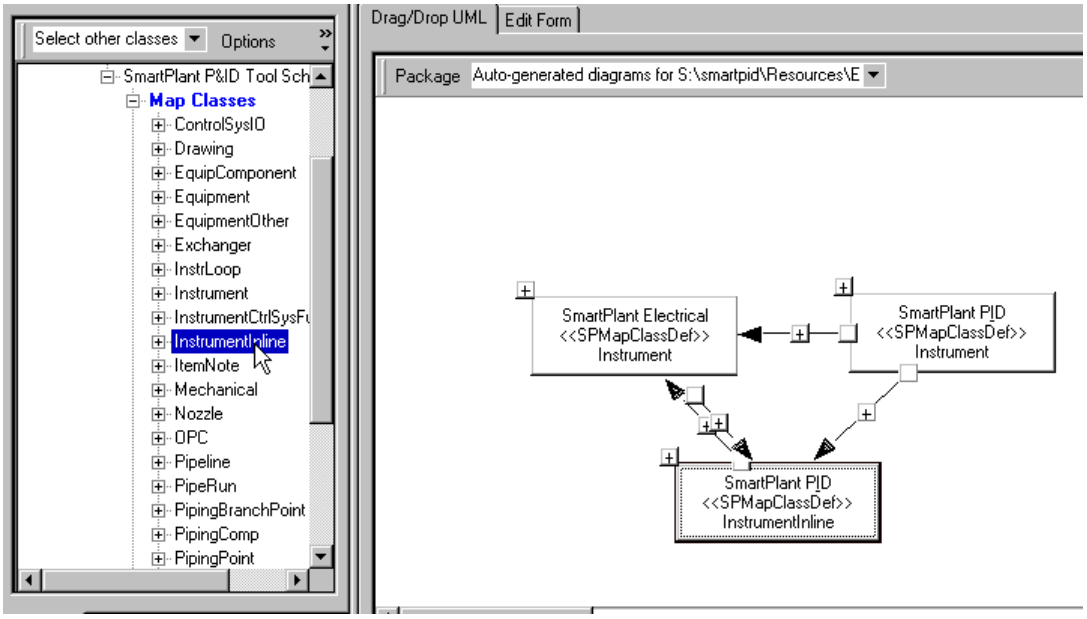
4. Expand the **Map Classes** node.

5. Drag and drop the map class definition onto the **Editable UML** tab to see how it maps to the other tool's map class definition.



TIP Dragging a map class from another application and dropping it on the **Editable UML** tab adds the map class to the diagram. If there is a data pathway or, in some cases, two pathways between the map classes, the software adds an arrow connector between the two map classes.

6. Continue dragging and dropping map class definitions from authoring tools to see mapping among them.

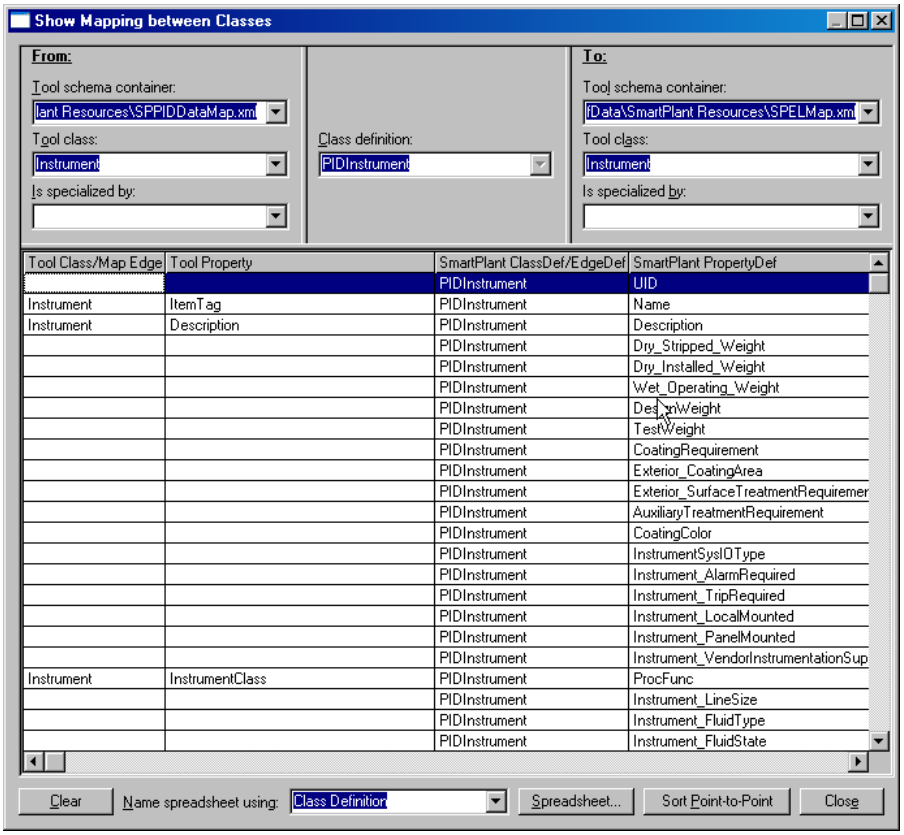


TIP Adding the InstrumentInline map class into this diagram resulted in two connectors being added to the diagram. The connector from InstrumentInline to the SmartPlant Electrical Instrument map class definition represents a valid point-to-point data pathway. The arrow between the SmartPlant P&ID Instrument map class and the InstrumentInline

map class represents a data pathway that, based on the mapping defined in the map schema, is a potential data pathway, but in actuality is not valid because an application will not be retrieving data from one of its map classes into another of its map classes.

7. To view point-to-point mapping for two map class definitions in a tabular view, click + on the connector between the two map class definitions. The **Show Mapping Between Classes** window appears with the two map class definitions populated based on the chosen connector.

   For example, click + on the connector between the SmartPlant Electrical Instrument map class and the SmartPlant P&ID Instrument map class displays the following window:



TIP You can sort the mapping and export it to an Excel spreadsheet from this window. For more information, see *View Tool-to-Tool Mapping for Classes and Properties* (on page 316) or *Export Tool Mapping to a Spreadsheet* (on page 325).

See Also

View Tool-to-SmartPlant Mapping in a Viewable UML View (on page 308)
View Tool-to-Tool Mapping for Enumerated Lists (on page 319)
View Tool-to-Tool Mapping for Classes and Properties (on page 316)
View Tool-to-Tool Mapping for Units of Measure (on page 322)
Viewing Authoring Tool Mapping (on page 307)

View Tool-to-Tool Mapping for Classes and Properties

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Another Tool Schema > Show Mapping Between Classes**.

   TIPS

   - You can also click the **Another Tool Schema File** button in the **Workflows** dialog box to view mapping between tools.
   - You can also open the **Show Mapping Between Classes** window by clicking + on the connector between two map class definitions on the **Drag/Drop UML** tab in the **Map Environment**.

3. Under **From**, select the tool map schema from which you want to see mapping in the **Tool schema container** list.

   TIP You can select the "to" information followed by the "from" information. Alternatively, you can also begin in the middle with the class definition and then select the two tool classes.

4. In the **Tool class** list, select the authoring tool class for which you want to view mapping.

   TIP If the tool class is specialized by another class definition, you can select the specialization for which you want to view mapping in the **Is specialized** by list.

5. If the selected class definition maps to multiple class definitions in the SmartPlant schema, click the SmartPlant class definition for which you want to view mapping in the **Class definition** list.

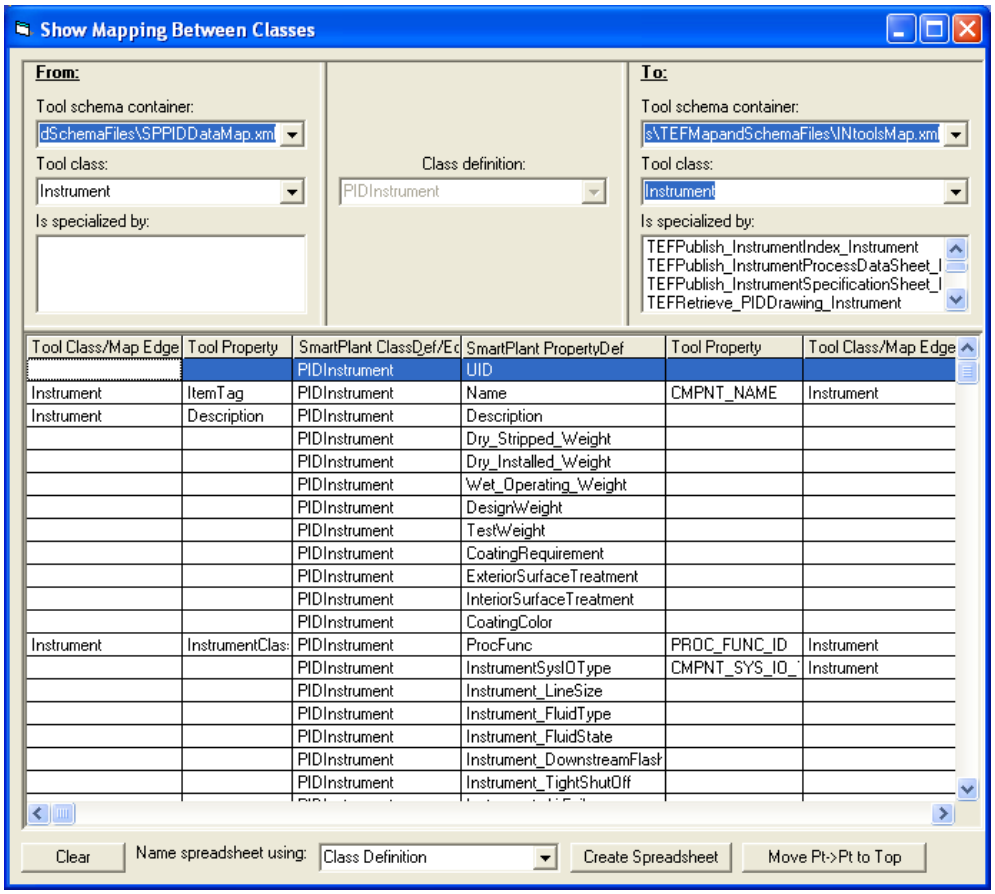
   TIPS

   - If the selected tool class only maps to a single class definition in the SmartPlant schema, the class definition is automatically selected by default in the Class definition list.
   - If the publish mapping for the map class is to an interface definition, then all class definitions that realize that interface definition appear in the list for the class definition.

6. Under **To**, select the tool map schema that you want to see how the selected schema maps to in the **Tool schema container** list.

7. In the **Tool class** list, select the tool class that you want to see mapping to.

The properties for the selected class appear in the table at the bottom of the **Show Mapping Between Classes** window.



TIPS

- If the tool class is specialized by another class definition, you can select the specialization for which you want to view mapping in the **Is specialized by** list.
- At the left of this control are the tool (map) classes and tool (map) properties. The middle two columns show the SmartPlant class definition selected and the SmartPlant property definitions for that class definition (via the realized interface definitions). Where a "from" map property appears in the same row as a SmartPlant property, there is a mapping relationship between the tool (map) property and the SmartPlant property.
- The rightmost two columns show the "to" tool (map) class and the "to" tool (map) property. As before, wherever a SmartPlant property and a "to" map property exist on the same row, there is a map relationship between the property and map property.
- For cases where a "from" map property and a "to" map property exist in the same row, there is a point-to-point mapping between the two (via the SmartPlant property in the middle).

8. To view properties that map across the selected tool map schemas at the top of the table, click **Move Pt -> Pt to Top**.

NOTES

- Not all authoring tools externalize all of their mapping in tool map schemas or map files. Some mapping may be defined in the authoring tool adapter code. This kind of mapping does not appear in mapping spreadsheets generated in the Schema Editor.
- To generate a spreadsheet containing the mapping, click **Create Spreadsheet**. For more information about creating mapping spreadsheets, see *Export Tool Mapping to a Spreadsheet* (on page 325).
- The table in the **Show Mapping Between Classes** window helps you see how properties map across tools. The columns in the table include:
  - **Tool Class/Map Edge** - Displays the selected class definition or edge definition in the tool map schema you are viewing mapping from.
  - **Tool Property** - Displays the properties for the class definition defined in the tool map schema you are viewing mapping from.
  - **SmartPlant ClassDef/EdgeDef** - Displays the class definition or edge definition in the SmartPlant schema that is mapped to by the selected class definition or edge definition in the tool map schema you are viewing mapping from.
  - **SmartPlant PropertyDef** - Displays the property definition in the SmartPlant schema for the class definition (via the realized interface definitions) in the **SmartPlant ClassDef/EdgeDef** column.
  - **Tool Property** - Displays the properties for the class definition defined in the tool map schema you are viewing mapping to.
  - **Tool Class/Map Edge** - Displays the selected class definition or edge definition in the tool map schema you are viewing mapping to.
- If a SmartPlant property and a "to" map property exist on the same row in the **Show Mapping Between Classes** table, there is a map relationship between the property and map property. If a "from" map property and a "to" map property exist in the same row, there is a point-to-point mapping between the two (via the SmartPlant property in the middle).

See Also

View Tool-to-SmartPlant Mapping in a Viewable UML View (on page 308)
View Tool-to-Tool Mapping in a Editable UML View (on page 313)
View Tool-to-Tool Mapping for Enumerated Lists (on page 319)
View Tool-to-Tool Mapping for Units of Measure (on page 322)
Viewing Authoring Tool Mapping (on page 307)

View Tool-to-Tool Mapping for Enumerated Lists

1. Open the two tool map schemas for which you want to view mapping. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

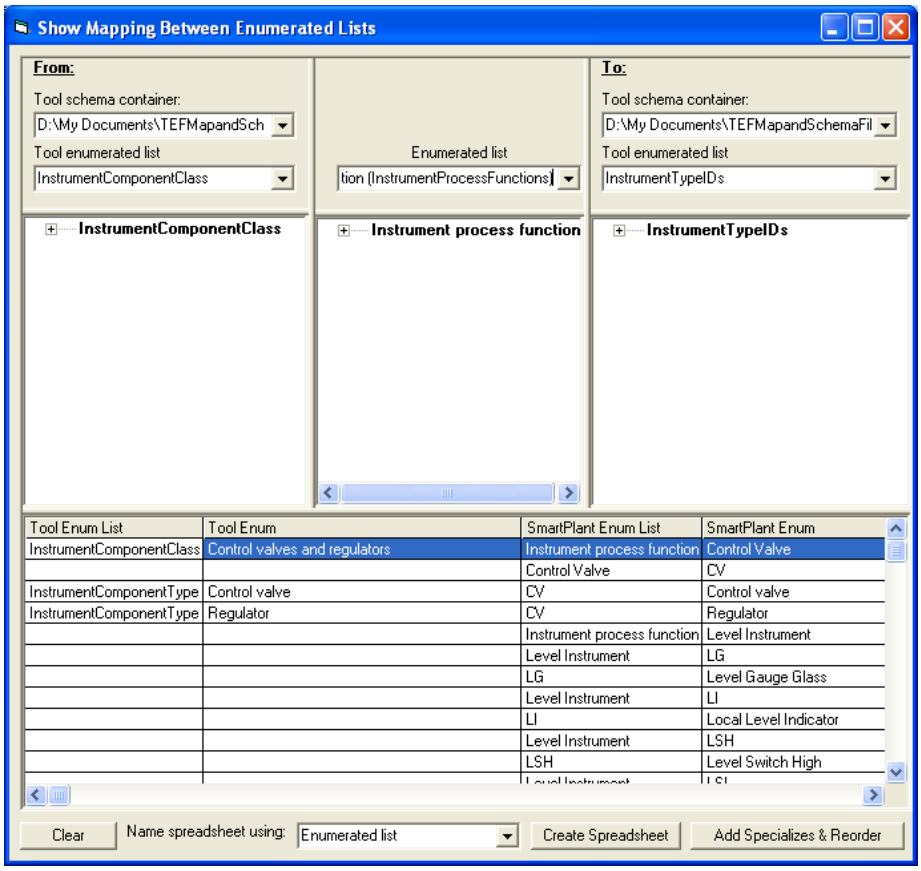
     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Another Tool Schema > Show Mapping Between Enum Lists**.

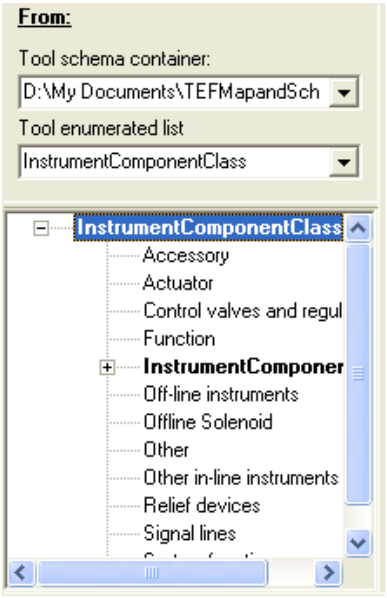
   TIP You can also click the **Another Tool Schema File** button in the **Workflows** dialog box to view mapping between tools.

3. Under **From**, select the tool map schema from which you want to see mapping in the **Tool schema container** list.
4. In the **Tool enumerated list** box, select the authoring tool enumerated list for which you want to view mapping.
5. Under **To**, select the tool map schema that you want to see how the selected schema maps to in the **Tool schema container** list.
6. In the **Tool enumerated list** box, select the tool list that you want to see mapping to.

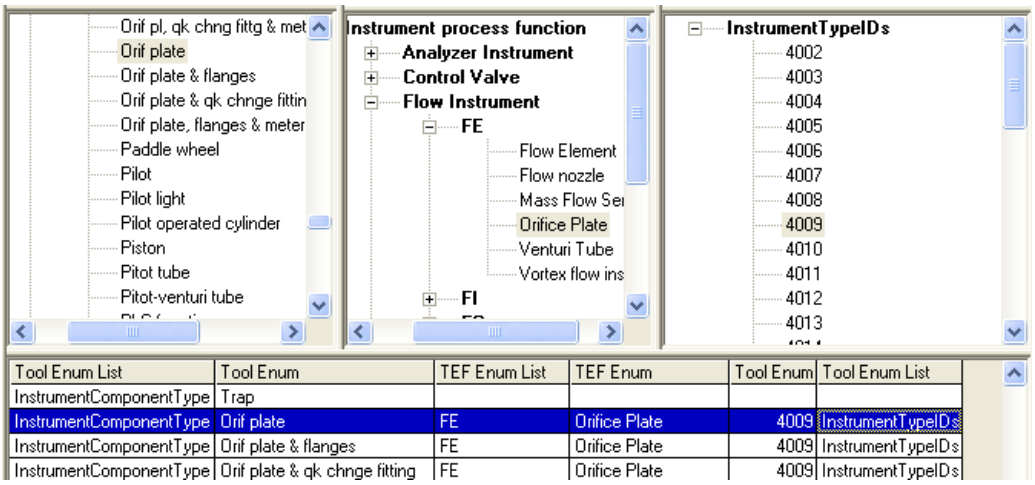
The enumerated lists from each tool map schema and the SmartPlant schema appear in the **Show Mapping Between Enumerated Lists** window. You can see how the enumerated lists map across tools in the table at the bottom of the window.



7. To view the entire enumerated list in one of the tools or the SmartPlant schema, expand the list in the appropriate tree view.



8. To view the mapping for a particular enumerated list entry, click the list entry in the tree view to highlight its mapping in the table and the other tree views.



NOTES

- Not all authoring tools externalize all of their mapping in tool map schemas or map files. Some mapping may be defined in the authoring tool adapter code. This kind of mapping does not appear in mapping spreadsheets generated in the Schema Editor.
- To generate a spreadsheet containing the mapping, click **Create Spreadsheet**. For more information about creating mapping spreadsheets, see *Export Tool Mapping to a Spreadsheet* (on page 325).

- The table in the **Show Mapping Between Enumerated Lists** window helps you see how properties map across tools. The columns in the table include:
  - **Tool Enum List** - Displays the selected enumerated list in the tool map schema you are viewing mapping from.
  - **Tool Enum** - Displays the enumerated list entry from the selected enumerated list defined in the tool map schema you are viewing mapping from.
  - **SmartPlant Enum List** - Displays the enumerated list type in the SmartPlant schema that is mapped to by the selected enumerated list in the tool map schema you are viewing mapping from.
  - **SmartPlant Enum** - Displays the enumerated list entry in the SmartPlant schema that is mapped to by the enumerated list entry in the tool map schema.
  - **Tool Enum** - Displays the enumerated list entry defined in the tool map schema you are viewing mapping to.
  - **Tool Enum** - Displays the selected enumerated list in the tool map schema you are viewing mapping to.
- If a SmartPlant enumeration and a "to" enumeration exist on the same row in the **Show Mapping Between Enumerated Lists** table, there is a map relationship between the enumeration and map enumeration. If a "from" map enumeration and a "to" map enumeration exist in the same row, there is a point-to-point mapping between the two (via the SmartPlant enumeration in the middle).

See Also

View Tool-to-SmartPlant Mapping in a Viewable UML View (on page 308)
View Tool-to-Tool Mapping in a Editable UML View (on page 313)
View Tool-to-Tool Mapping for Classes and Properties (on page 316)
View Tool-to-Tool Mapping for Units of Measure (on page 322)
Viewing Authoring Tool Mapping (on page 307)

View Tool-to-Tool Mapping for Units of Measure

1. Open the two tool map schemas for which you want to view mapping. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

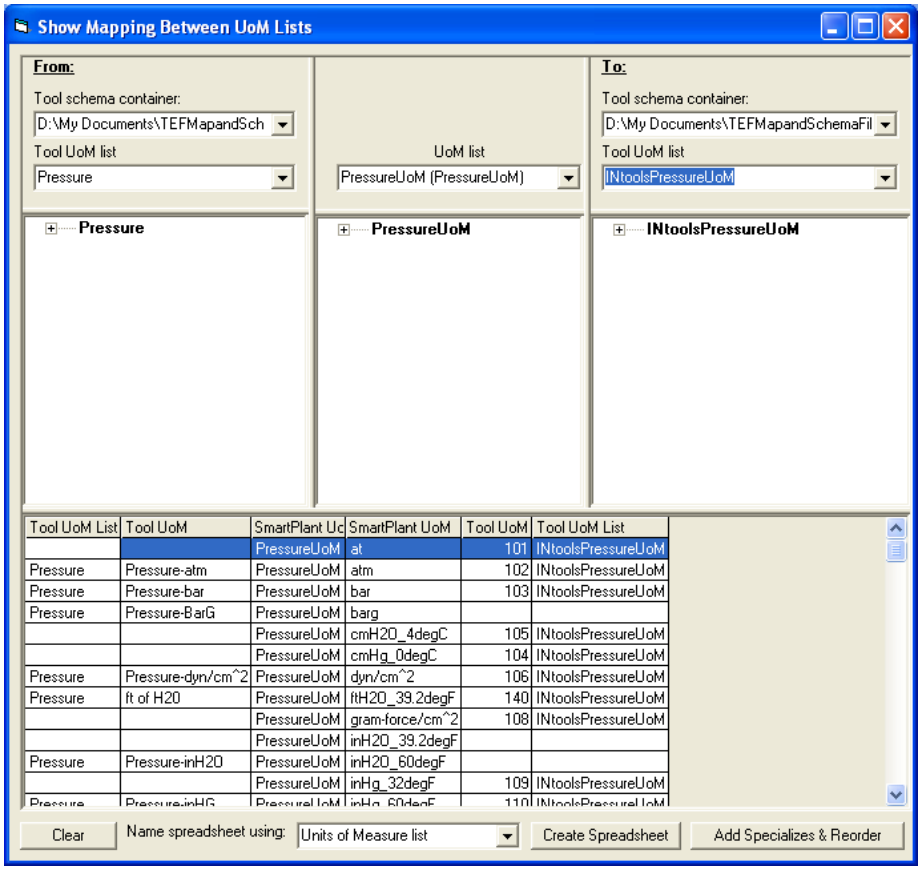
     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Another Tool Schema > Show Mapping Between UoM Lists**.

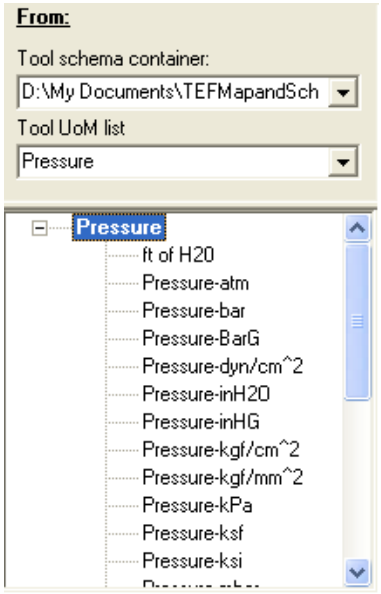
   TIP You can also click the **Another Tool Schema File** button in the **Workflows** dialog box to view mapping between tools.

3. Under **From**, select the tool map schema from which you want to see mapping in the **Tool schema container** list.
4. In the **Tool UoM list** box, select the authoring tool's unit of measure list for which you want to view mapping.
5. Under **To**, select the tool map schema that you want to see how the selected schema maps to in the **Tool schema container** list.
6. In the **Tool UoM list** box, select the tool list that you want to see mapping to.

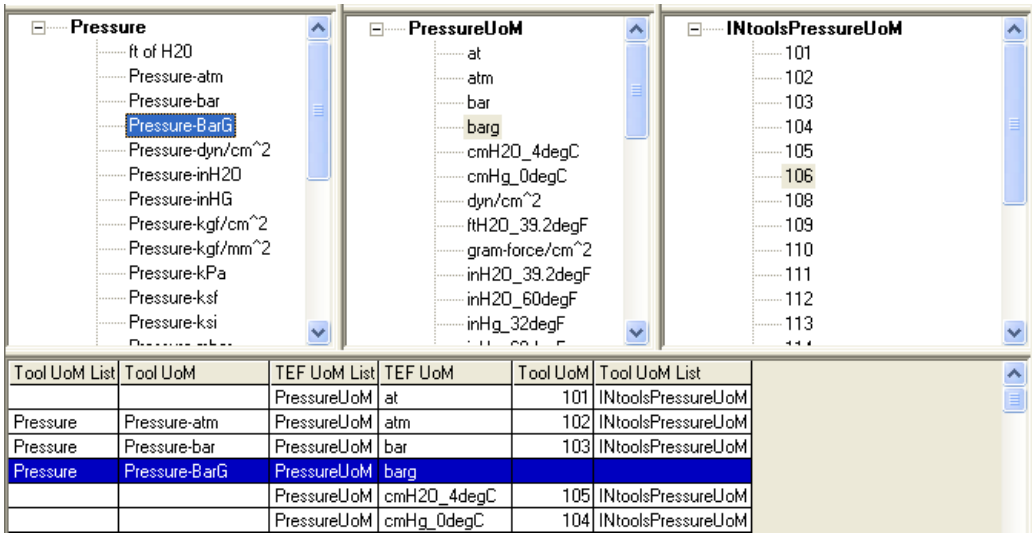
   The UoM lists from each tool map schema and the SmartPlant schema appear in the **Show Mapping Between UoM Lists** window. You can see how the UoMs map across tools in the table at the bottom of the window.



7. To view the entire UoM list in one of the tools or the SmartPlant schema, expand the list in the appropriate tree view.



8. To view the mapping for a particular UoM, click the UoM in the tree view to highlight its mapping in the table and the other tree views.



Tool UoM List	Tool UoM	TEF UoM List	TEF UoM	Tool UoM	Tool UoM List
		PressureUoM	at	101	INtoolsPressureUoM
Pressure	Pressure-atm	PressureUoM	atm	102	INtoolsPressureUoM
Pressure	Pressure-bar	PressureUoM	bar	103	INtoolsPressureUoM
Pressure	Pressure-BarG	PressureUoM	barg		
		PressureUoM	cmH20_4degC	105	INtoolsPressureUoM
		PressureUoM	cmHg_0degC	104	INtoolsPressureUoM

NOTES

- Not all authoring tools externalize all of their mapping in tool map schemas or map files. Some mapping may be defined in the authoring tool adapter code. This kind of mapping does not appear in mapping spreadsheets generated in the Schema Editor.
- To generate a spreadsheet containing the mapping, click **Create Spreadsheet**. For more information about creating mapping spreadsheets, see *Export Tool Mapping to a Spreadsheet* (on page 325).

- The table in the **Show Mapping Between UoMs** window helps you see how properties map across tools. The columns in the table include:
  - **Tool UoM List** - Displays the selected UoM list in the tool map schema you are viewing mapping from.
  - **Tool UoM** - Displays the UoM list entry from the selected UoM list defined in the tool map schema you are viewing mapping from.
  - **SmartPlant UoM List** - Displays the UoM list type in the SmartPlant schema is mapped to by the selected UoM list in the tool map schema you are viewing mapping from.
  - **SmartPlant UoM** - Displays the UoM list entry in the SmartPlant schema that is mapped to by the UoM list entry in the tool map schema.
  - **Tool UoM** - Displays the UoM list entry defined in the tool map schema you are viewing mapping to.
  - **Tool UoM List** - Displays the selected UoM list in the tool map schema you are viewing mapping to.
- If a SmartPlant enumeration and a "to" UoM exist on the same row in the **Show Mapping Between Enumerated Lists** table, there is a map relationship between the UoM and map UoM. If a "from" map UoM and a "to" map UoM exist in the same row, there is a point-to-point mapping between the two (via the SmartPlant UoM in the middle).

See Also

View Tool-to-SmartPlant Mapping in a Viewable UML View (on page 308)
View Tool-to-Tool Mapping in a Editable UML View (on page 313)
View Tool-to-Tool Mapping for Enumerated Lists (on page 319)
View Tool-to-Tool Mapping for Classes and Properties (on page 316)
Viewing Authoring Tool Mapping (on page 307)

Export Tool Mapping to a Spreadsheet

1. Open the two tool map schemas for which you want to export mapping.

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

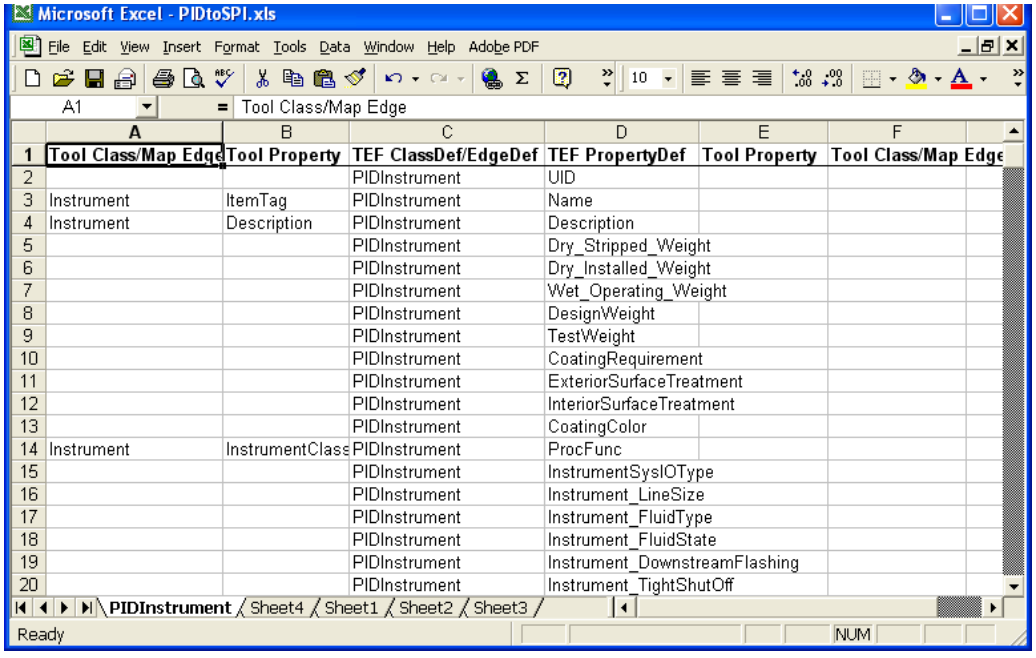
     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.
2. Click **Tools > Another Tool Schema** and then click the type of mapping that you want to export. For example, to export enumerated list mapping, click **Tools > Another Tool Schema > Show Mapping Between Enum Lists**.

   TIP You can also click the **Another Tool Schema File** button in the **Workflows** dialog box to view and export mapping between tools.

3. Under **From**, select the tool map schema from which you want to see mapping and the class definition, UoM list, or enumerated list for which you want to export mapping.
4. If the selected class definition, UoM, or enumerated list maps to multiple class definitions in the SmartPlant schema, click the SmartPlant object for which you want to view mapping in the **Class definition** list.

   TIP If the selected object only maps to a single class definition in the SmartPlant schema, the class definition is automatically selected by default in the **Class definition** list.
5. Under **To**, select the tool map schema and the class definition, UoM list, or enumerated list that you want to see how the selected schema maps to.
6. In the **Name spreadsheet using** list, select the type of object that you want to use to name the mapping spreadsheet. For example, for a class definition spreadsheet, you can choose to name the spreadsheet using the class definition name in the SmartPlant schema, the class definition you are viewing mapping from, or the class definition you are viewing mapping to.
7. Click **Create Spreadsheet**.
8. In the **Update Spreadsheet File** dialog box, type the name of a new Excel workbook or browse to an existing workbook, and click **Open**.

   TIP If you select an existing Excel workbook, the new mapping spreadsheet is added as a new tab in the existing workbook.
9. To view the spreadsheet, browse to the location where you saved the workbook, and double-click the file to open it in Excel.



	A	B	C	D	E	F
1	Tool Class/Map Edge	Tool Property	TEF ClassDef/EdgeDef	TEF PropertyDef	Tool Property	Tool Class/Map Edge
2			PIDInstrument	UID		
3	Instrument	ItemTag	PIDInstrument	Name		
4	Instrument	Description	PIDInstrument	Description		
5			PIDInstrument	Dry_Stripped_Weight		
6			PIDInstrument	Dry_Installed_Weight		
7			PIDInstrument	Wet_Operating_Weight		
8			PIDInstrument	DesignWeight		
9			PIDInstrument	TestWeight		
10			PIDInstrument	CoatingRequirement		
11			PIDInstrument	ExteriorSurfaceTreatment		
12			PIDInstrument	InteriorSurfaceTreatment		
13			PIDInstrument	CoatingColor		
14	Instrument	InstrumentClass	PIDInstrument	ProcFunc		
15			PIDInstrument	InstrumentSysIOType		
16			PIDInstrument	Instrument_LineSize		
17			PIDInstrument	Instrument_FluidType		
18			PIDInstrument	Instrument_FluidState		
19			PIDInstrument	Instrument_DownstreamFlashing		
20			PIDInstrument	Instrument_TightShutOff		

TIP The columns in the spreadsheet correspond to the columns in the table in the **Show Mapping Between Classes**, **Show Mapping Between UoMs**, **and Show Mapping Between Enumerated Lists** windows. For more information, see *View Tool-to-Tool Mapping for Classes and Properties* (on page 316), *View Tool-to-Tool Mapping for Units of Measure* (on page 322), or *View Tool-to-Tool Mapping for Enumerated Lists* (on page 319).

NOTE You can download spreadsheets containing the latest authoring tool mapping for the current releases of the authoring tools from the Intergraph Smart Support web site. To download the spreadsheets, browse to the following web address, and type your Intergraph Smart Support user name and password: *http://smartsupport.intergraph.com* (*https://smartsupport.intergraph.com*). Click **View Downloads**, select **SmartPlant Foundation**, and then click **Technical Notes and Whitepapers**. Click the link to the mapping spreadsheets to download them to your computer.

See Also

Viewing Authoring Tool Mapping (on page 307)

Work Processes for Mapping for Tools with Metadata Adapters

For authoring tools that deliver metadata adapters, there are two work processes for defining objects in the tool metadata, the tool map schema, and the SmartPlant schema, and then mapping those objects for publish and retrieve. The main difference between the two work processes is the starting point for defining objects: you can either start in the authoring tool to define objects or start in the SmartPlant schema.

NOTE If you modify tool mapping using the metadata adapter in conjunction with the Schema Editor, you should always connect to SmartPlant and work through the **Map Environment** for mapping modification. If you switch to modifying the tool map schema outside the Map Environment (as described in *Work Process for Mapping for Tools without Metadata Adapters* (on page 427)), the Schema Editor removes information from the tool map schema that may be required by the tool metadata adapter. You must choose a method for modifying your tool map schema and continue that way.

Understanding Mapping

Before you begin defining authoring tool mapping, you must understand the following:

- The SmartPlant schema model and how it works. For more information, see *SmartPlant Schema Overview* (on page 18) and *A Gentle* In*troduction to the SmartPlant Schema*, delivered with SmartPlant Foundation.
- How tool map schemas work and how they are used by the authoring tools. For more information, see *Tool Map Schemas* (on page 259).
- The authoring tool data model for which you are defining mapping. For more information, see your authoring tool documentation.
- The delivered tool map schema and mapping for the authoring tool. For more information see *Viewing Authoring Tool Mapping* (on page 307) and rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.

To do mapping using an authoring tool's metadata adapter, that authoring tool must also be registered with the appropriate SmartPlant Foundation plant.

The actual order of the steps in the process will vary, depending on whether you start the process in the SmartPlant schema or the tool metadata. The following sections illustrate the process from these two perspectives:

- *Mapping New Objects Originating in the Tool Metadata* (on page 332)
- *Mapping New Objects Originating in the SmartPlant Schema* (on page 363)

Before You Begin Mapping for Tools with Metadata Adapters

Intergraph recommends that you make changes to the tool metadata, tool map schema, and SmartPlant schema in a test environment, and then move the changes to your production system.

For some authoring tools, you may need to reproduce your changes in the production environment. For other tools, you may be able to copy your changes using tool software. For more information, see your authoring tool documentation.

Before you begin mapping, you should also do the following:

- Back up your tool metadata database
- Back up your tool map schema

Make sure that your tool map schema file is editable. Some tool map schemas are delivered as read-only files. For more information about the location of tool map schema files for each authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.

Synchronize Tool Metadata with the Tool Map Schema

After you check out the SmartPlant schema CMF file from the Desktop Client and launch the Schema Editor, you can synchronize your tool metadata with the tool map schema.

1. In the **Set Active Configurations** dialog box, select the schema version or versions that you want to look at, and click **OK**.

   TIP By default, the changes you make are applied to all versions of the schema managed by the CMF file, to support forward and backward compatibility between versions of the software. Click **Advanced** to view the **Set Configurations for new objects window**, where you can select a specific version to which to write changes.

2. In the Schema Editor, click **File > SmartPlant > Edit Engineering Tool Options**.

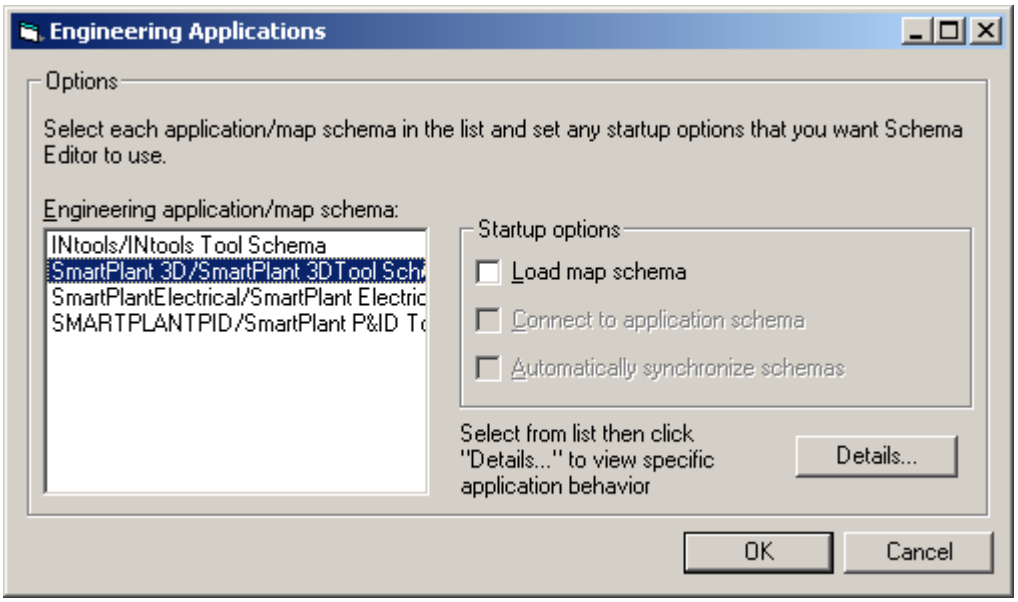
   TIP A message box appears if authoring tools without metadata adapters are registered with the SmartPlant Foundation plant. Mapping for these authoring tools can be defined by opening their tool map schemas.

3. Under **Engineering application/map schema**, select the authoring tool and tool map schema that you want to work with in the Schema Editor.

   TIPS

   - You can load multiple tool map schemas and metadata by repeating steps 2 and 3 for each authoring tool.
   - The authoring tool that you want to define mapping for must be registered with the SmartPlant Foundation site and plant from which you checked out the SmartPlant schema CMF file to appear in the **Engineering application/map schema** list.

4. Under **Startup options**, select the **Load map schema** and **Connect to application schema** check boxes.



TIP For specific details about what types of objects that you can edit in the authoring tool via the metadata adapter and how save works for the tool's metadata adapter, select a tool and then click the **Details** button.

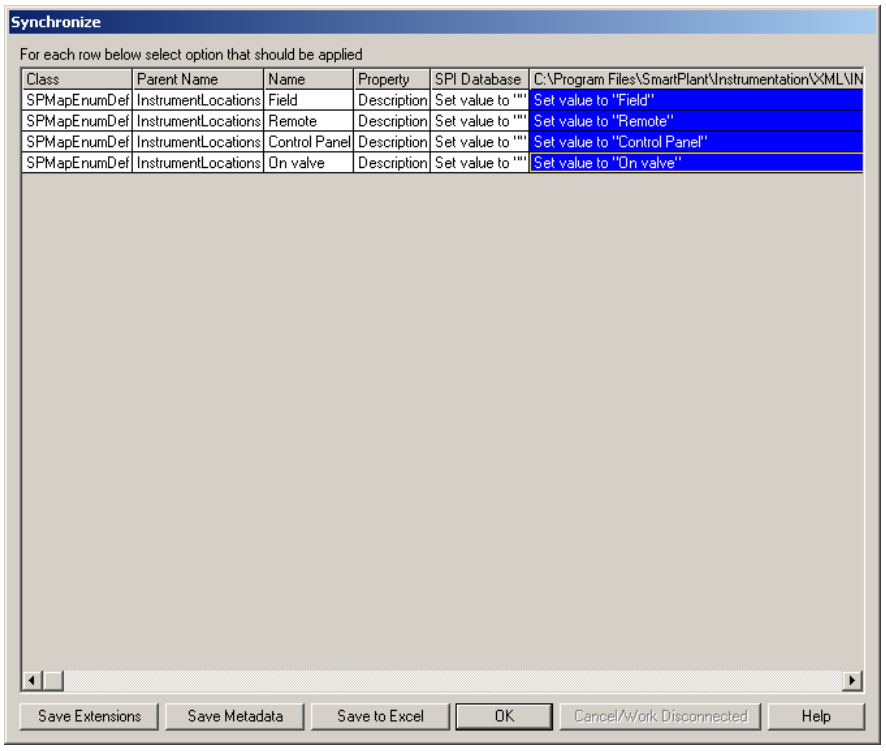
5. To synchronize additional tool schemas at the same time, repeat steps 4-5 for each authoring tool that you want to synchronize.

6. Click **Finish** to review synchronization details in the **Synchronize** dialog box.

TIPS

- Connecting to the metadata adapter and generating a comparison of the tool metadata and the tool map schema can be a time consuming task, depending on your computer memory and processor speed.
- When you click **Finish**, the Schema Editor generates a tool schema from the tool metadata in the tool database and compares it to the tool map schema file.
- If no changes have been made to the authoring tool metadata or the tool map schema since you installed the software, there may no differences to synchronize.
- Once you are familiar with synchronization and the default actions for the authoring tool you are working with, you may want to select the **Automatically synchronize schemas** check box before you click **Finish** to avoid reviewing synchronization results every time you load a tool map schema and tool metadata.

7. In the **Synchronize** dialog box, select the appropriate action in the ***<Tool>* Database** or ***<Tool Map Schema Path>*** column in the appropriate row for the change to update inconsistencies between the tool metadata and tool map schema. Default actions appear in blue in the appropriate column and row for each difference.



IMPORTANT

- Some applications do not support both possible synchronization actions that appear in the **Synchronization** dialog box. For example, SmartPlant P&ID does not support deleting existing properties or enumerated lists. If properties exist in the SmartPlant P&ID tool metadata that are not in the tool map schema, then the missing properties must be added to the tool map schema. They cannot be deleted from the tool metadata database.
- If you select a synchronization action that is not supported by the selected authoring tool, the Schema Editor displays a message box to let you know that the action is not supported.

TIP You can export the synchronize instructions to Microsoft Excel by clicking **Save to Excel**. Saving to a spreadsheet allows you to sort, filter, and share synchronization information.

8. Click **OK** to synchronize the tool map schema and tool metadata.

TIPS

- If you selected more than one tool map schema to synchronize, repeat steps 7-8 for each authoring tool that you selected.
- After you synchronize the last tool map schema, the **Map Environment** appears.

Synchronize Dialog Box

Allows you to review differences between the tool map schema and the tool metadata database. You can select an action to resolve the difference.

Class - Specifies the class definition for the object that is inconsistently defined in the tool map schema and the tool metadata database. For example, if an enumerated list definition has the inconsistency, the class is SPMapEnumDef.

Parent Name - Specifies the parent object for the for the object that is inconsistently defined. For map enumerations, the parent object is the map enumerated list. For map units of measure, the parent object is the map unit of measure list, and for map properties, the parent object is the map class. For all other map objects, this column is blank.

Name - Specifies the name of the object that is inconsistently defined.

Property - Specifies the name of the property that is inconsistently defined, if the inconsistency is on a property.

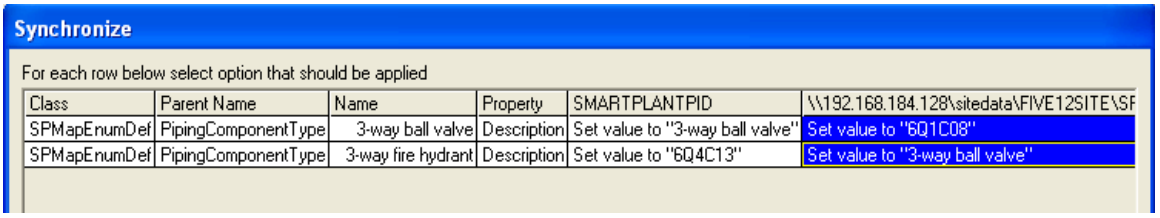
<Tool> Database - Specifies a change that you want to make in the tool database. For example, you can choose to delete an object that does not exist in the tool map schema or set a value for an existing property to the value defined in the tool map schema. If the action in this column is selected (blue) when you click **OK**, the tool database is updated with the selected action.

NOTE You can only delete objects for authoring tools that allow deletion in the metadata database. If you click a **Delete** action in the **<Tool> Database** column for an authoring tool that does not support deletions, the Schema Editor displays a message box to let you know that you cannot select that action. For more information about rules for additions and deletions using metadata adapters, see *Mapping with Metadata Adapters* (on page 291).

<Tool Map Schema Path> - Specifies a change that you want to make in the tool map schema. For example, you can choose to add a missing object that exists in the tool metadata database to the tool map schema or set a value for an existing property in the tool map schema to the value defined in the tool metadata database. If the action in this column is selected (blue) when you click **OK**, the tool map schema is updated with the selected action.

For example, the following graphic shows two selected updates that the software will make to property descriptions in the tool map schema when the user clicks **OK**:

- Change the description of the 3-way ball valve property in the tool map schema from "3-way ball valve" to "6Q1C08" to match the value currently defined in the tool metadata database
- Change the description of the 3-way fire hydrant in the tool map schema from "6Q4C13" to "3-way ball valve" to match the value currently defined in the tool metadata database



Save Extensions - Saves the meta schema extensions added by the metadata adapter to an XML file.

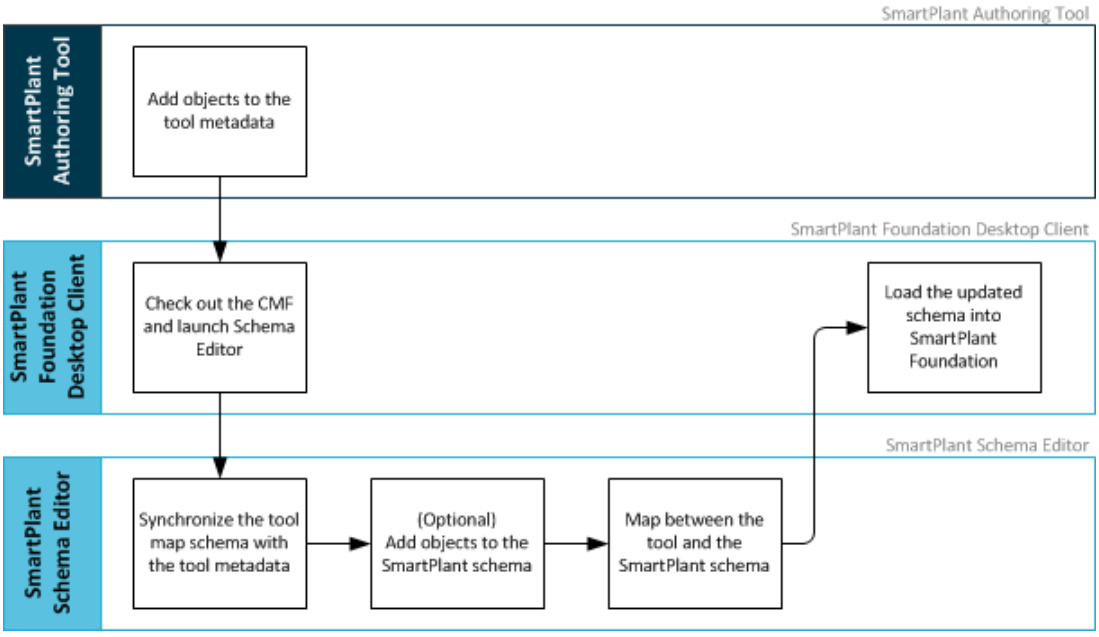
Save Metadata - Saves metadata to an XML file.

Save to Excel - Saves the contents of the **Synchronize** dialog box to a Microsoft Excel spreadsheet.

Mapping New Objects Originating in the Tool Metadata

The following graphic provides an overview of the work process for mapping new objects when you start inside the authoring tool. The graphic covers the work process for adding and mapping objects that do not already exist in the tool database, tool map schema, or SmartPlant schema. If the object already exists in the tool map schema or the SmartPlant schema, you can skip the steps that involve adding new objects in the work process.

This approach is most useful when you have existing metadata in the authoring tool that needs to be entered in the SmartPlant schema and mapped. Intergraph recommends starting in the authoring tool when you define new properties so that the property definition fits into the authoring tool's data model appropriately.



What do you want to do?

- *Add Objects to the Tool Metadata* (on page 333)
- *Check Out the CMF and Launch Schema Editor* (on page 333)
- *Synchronize the Tool Map Schema with the Tool Metadata* (on page 333)
- *Add Objects to the SmartPlant Schema* (on page 334)
- *Map Between the Tool and the SmartPlant Schema* (on page 334)
- *Load the Updated Schema into SmartPlant Foundation* (on page 334)

Add Objects to the Tool Metadata

If you want to define mapping between an authoring tool and the SmartPlant schema, you can start by adding the object that you want to map to the authoring tool metadata database. Any supporting objects, such as enumerated lists or units of measure, also need to be added in the tool database so that the objects can be published and retrieved. For example, if you add a new property to your authoring tool database and values for that property are defined by an enumerated list, you must also define the enumerated list in the tool database.

NOTE The procedure for adding new objects to the tool database varies depending on the authoring tool you are using. For example, to add new objects to SmartPlant P&ID, you must use Data Dictionary Manager. For more information about adding new objects to the tool database, see the appropriate authoring tool documentation.

Check Out the CMF and Launch Schema Editor

Before you can make modifications to the SmartPlant schema, must check the CMF file out of the SmartPlant Foundation Desktop Client.

- Check out the SmartPlant schema CMF file from the SmartPlant Foundation Desktop Client.
- Make sure your scope is set to the plant, right-click the new revision of the CMF file in SmartPlant Foundation, and click **Launch Schema Editor**. This provides the Schema Editor with the appropriate plant and SmartPlant Foundation site for connection.
- Select the schema version or versions that you want to view in the **Specify configurations to display** window of the **Active Configurations** dialog box.
- Click the **Advanced** button on the **Active Configurations** dialog box to view the **Set Configurations for new objects** window, where you can specify what version or versions of the schema changes should be written to. For example, to modify the 2016 schema, select the 2016 configuration for new objects.

NOTE By default, all configurations are checked to allow forward and backward compatibility between schema versions when creating new objects or modifying existing objects.

Opening the CMF file also allows you to see which authoring tools that are registered with the SmartPlant Foundation plant have available metadata adapters.

Synchronize the Tool Map Schema with the Tool Metadata

Synchronization allows you to keep the tool map schema and the tool metadata in the authoring tool in synch without requiring manual steps. In this work process, synchronization allows you to automatically update the tool map schema with the changes you made to the tool metadata in the authoring tool.

In the Schema Editor, click the **File > SmartPlant > Edit Engineering Tools** command to select the authoring tool or tools that you want to defining mapping for. When you select one or more authoring tools to load, select the **Load Map Schema** check box and the **Connect to application schema** check box. The synchronization between the tool map schema .XML file and the tool metadata, located in the tool database, begins when you click **Finish**.

After the Schema Editor connects to the tool database through the metadata adapter, the software displays a list of inconsistencies between the tool map schema and the tool metadata. When you add new properties, enumerated lists, or enumerated entries in the authoring tool, these additions show up as inconsistencies in the **Synchronization** dialog box.

When you review the additions to the application's metadata, you can choose to add the new object to the tool map schema or delete it from the application metadata (only if deletion of the object type is allowed by the authoring tool). By default, the action that the software will take to resolve the inconsistency is selected in the table, but you can change that action by selecting the other action in the authoring tool metadata column or the tool map schema column.

For more details about using the metadata adapter to synchronize a tool metadata and tool map schema, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

Add Objects to the SmartPlant Schema

After you synchronize your tool's metadata with the tool map schema, you can add new objects to the SmartPlant schema in the **Mapping Environment**, which appears after you click **OK** in the **Synchronize** dialog box. To add a new object in the SmartPlant schema, navigate to the tool map class definition for the property or the map enumerated list that you created or to which you added new entries. On the **Edit** dialog box or the **Edit Form** tab, click the **Publish** or **Retrieve** tab, and then use the toolbar on the right side of the dialog box to add new objects to the SmartPlant schema. For more information about adding objects to the SmartPlant schema, see *Extending the SmartPlant Schema* (on page 146).

Map Between the Tool and the SmartPlant Schema

When the object exists in both the tool map schema and the SmartPlant schema, you can map between the objects using the toolbar on the **Publish** or **Retrieve** tab.

NOTES

- Some authoring tools may have additional steps for performing the mapping between the authoring tool and the SmartPlant schema. For mapping rules for each authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.
- The **Display name** field allows you to assign a unique display name to a definition in the tool map schema. This field is available in the following dialog boxes: **Edit Map Class Definition**, **Edit Map Property Definition**, **Map Relationship Definition**, **Map Enumerated List Definition**, **Map Enumeration**, **Map Unit of Measure List Definition**, **Map Unit of Measure**, and **Map Edge Definition**.

Load the Updated Schema into SmartPlant Foundation

Loading the updated schema into SmartPlant Foundation involves the following high-level tasks:

- Checking the CMF file back into the Desktop Client.
- Running the **CMF File Manager** on the CMF file to create an XML file for loading.
- Importing the schema using the **Schema Import Wizard**.
- Regenerating the component schemas.

Mapping Objects Created in the Authoring Tool

Create and Map New Enumerated List Entries Starting in the Tool

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Create the New Enumerated List Entry in the Authoring Tool

- Use the appropriate authoring tool software to add the new enumerated list entry to an existing enumerated list in the authoring tool metadata database.

  For example, for SmartPlant P&ID or SmartPlant Electrical, add a new select list entry to the appropriate select list in Data Dictionary Manager.

  TIP Terminology for enumerated lists differs from tool-to-tool. Some authoring tools call enumerated lists select lists, picklists, codelists, or lookup tables. For more information, see the documentation for the authoring tool for which you want to add metadata.

Load the Tool Map Schema and Open the Map Enumerated List

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

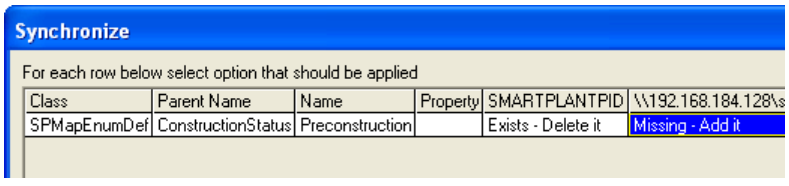
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

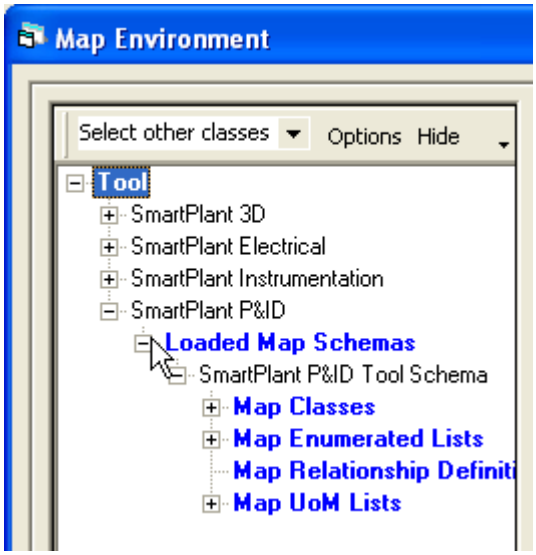
7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

   TIP The new enumerated list entry that you added in the authoring tool appears in the **Synchronization** dialog box to be added to the tool map schema file.



8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.

9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>*** **Tool Schema** node to display contents of the authoring tool map schema.



10. Expand the **Map Enumerated List** node and double-click the map enumerated list to which you added the new list entry.

Add the New List Entries to the SmartPlant Schema

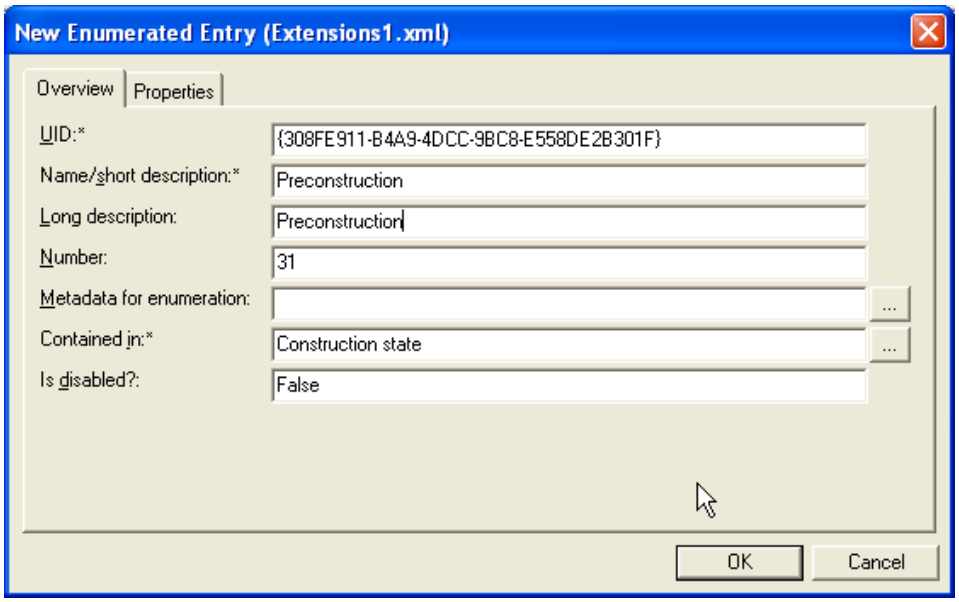
Before you add new entries to the SmartPlant schema, you need to verify that the entries or entries that have equivalent meanings do not already exist.

1. If you want to start by defining publish mapping for the new list entry, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated entry, you can click the **Retrieve** tab instead. The procedure for adding enumerated entries and mapping is the same for both tabs.

2. Above **Unmapped SmartPlant enumerations** on the right side of the dialog box, click **New SmartPlant Enumeration** on the toolbar.

3. In the **New Enumerated Entry** dialog box, type the name and long description for the new enumerated list entry, and click **OK**.

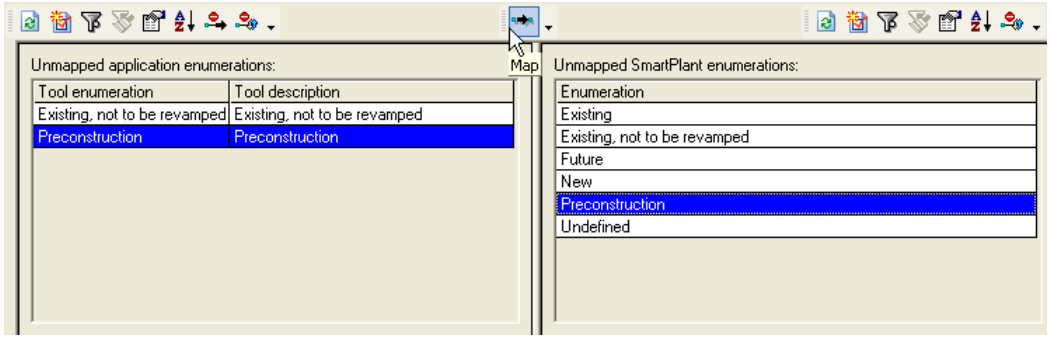


TIPS

- The Schema Editor automatically provides number for the enumerated list entry. This number may need to be modified to match the corresponding codelist value if the entry will be mapped with Smart 3D.
- When you click **OK**, the new list entry appears in the **Unmapped SmartPlant enumerations** table in the **Edit Map Enumerated List Definition** dialog box.

Map Tool List Entries to SmartPlant Schema List Entries

1. In the **Unmapped application enumerations** table, select the tool enumerated list entry that you want to map.
2. In the **Unmapped SmartPlant enumerations** table, select the SmartPlant enumerated list entry that you want to map.
3. On the toolbar, click Map .



TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.
5. Click **OK** in the **Edit Map Enumerated List Definition** dialog box.

Save the Tool Map Schema and Schema Extensions File

- Click **File > Save > All Modified Files** to save the SmartPlant schema extensions and the tool map schema. For some tools, saving the tool map schema also saves changes to the tool metadata database.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click **Yes** in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.
- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

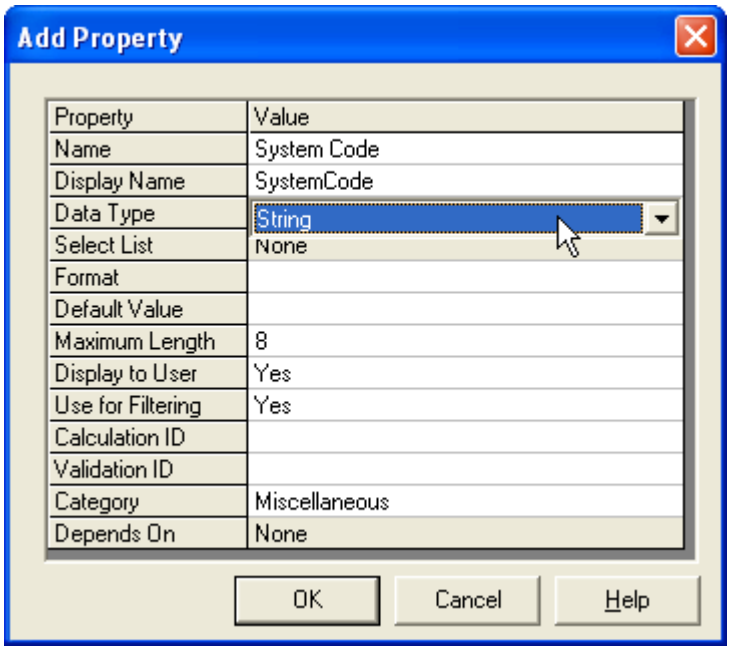
Create and Map New String Properties Starting in the Tool

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Create the New Property in the Authoring Tool

- Use the appropriate authoring tool software to add the new string property to the authoring tool metadata database. Be sure to set the data type to *string* for a string property.

For example, for SmartPlant P&ID or SmartPlant Electrical, add a property to the appropriate database table in Data Dictionary Manager.



Load the Tool Map Schema and Open a Map Class Definition

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

TIPS

- For more options, click **Advanced** on the **Set Active Configuration** dialog box.
- By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
- If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

   TIP The new property that you added in the authoring tool appears in the **Synchronization** dialog box to be added to the tool map schema file.

8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>* Tool Schema** node to display contents of the authoring tool map schema.
10. Expand the **Map Classes** node and double-click the map class definition to which you added the new property.

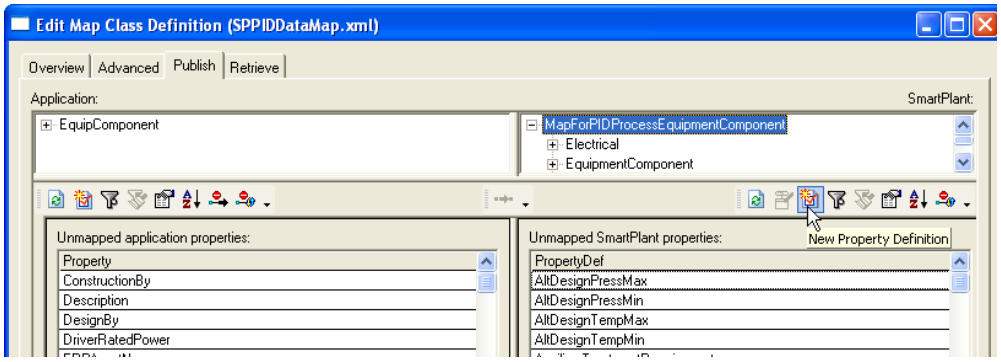
Add the New Property to the SmartPlant Schema

Before you add new properties to the SmartPlant schema, you need to verify that the properties or properties that have equivalent meanings do not already exist.

1. If you want to start by defining publish mapping for the new property, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated property, you can click the **Retrieve** tab instead. The procedure for adding properties and mapping is the same.

2. Above **Unmapped SmartPlant properties** on the right side of the dialog box, click **New SmartPlant Property Definition** on the toolbar.



3. In the **New Property Definition** dialog box, define the new SmartPlant schema property, and click **OK** to create the property definition. For more information, see *Create a Property Definition* (on page 172).

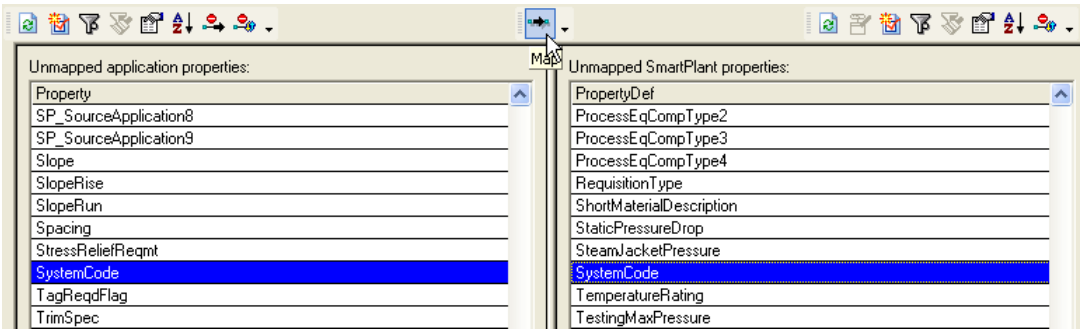
   TIPS

   - In the **Scoped by property type** box, be sure to select **string** as the property type for the new string property.
   - When you click **OK**, the new property appears in the **Unmapped SmartPlant properties** table in the **Edit Map Class Definition** dialog box.

Map Tool Property to the SmartPlant Schema Property

1. In the **Unmapped application properties** table, select the tool map property definition that you want to map.

2. In the **Unmapped SmartPlant properties** table, select the SmartPlant property definition that you want to map.
3. On the toolbar, click **Map** .



TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.
5. Click **OK** in the **Edit Map Class Definition** dialog box.

Save the Tool Map Schema and Schema Extensions File

- Click **File > Save > All Modified Files** to save the SmartPlant schema extensions and the tool map schema. For some tools, saving the tool map schema also saves changes to the tool metadata database.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click **Yes** in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.
- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

See Also

Before You Begin Mapping for Tools with Metadata Adapters (on page 328)
Mapping with Metadata Adapters (on page 291)

Create and Map New Properties with Enumerated Lists Starting in the Tool

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Create the Enumerated List, List Entries, and Property in the Authoring Tool

- Use the appropriate authoring tool software to add new enumerated list, enumerated list entries, and property that uses the enumerated list to the authoring tool metadata database. Be sure to set the data type to *string* for a string property.

  For example, for SmartPlant P&ID or SmartPlant Electrical, create a new select list, add entries to the select list, and then create a property associated with the new select list in the appropriate database table in Data Dictionary Manager. The new property should have a data type of *select list* and a relationship to the select list that you created.

  TIP Terminology for enumerated lists differs from tool-to-tool. Some authoring tools call enumerated lists select lists, picklists, codelists, or lookup tables. For more information, see the documentation for the authoring tool for which you want to add metadata.

Load the Tool Map Schema and Open a Map Class Definition

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

   TIPS

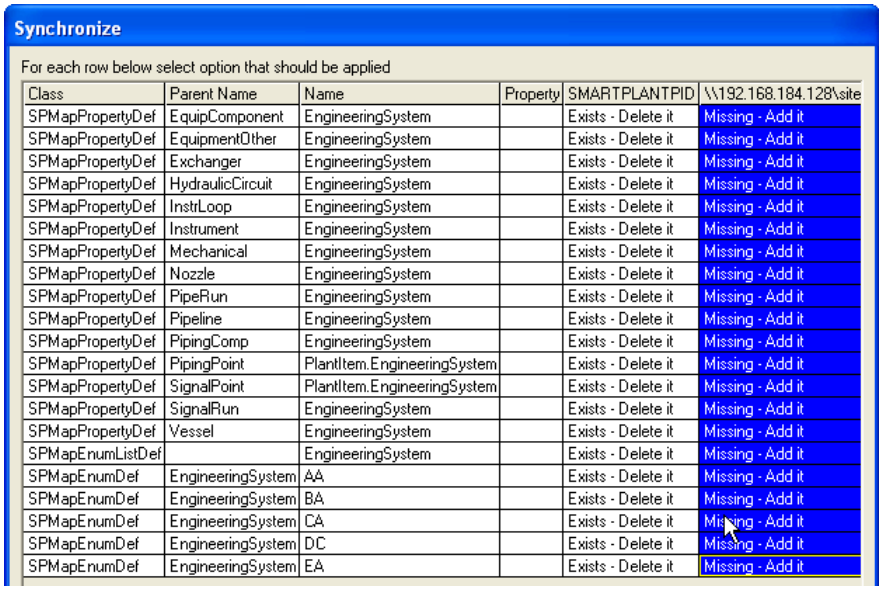
   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

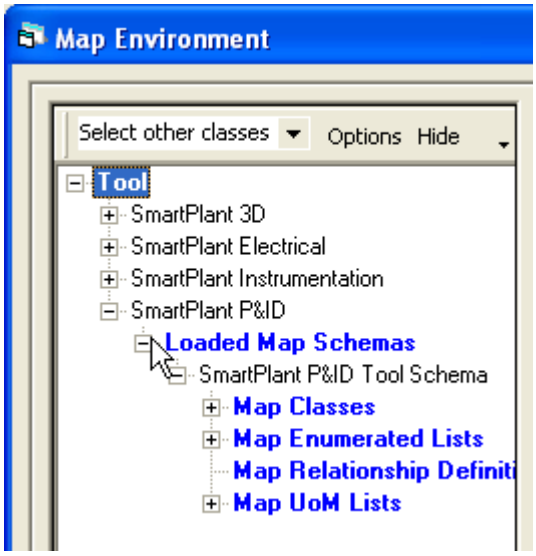
   TIP The new property, enumerated list, and list entries that you added in the authoring tool appear in the **Synchronization** dialog box to be added to the tool map schema file. As you

can see in the following graphic, if you add a property to the Plant Item table in Data Dictionary Manager for SmartPlant P&ID, multiple map classes inherit that property, and therefore, multiple map classes are updated in the tool map schema during synchronize.



Class	Parent Name	Name	Property	SMARTPLANTPID	\\192.168.184.128\site
SPMapPropertyDef	EquipComponent	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	EquipmentOther	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Exchanger	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	HydraulicCircuit	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	InstrLoop	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Instrument	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Mechanical	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Nozzle	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipeRun	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Pipeline	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipingComp	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipingPoint	PlantItem.EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	SignalPoint	PlantItem.EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	SignalRun	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Vessel	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapEnumListDef		EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	AA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	BA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	CA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	DC		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	EA		Exists - Delete it	Missing - Add it

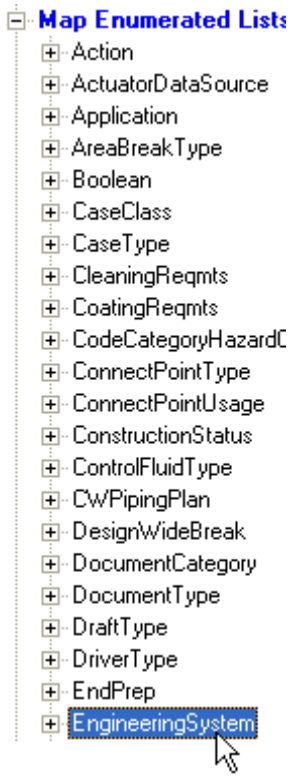
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.



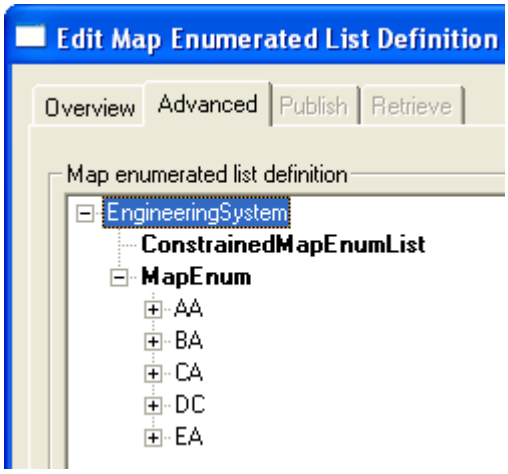
Add the New Enumerated List to the SmartPlant Schema

Before you create the list in the SmartPlant schema, you need to verify that the enumerated list or a list that has an equivalent meaning does not already exist.

1. To add a new enumerated list to the SmartPlant Schema, expand the **Map Enumerated Lists** node, and double-click the new map enumerated list in the tree view.

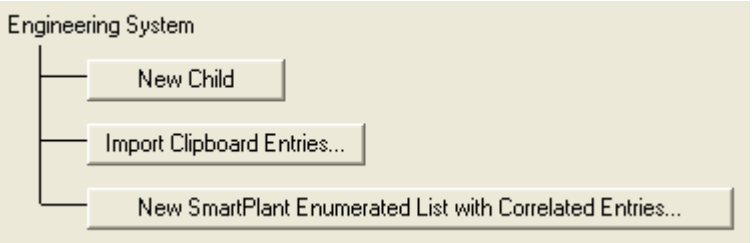


2. In the **Edit Map Enumerated List Definition** dialog box, click the **Advanced** tab.



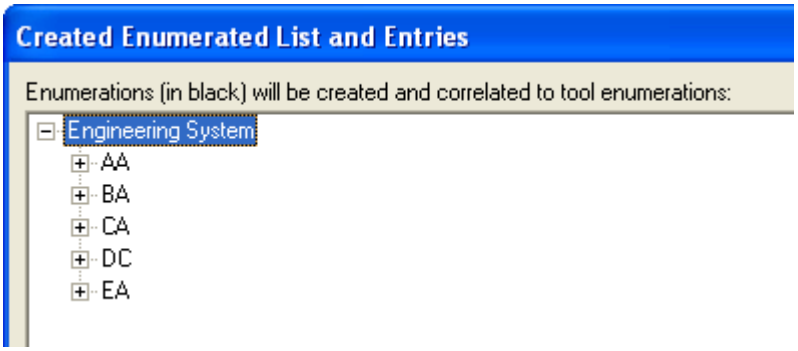
The enumerated list and entries that you created in the authoring tool appear in the **Map enumerated list definition** tree on the **Advanced** tab.

3. To create the equivalent list in the SmartPlant schema, click **New SmartPlant Enumerated List with Correlated Entries**.



TIP Once you click this button, the Schema Editor will create a list with the same entries in the SmartPlant schema and automatically map them to the corresponding entries in the map file. It will also prompt you with the option to create the retrieve mapping relationships at the same time. Click **Yes** to create the retrieve mapping relationships. Click **No** to create mapping for publish only.

4. Review the new SmartPlant schema enumerated list and entries in the **Created Enumerated List and Entries** dialog box, and then click **OK** to create the enumerated list.



TIPS

- You can right-click the list or list entries to make changes before you save the list and its entries in your schema extensions file.
- The software creates the SmartPlant enumerated list definition and creates the publish map for the authoring tool map enumerated list definition and entries to the new enumerated list definition and entries in the SmartPlant schema.

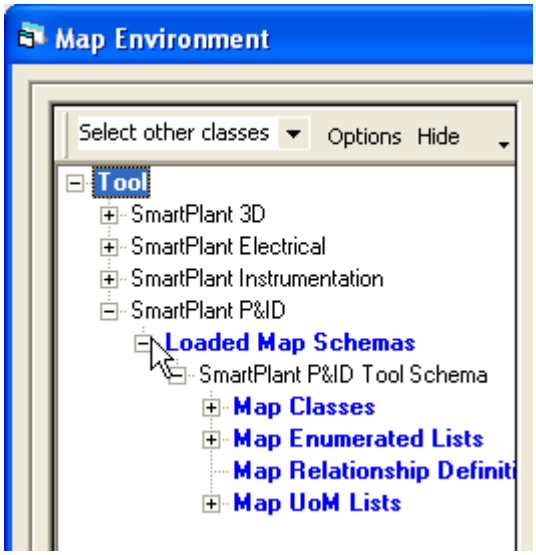
5. Click **OK** in the **Edit Map Enumerated List Definition** dialog box.

Add the New Property to the SmartPlant Schema

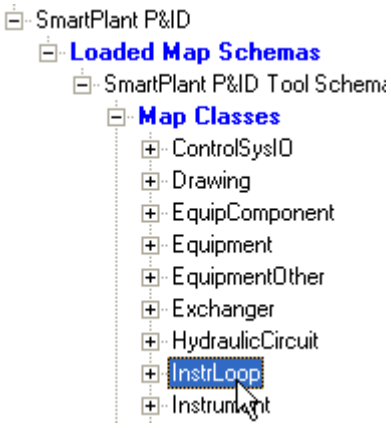
After you add the new enumerated list and its entries to the tool map schema and the SmartPlant schema, you can add the property that is scoped by the new enumerated list.

1. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.

2. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.



3. To add a new property, expand the **Map Classes** node, and double-click the tool map class to which you want to add the new property.

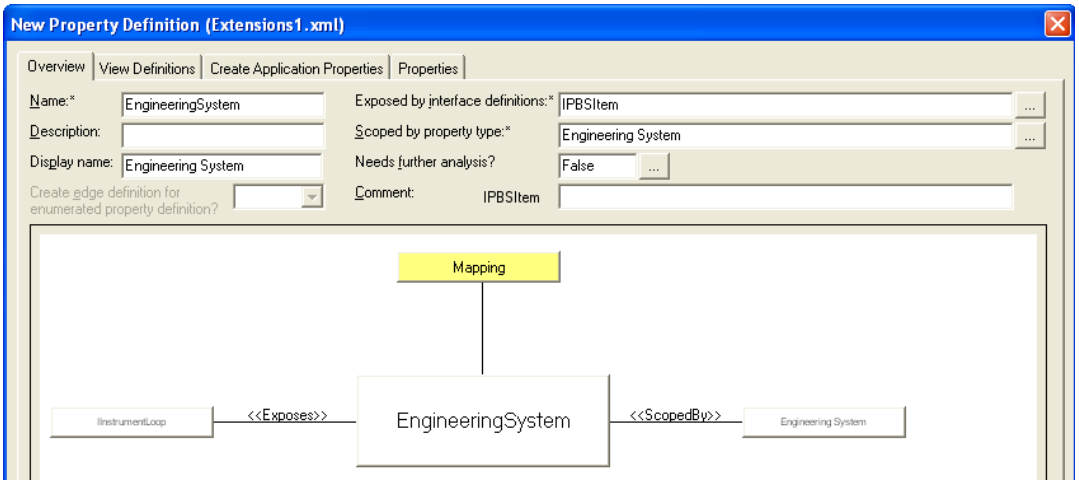


4. If you want to start by defining publish mapping for the new property, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated property, you can click the **Retrieve** tab instead. The procedure for adding properties and mapping is the same for both tabs.

5. Above **Unmapped SmartPlant properties** on the right side of the dialog box, click **New Property Definition** on the toolbar.

6. In the **New Property Definition** dialog box, define the new SmartPlant schema property, and click **OK** to create the property definition. For more information, see *Create a Property Definition* (on page 172).

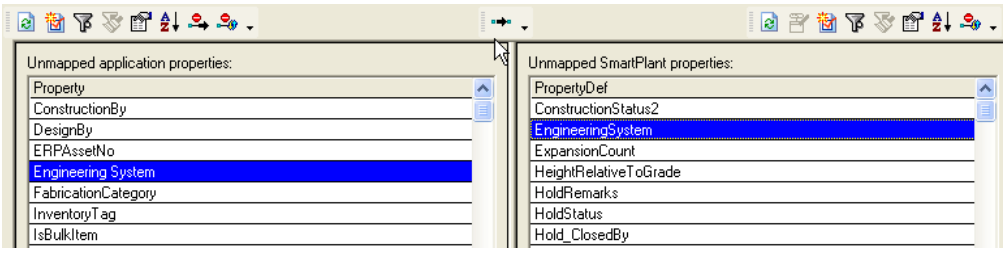


TIPS

- By default, the Schema Editor defines the interface definition that exposes the selected property definition based on the tool map class definition you selected. However, the default is only a suggestion based on context, and you can change the interface definitions that expose this property definition in the **Exposed by interface definitions** box.
- In the **Scoped by property type** box, be sure to select the new enumerated list you created previously as the property type for the new property.
- When you click **OK**, the new property appears in the **Unmapped SmartPlant properties** table in the **Edit Map Class Definition** dialog box.

Map Tool Property to the SmartPlant Schema Property

1. In the **Unmapped application properties** table, select the tool map property definition that you want to map.
2. In the **Unmapped SmartPlant properties** table, select the SmartPlant property definition that you want to map.
3. On the toolbar, click Map .



TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.
5. Click **OK** in the **Edit Map Class Definition** dialog box.

Save the Tool Map Schema and Schema Extensions File

1. Click **File > Save > Tool Schema File**. For some tools, saving the tool map schema also saves changes to the tool metadata database.
2. To save the changes to your schema extensions file, click **File > Save > Schema File**.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click **Yes** in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.
- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

See Also

Before You Begin Mapping for Tools with Metadata Adapters (on page 328)
Mapping with Metadata Adapters (on page 291)

Automatically Create and Map New Enumerated Entries Created in the Tool

This procedure is for tools that have metadata adapters. The steps also apply to map unit of measure list entries created in the authoring tool. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Create the New Enumerated List Entry in the Authoring Tool

- Use the appropriate authoring tool software to add the new enumerated list entry to an existing enumerated list in the authoring tool metadata database.

  For example, for SmartPlant P&ID or SmartPlant Electrical, add a new select list entry to the appropriate select list in Data Dictionary Manager.

  TIP Terminology for enumerated lists differs from tool-to-tool. Some authoring tools call enumerated lists select lists, picklists, codelists, or lookup tables. For more information, see the documentation for the authoring tool for which you want to add metadata.

Load the Tool Map Schema and Open the Map Enumerated List

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.

2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

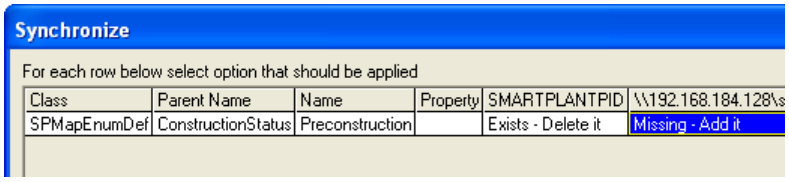
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

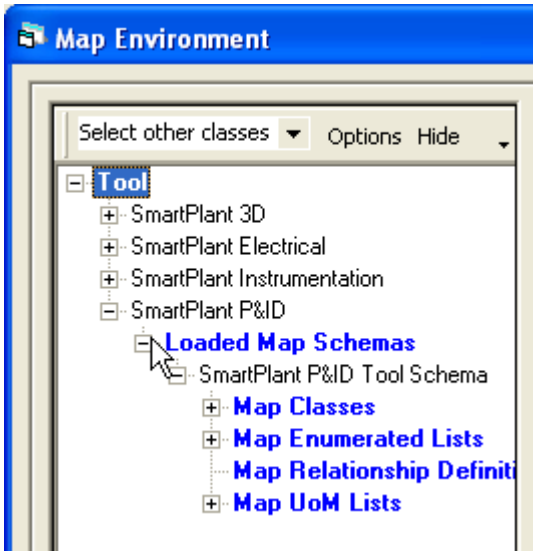
7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

   TIP The new enumerated list entry that you added in the authoring tool appears in the **Synchronization** dialog box to be added to the tool map schema file.



8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.

9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>*** **Tool Schema** node to display contents of the authoring tool map schema.

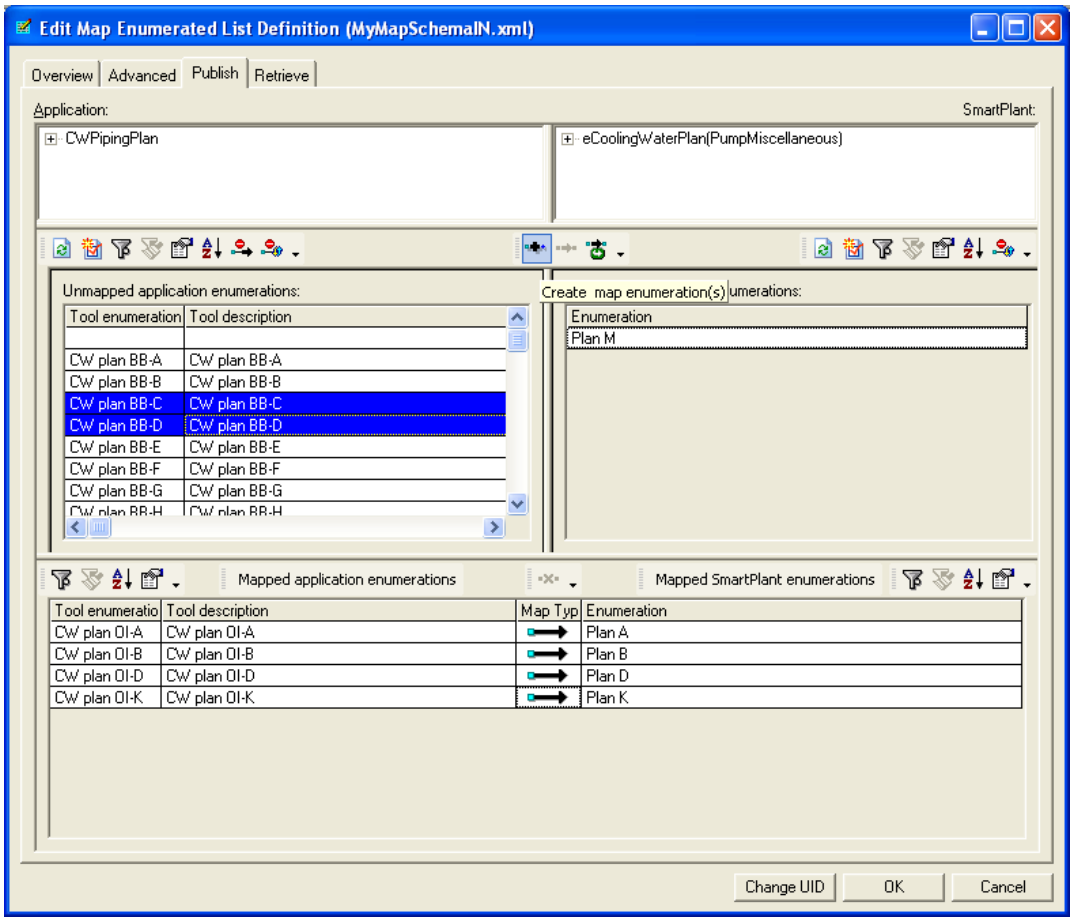


10. Expand the **Map Enumerated List** node and double-click the map enumerated list to which you added the new list entry.

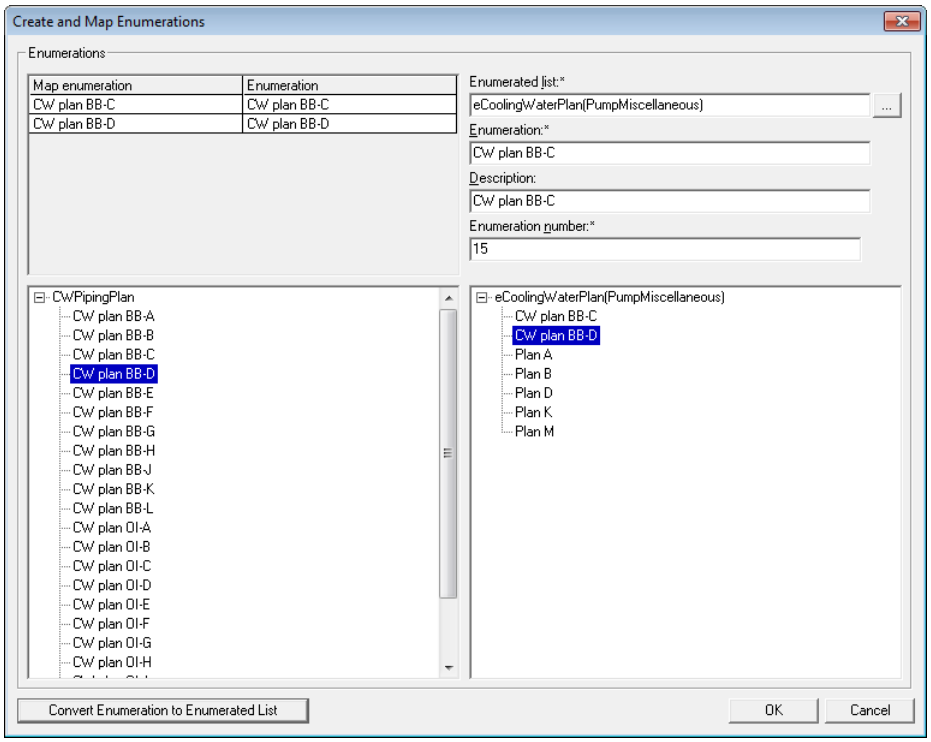
Automatically Create and Map Enumerations

1. Click the **Publish** or **Retrieve** tab in the **Edit Map Enumerated List Definition** dialog box.

2. Under **Unmapped application enumerations**, select the tool map enumerations that you want to create in the SmartPlant schema and map, and then click **Create and map enumerations**  on the toolbar.



3. In the **Create and Map Enumerations** dialog box, review the auto creation and mapping information, and click **OK** to create and map the enumerated list entries.



NOTE The **Convert Enumeration to Enumerated List** command changes the **Enumerated list** field to **Parent enumerated list** and the **Enumeration** field to **Enumerated list**. After this command is activated, the command name changes to **Convert Enumerated List to Enumeration**, which changes the fields back to the original state. You can use these different fields based on the type of information you want to add or create.

4. Click **OK** in the **Edit Map Enumerations** dialog box to save your changes.

Automatically Create and Map New Properties Created in the Tool

This procedure is for tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Create the Enumerated List, List Entries, and Property in the Authoring Tool

- Use the appropriate authoring tool software to add new enumerated list, enumerated list entries, and property that uses the enumerated list to the authoring tool metadata database. Be sure to set the data type to *string* for a string property.

  For example, for SmartPlant P&ID or SmartPlant Electrical, create a new select list, add entries to the select list, and then create a property associated with the new select list in the appropriate database table in Data Dictionary Manager. The new property should have a data type of *select list* and a relationship to the select list that you created.

  TIP Terminology for enumerated lists differs from tool-to-tool. Some authoring tools call enumerated lists select lists, picklists, codelists, or lookup tables. For more information, see the documentation for the authoring tool for which you want to add metadata.

Load the Tool Map Schema and Open a Map Class Definition

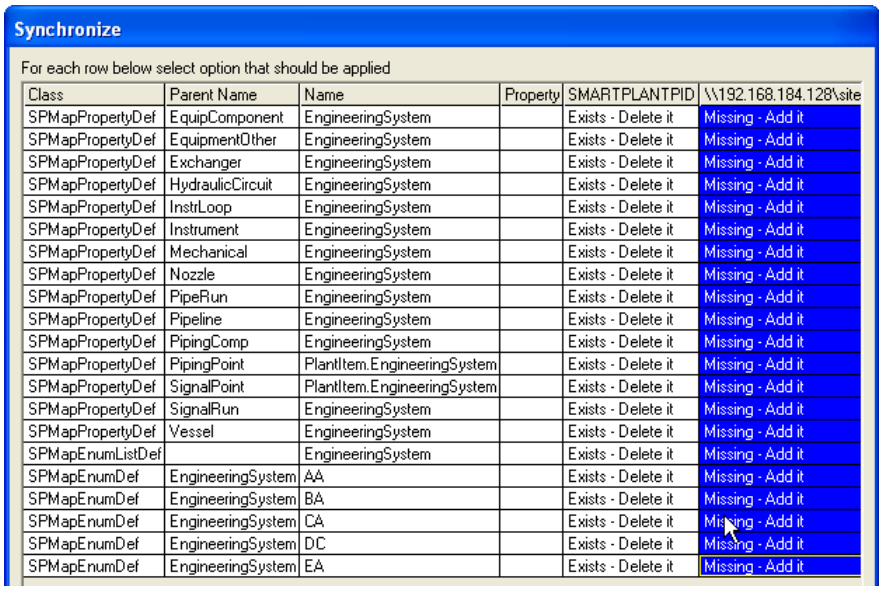
1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

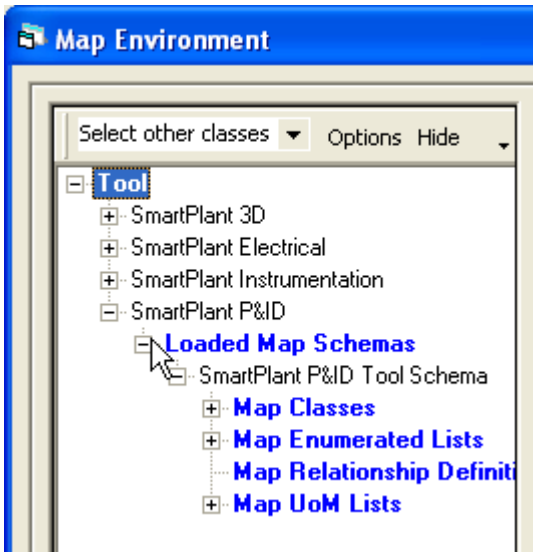
   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.
7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

TIP The new property, enumerated list, and list entries that you added in the authoring tool appear in the **Synchronization** dialog box to be added to the tool map schema file. As you can see in the following graphic, if you add a property to the Plant Item table in Data Dictionary Manager for SmartPlant P&ID, multiple map classes inherit that property, and therefore, multiple map classes are updated in the tool map schema during synchronize.



Class	Parent Name	Name	Property	SMARTPLANTPID	\\192.168.184.128\site
SPMapPropertyDef	EquipComponent	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	EquipmentOther	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Exchanger	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	HydraulicCircuit	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	InstrLoop	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Instrument	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Mechanical	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Nozzle	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipeRun	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Pipeline	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipingComp	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	PipingPoint	PlantItem.EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	SignalPoint	PlantItem.EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	SignalRun	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapPropertyDef	Vessel	EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapEnumListDef		EngineeringSystem		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	AA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	BA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	CA		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	DC		Exists - Delete it	Missing - Add it
SPMapEnumDef	EngineeringSystem	EA		Exists - Delete it	Missing - Add it

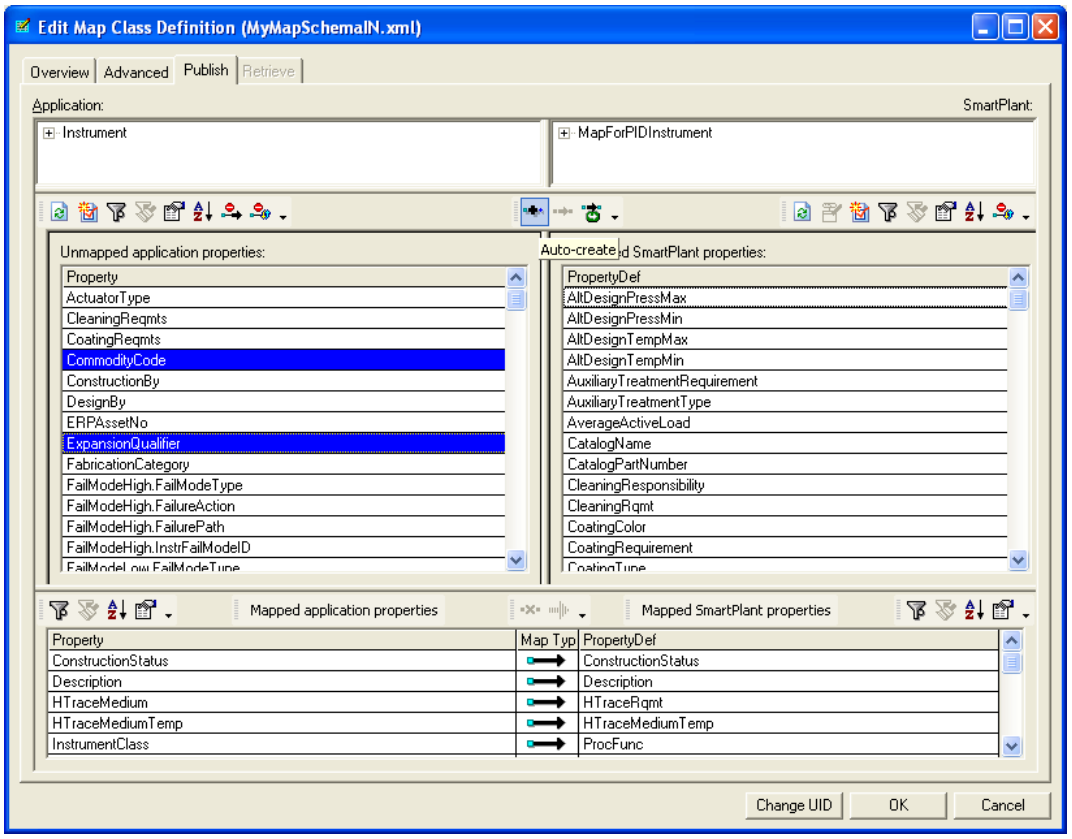
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.



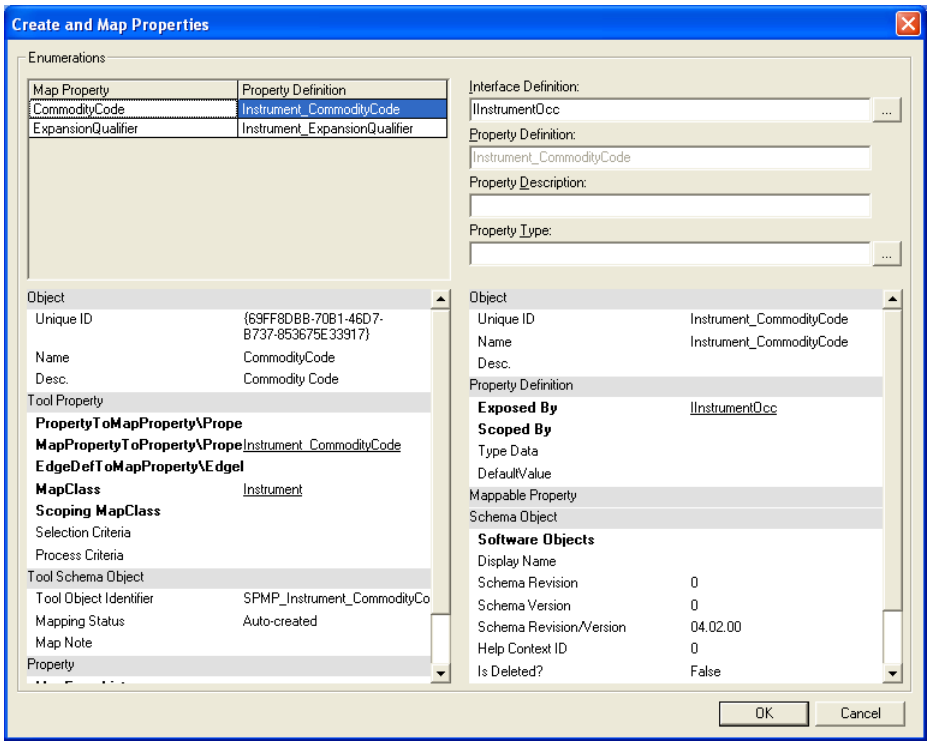
Automatically Create and Map the New Property

1. Click the **Publish** or **Retrieve** tab in the **Edit Map Class Definition** dialog box.

2. Under **Unmapped application properties**, select the tool properties that you want to create in the SmartPlant schema and map, and then click **Auto-create** on the toolbar.



3. In the **Create and Map Properties** dialog box, review the auto creation and mapping information, make any necessary changes to the displayed information, and click **OK** to create and map the properties.



4. Click **OK** in the **Edit Map Class Definition** dialog box to save your changes.

Automatically Correlate New Tool Map Objects with Schema Objects

The Schema Editor provides auto-correlation during mapping, which allows users to view a list of the best matches, based on a default algorithm, for mapping between map property definitions, map enumerated lists, and map unit of measure lists and existing objects in the schema. The **Auto-map** button is available on the **Publish** and **Retrieve** tabs on edit dialog boxes for map class definitions, map enumerated list definitions, and map unit of measure definitions.

After you create the map enumerated lists or entries, map unit of measure lists or entries, and map class definitions in the authoring tool, do the following:

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.

6. Click **OK** to set the configuration.

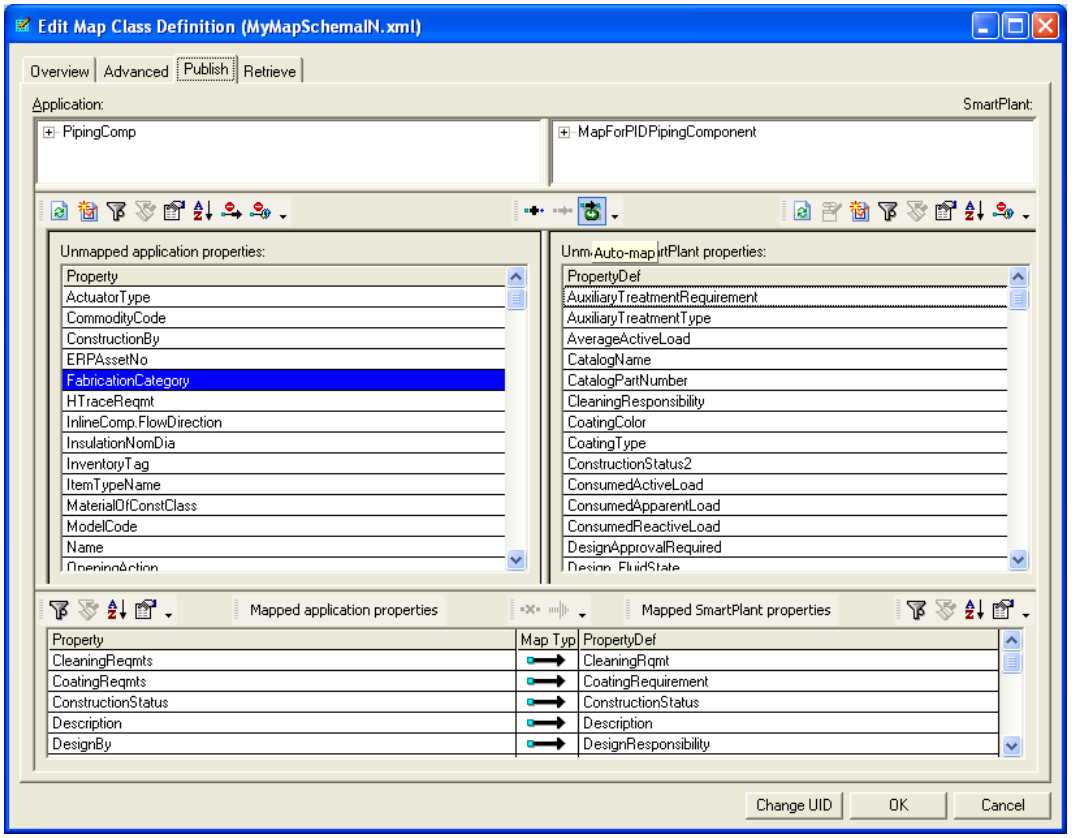
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

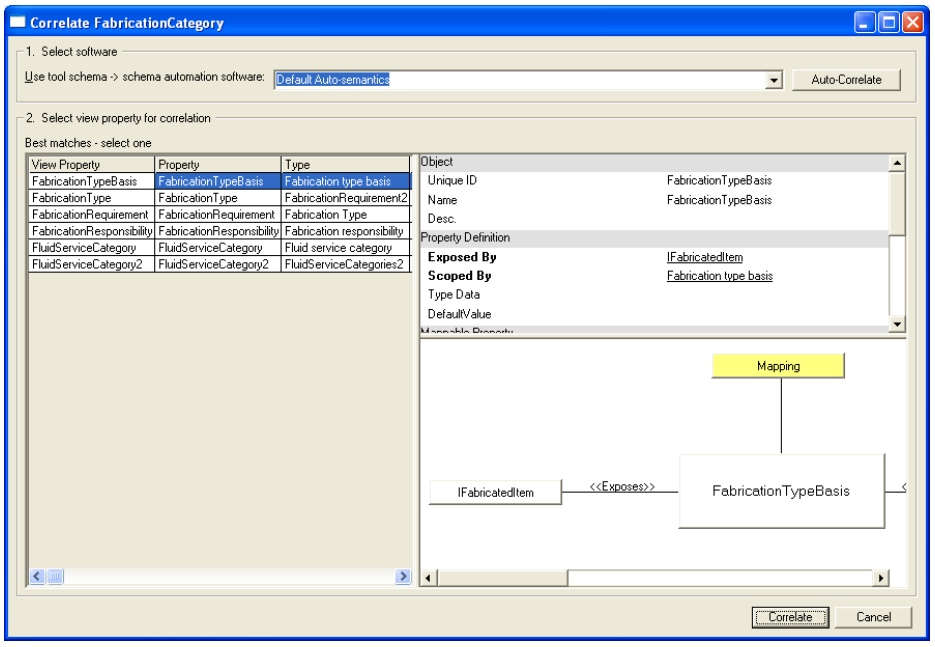
7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>*** **Tool Schema** node to display contents of the authoring tool map schema.
10. Expand the **Map Classes**, **Map Enumerated Lists**, or **Map UoM Lists** node and double-click the object that you want to automatically correlate with an existing object in the SmartPlant schema.
11. Click the **Publish** or **Retrieve** tab.

12. In the **Unmapped** table on the left, select the map properties, map enumerations, or map UoMs that you want the software to automatically correlate and map with existing SmartPlant schema objects, and click **Auto-map** .



NOTE If you select multiple enumerations under **Unmapped application properties** and then click **Auto-map**, the **Auto-Correlate Enumerations** dialog box appears. For more information, see *Automatically Correlate a Tool Enumerated List with a Schema Enumerated List* (on page 359).

13. In the **Correlate** dialog box, review the proposed matches in the SmartPlant schema, and select one to map.



14. Click **Correlate** to map the tool schema object to the selected SmartPlant schema object.

Automatically Correlate a Tool Enumerated List with a Schema Enumerated List

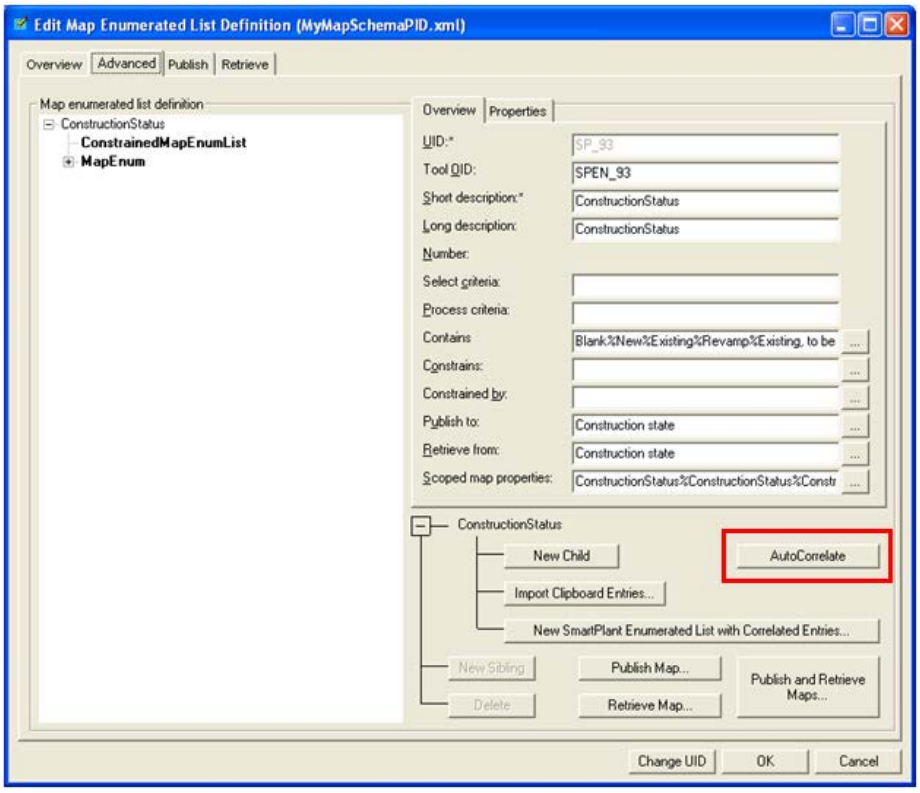
1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

TIPS

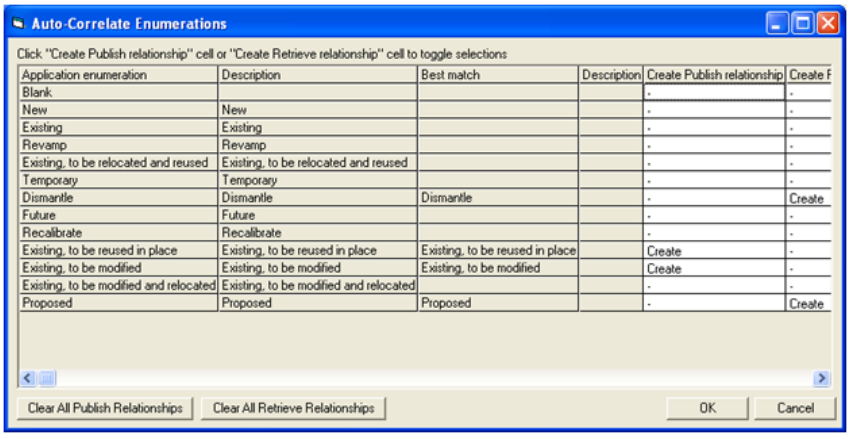
- For more options, click **Advanced** on the **Set Active Configuration** dialog box.
- By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
- If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).
8. Expand the **SPMapEnumListDef** node, then double-click an enumerated list to correlate.
9. On the **Advanced** tab, click **AutoCorrelate**. The entries in the map enumerated list are auto-correlated with the entries from the mapped-to enumerated list.



The **Auto-Correlate Enumerations** dialog box displays all possible relationships for publish or retrieve that can be created.



Application enumeration	Description	Best match	Description	Create Publish relationship	Create R
Blank				-	-
New	New			-	-
Existing	Existing			-	-
Revamp	Revamp			-	-
Existing, to be relocated and reused	Existing, to be relocated and reused			-	-
Temporary	Temporary			-	-
Dismantle	Dismantle	Dismantle		-	Create
Future	Future			-	-
Recalibrate	Recalibrate			-	-
Existing, to be reused in place	Existing, to be reused in place	Existing, to be reused in place		Create	-
Existing, to be modified	Existing, to be modified	Existing, to be modified		Create	-
Existing, to be modified and relocated	Existing, to be modified and relocated			-	-
Proposed	Proposed	Proposed		-	Create

Application enumeration	The Name for the application's enumeration.
Description	The Description for the application's enumeration.
Best match	The Name for the enumeration in the schema that best matches the application's enumeration.
Description	The Description for the enumeration in the schema that best matches the application's enumeration.
Create Publish relationship	Indicates if a publish relationship (MapEnumToEnum) should be created between the application's enumeration and the best match enumeration. The possible values for each cell in this column are Create Don't create "-" (not applicable)
Create Retrieve relationship	Indicates if a retrieve relationship (EnumToMapEnum) should be created between the application's enumeration and the best match enumeration. The possible values for each cell in this column are Create Don't create "-" (not applicable)

10. In either the **Create Publish Relationship** or **Create Retrieve Relationship**, click in the cell and select **Create**, **Don't Create**, or "**-**".
    - To set all publish relationships to **Don't Create**, click **Clear All Publish Relationships**.
    - To set all retrieve relationships to **Don't Create**, click **Clear All Retrieve Relationships**.

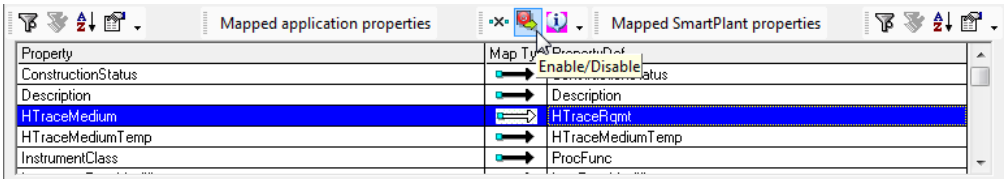
Disable or Enable an Existing Property Relationship in the Schema

You can disable mapped properties in the schema with the **Enable/Disable** command. This allows you to ignore the relationship between a tool and SmartPlant schema property, or enable a previously disabled relationship.

Disable an existing property

1. In the **Mapped application properties** table, select the tool map property definition you want to disable.

2. On the toolbar, click **Enable/Disable** .



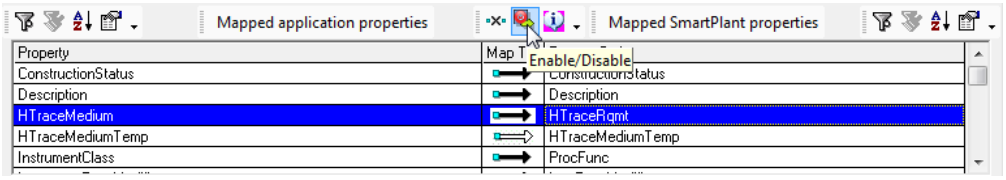
3. If you want to disable both publish and retrieve mapping for the property, click the **Retrieve** tab, and repeat steps 1-3 of this procedure.

4. Click **OK**.

Enable an existing disabled property

1. In the **Mapped application properties** table, select the tool map property definition you want to enable.

2. On the toolbar, click **Enable/Disable** .



3. If you want to enable both publish and retrieve mapping for the property, click the **Retrieve** tab, and repeat steps 1-3 of this procedure.

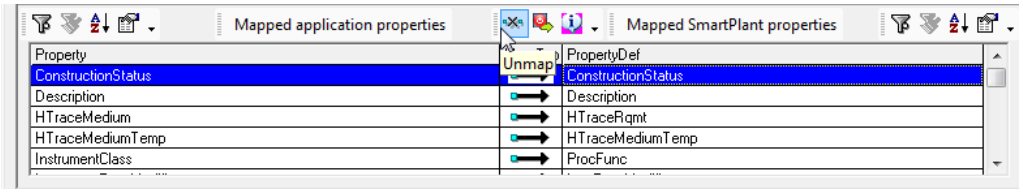
4. Click **OK**.

Unmap Existing Properties in the Schema

Unmapping a mapped property removes the relationship between the tool and SmartPlant schema properties. Unmapping a mapped property is only recommended if you never need to publish or retrieve the relationship again. For example, if the initial mapping is incorrect, the correct solution is to unmap the relationship and create a new, correct mapping relationship.

1. In the **Mapped application properties** table, select the tool map property definition that you want to unmap.

2. On the toolbar, click **Unmap** .



IMPORTANT Unmapping a relationship is a permanent action. The only way to restore the map relationship is to manually add the relationship with the **Map** command.

3. If you want to remove both publish and retrieve mapping for the property, click the **Retrieve** tab, and repeat steps 1-3 of this procedure.

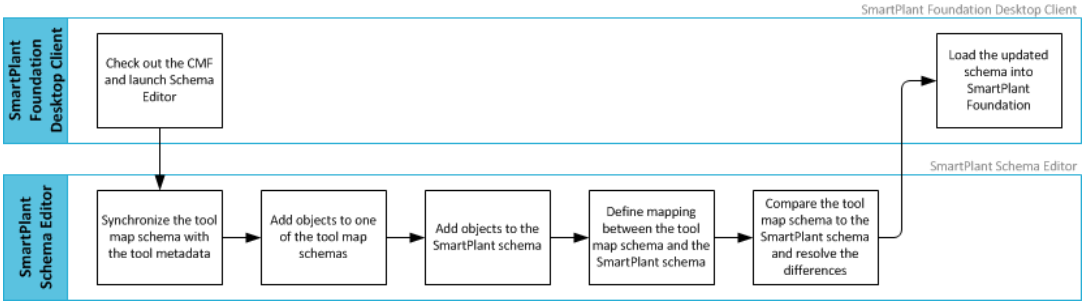
4. Click **OK**.

NOTE The recently unmapped properties are displayed in the **Unmapped application properties** and **Unmapped SmartPlant properties** tables.

Mapping New Objects Originating in the SmartPlant Schema

The following graphic provides an overview of the work process for mapping new objects when you start in the SmartPlant schema via the Schema Editor. Using this work process allows you to propagate additions out to the other authoring tool database from inside the Schema Editor. The graphic covers the work process for adding and mapping objects that do not already exist in the tool database, tool map schema, or SmartPlant schema. If the object already exists in the tool map schema or the SmartPlant schema, you can skip the steps that involve adding new objects in the work process.

This approach is most useful when you want to add a new enumerated entry or enumerated list that you want to add to all the authoring tools and the SmartPlant schema. Intergraph recommends starting in the SmartPlant schema when you define new enumerated lists and add enumerated list entries to existing enumerated lists.



What do you want to do?

- *Check Out the CMF and Launch Schema Editor* (on page 364)
- *Synchronize the Tool Map Schema with the Tool Map Schema* (on page 364)
- *Add Objects to One of the Tool Map Schemas* (on page 364)
- *Add Objects to the SmartPlant Schema* (on page 365)
- *Define Mapping Between the Tool Map Schema and the SmartPlant Schema* (on page 365)
- *Compare the Tool Map Schema to the SmartPlant Schema and Resolve the Differences* (on page 365)
- *Load the Updated Schema into SmartPlant Foundation* (on page 366)

Check Out the CMF and Launch Schema Editor

Before you can make modifications to the SmartPlant schema, must check the CMF file out of the SmartPlant Foundation Desktop Client.

- Check out the SmartPlant schema CMF file from the SmartPlant Foundation Desktop Client.
- Make sure your scope is set to the plant, right-click the new revision of the CMF file in SmartPlant Foundation, and click **Launch Schema Editor**. This provides the Schema Editor with the appropriate plant and SmartPlant Foundation site for connection.
- Select the schema version or versions that you want to view in the **Specify configurations to display** window of the **Active Configurations** dialog box.
- Click the **Advanced** button on the **Active Configurations** dialog box to view the **Set Configurations for new objects** window, where you can specify what version or versions of the schema changes should be written to. For example, to modify the 2016 schema, select the 2016 configuration for new objects.

NOTE By default, all configurations are checked to allow forward and backward compatibility between schema versions when creating new objects or modifying existing objects.

Opening the CMF file also allows you to see which authoring tools that are registered with the SmartPlant Foundation plant have available metadata adapters.

Synchronize the Tool Map Schema with the Tool Map Schema

Synchronization allows you to keep the tool map schema and the tool metadata in the authoring tool in synch without requiring manual steps. If you are planning to add new objects to the SmartPlant schema first, you may not see any inconsistencies to be resolved. However, the synchronization process is required before you load the tool map schema to begin mapping in the Schema Editor.

In the Schema Editor, click the **File > SmartPlant > Edit Engineering Tools** command to select the authoring tool or tools that you want to defining mapping for. When you select one or more authoring tools to load, select the **Load Map Schema** check box and the **Connect to application schema** check box. The synchronization between the tool map schema .XML file and the tool metadata, located in the tool database, begins when you click **Finish**.

After the Schema Editor connects to the tool database through the metadata adapter, the software displays a list of inconsistencies between the tool map schema and the tool metadata. When you add new properties, enumerated lists, or enumerated entries in the authoring tool, these additions show up as inconsistencies in the **Synchronization** dialog box.

For more details about using the metadata adapter to synchronize a tool metadata and tool map schema, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

Add Objects to One of the Tool Map Schemas

You can add objects the tool map schema that you synchronized previously using the **Map Environment** window in the Schema Editor. This allows you to define an object in one authoring tool map schema, save it in the tool metadata database, and then copy that change to the other authoring tools using compare.

For some authoring tools, such as SmartPlant Instrumentation, changes to the metadata in the tool database are made automatically when you add the object to the tool map schema. For

other authoring tools, changes are only saved when you click **File > Save > Tool Schema File** command in the Schema Editor to save the tool map schema. This allows you to cancel your changes when you are adding new tool properties, enumerated lists, or enumerated entries before you save the tool map schema.

Add Objects to the SmartPlant Schema

First, verify that the objects that you want to map exist in the SmartPlant schema. If the objects do not exist, you can create them in the SmartPlant schema in the **Map Environment** window. For more information about extending the delivered SmartPlant schema, see *Extending the SmartPlant Schema* (on page 146).

Define Mapping Between the Tool Map Schema and the SmartPlant Schema

After you define the objects that you want to map in the tool map schema and the SmartPlant schema, you can define the publish and retrieve mapping for those objects. Objects can have different mapping defined for publish and retrieve.

NOTES

- Some authoring tools may have additional steps for performing the mapping between the authoring tool and the SmartPlant schema. For mapping rules for each authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.
- The **Display name** field allows you to assign a unique display name to a definition in the tool map schema. This field is available in the following dialog boxes: **Edit Map Class Definition**, **Edit Map Property Definition**, **Map Relationship Definition**, **Map Enumerated List Definition**, **Map Enumeration**, **Map Unit of Measure List Definition**, **Map Unit of Measure**, and **Map Edge Definition**.

Compare the Tool Map Schema to the SmartPlant Schema and Resolve the Differences

After you add the new object to the SmartPlant schema, you can use the **File > SmartPlant > Compare (Active Tool Schema)** with Schema command to perform a comparison between the tool map schema for each of the other authoring tools and the SmartPlant schema. Because the tool map schema and the tool metadata were synchronized at the beginning of the work process, you are essentially comparing the SmartPlant schema with the tool metadata.

The **View Comparison Results** window uses color coding to show you which objects are the same (green) in both and are completely mapped, which ones are different or have not been completely mapped (blue), and which ones have not been mapped at all (red) in the tool map schema and the SmartPlant schema. By default, the comparison results only show those objects that are different or that are unmapped.

For those objects that appear in blue in the **View Comparison Results** window, you can resolve these differences by double-clicking the node in the tree view to display the **Map Environment**. You can use the **Publish** and **Retrieve** tabs in the **Map Environment** to resolve differences by doing the following:

- Adding objects to tool map schema and metadata using the toolbar on the left side of the tab

- Mapping between the tool and SmartPlant schema by selecting objects on both sides of the dialog box and clicking the **Map** button in the middle of the tab.
- Defining properties that you do not want to map, which means that you do not want them to show up in the Unmapped application properties or the **Unmapped SmartPlant properties** tables on the **Publish** and **Retrieve** tabs
- Defining properties that you want to ignore during mapping, which can be used for internal application properties that are not required for mapping and that should not be synchronized in the tool map schema

You can also switch between authoring tools for comparison to make adding objects to the tool schema and metadata easier by loading multiple tool map schemas using the **File > SmartPlant > Edit Engineering Tool Options** and then setting another tool map schema as the active schema file.

You can repeat this process for each integrated authoring tool to propagate the change you made for one tool into all the other authoring tools for which the change is relevant.

Load the Updated Schema into SmartPlant Foundation

Loading the updated schema into SmartPlant Foundation involves the following high-level tasks:

- Checking the CMF file back into the Desktop Client.
- Running the **CMF File Manager** on the CMF file to create an XML file for loading.
- Importing the schema using the **Schema Import Wizard**.
- Regenerating the component schemas.

Mapping Objects Created in the SmartPlant Schema

Create and Map New Enumerated List Entries Starting in the SmartPlant Schema

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

Load the Tool Map Schema and Open the Map Enumerated List

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

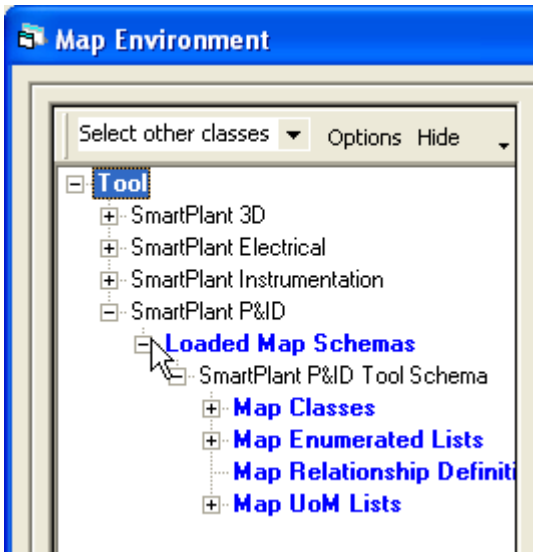
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.

- By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
- If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>*** **Tool Schema** node to display contents of the authoring tool map schema.



10. To add new entries to an existing enumerated list, expand the **Map Enumerated List** node and double-click the map enumerated list that you want to modify.

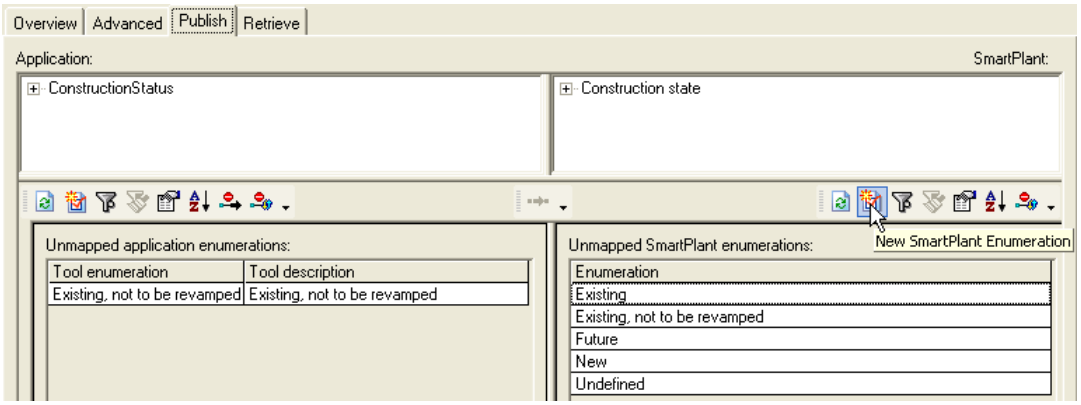
Add the New List Entries to the SmartPlant Schema

Before you add new entries to the SmartPlant schema, you need to verify that the entries or entries that have equivalent meanings do not already exist.

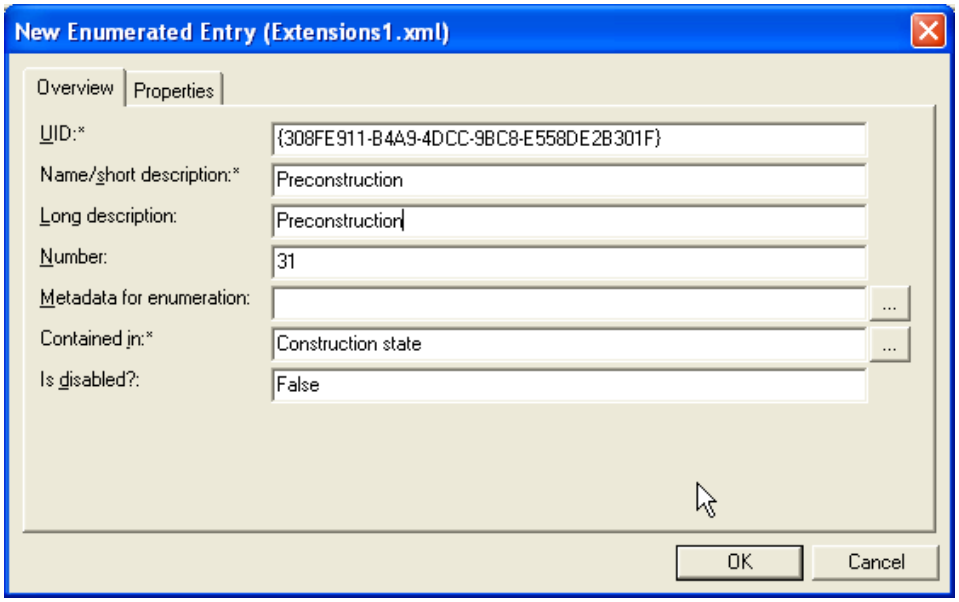
1. If you want to start by defining publish mapping for the new list entry, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated entry, you can click the **Retrieve** tab instead. The procedure for adding enumerated entries and mapping is the same for both tabs.

2. Above **Unmapped SmartPlant enumerations** on the right side of the dialog box, click **New SmartPlant Enumeration** on the toolbar.



3. In the **New Enumerated Entry** dialog box, type the name and long description for the new enumerated list entry, and click **OK**.

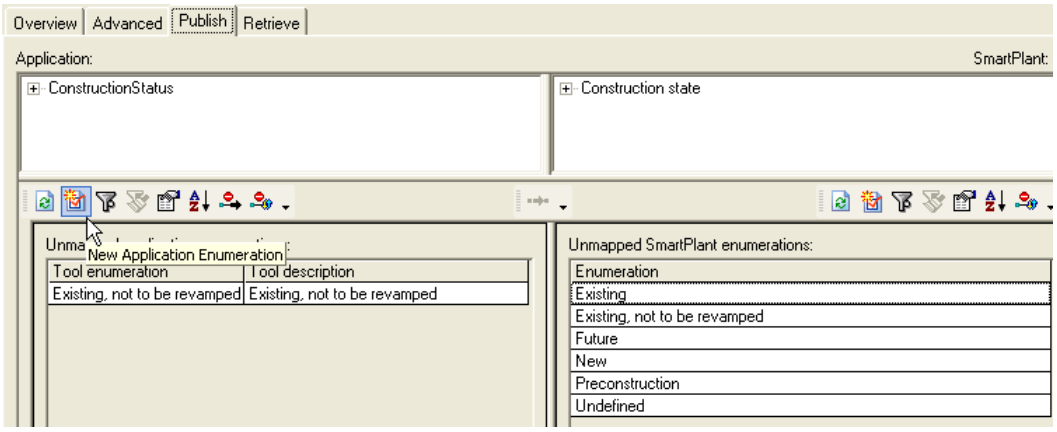


TIPS

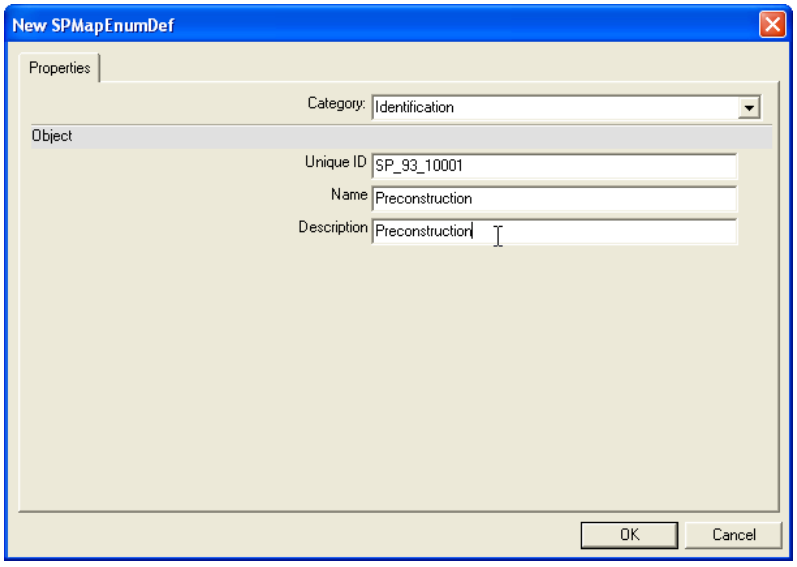
- The Schema Editor automatically provides number for the enumerated list entry. This number may need to be modified to match the corresponding codelist value if the entry will be mapped with Smart 3D.
- When you click **OK**, the new list entry appears in the **Unmapped SmartPlant enumerations** table in the **Edit Map Enumerated List Definition** dialog box.

Add the New List Entries to the Tool Map Schema and Metadata

1. Above **Unmapped application enumerations** on the left side of the dialog box, click **New Application Enumeration** on the toolbar.



2. In the **New SPMapEnumDef** dialog box, type the name and long description for the new enumerated list entry, and click **OK**.



TIPS

- The metadata adapter automatically fills in the appropriate unique identifier for the enumerated list entry.
- When you click **OK**, the new list entry appears in the **Unmapped SmartPlant enumerations** table in the **Edit Map Enumerated List Definition** dialog box.

IMPORTANT For some authoring tools, such as SmartPlant Instrumentation, changes to the metadata in the tool database are made automatically when you add the object to the tool map schema. For other authoring tools, changes are only saved when you click **File > Save > Tool Schema File** command in the Schema Editor to save the tool map schema. This allows you to cancel your changes when you are adding new tool properties, enumerated lists, or enumerated

entries before you save the tool map schema. For more information, see *Mapping with Metadata Adapters* (on page 291).

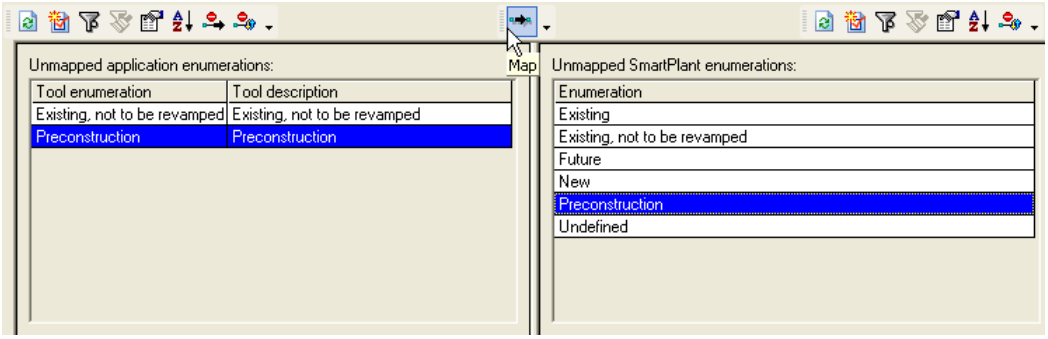
Map Tool List Entries to SmartPlant Schema List Entries

1. In the **Unmapped application enumerations** table, select the tool enumerated list entry that you want to map.

   TIP For Smart 3D, mapping for enumerated list entries is done using the index number assigned to the enumerated list entry. Therefore, you are not required to do any mapping in the **Map Environment** for Smart 3D enumerated list entries once the appropriate entries are added to the SmartPlant schema and the tool map schema and metadata.

2. In the **Unmapped SmartPlant enumerations** table, select the SmartPlant enumerated list entry that you want to map.

3. On the toolbar, click **Map** .



   TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.

5. Click **OK** in the **Edit Map Enumerated List Definition** dialog box.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

Save the Tool Map Schema and Schema Extensions File

- Click **File > Save > All Modified Files** to save the SmartPlant schema extensions and the tool map schema. For some tools, saving the tool map schema also saves changes to the tool metadata database.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click **Yes** in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.

- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

Propagating the Enumerated List Entry to Other Authoring Tools Using Compare

- After you add new objects to the SmartPlant schema, do a compare with other authoring tool map schemas to identify changes that need to be made in the other authoring tools and then add new objects and define mapping in the **Map Environment**. For more information, see *Compare a Tool Map Schema to the SmartPlant Schema* (on page 385).

Create and Map New String Properties Starting in the SmartPlant Schema

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

IMPORTANT When you add new properties starting in the SmartPlant schema, the authoring tool metadata adapter uses classes exposed in the tool map schema to determine where the property can be placed in the authoring tool metadata. In some cases, the tool map schema does not totally represent the tool data model, and because of this, classes available to add properties to in the authoring tool may not be available in the tool map schema. For example, if you want to add a property to the Plant Item table in the SmartPlant P&ID data dictionary so that all plant item types in SmartPlant P&ID inherit the property, you must define the new property in SmartPlant P&ID Data Dictionary Manager, not in the Schema Editor. However, if you want to define a new property on a map class that exists in the tool map schema, you can use the following procedure to do so.

Load the Tool Map Schema and Open a Map Class Definition

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

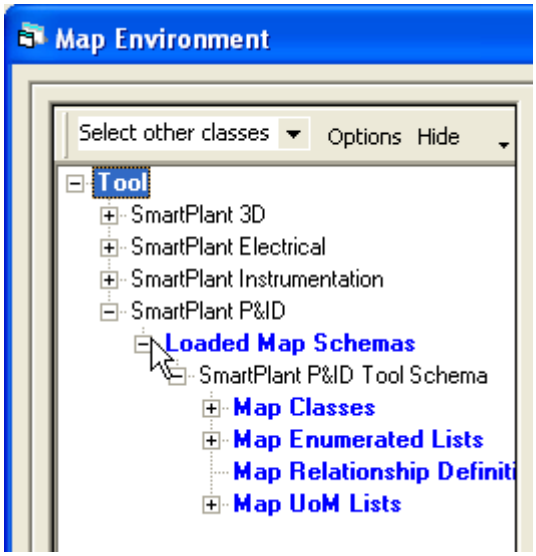
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be

prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the ***<Authoring Tool>*** **Tool Schema** node to display contents of the authoring tool map schema.



10. To add a new property, expand the Map Classes node, and double-click the tool map class to which you want to add the new property.

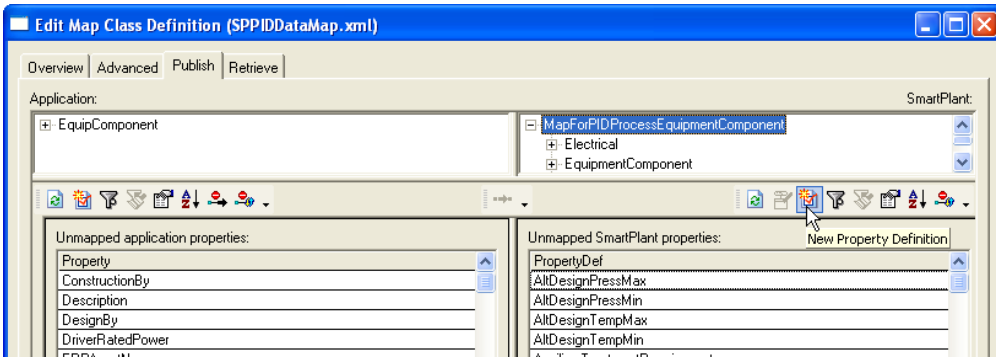
Add the New Property to the SmartPlant Schema

Before you add new properties to the SmartPlant schema, you need to verify that the properties or properties that have equivalent meanings do not already exist.

1. If you want to start by defining publish mapping for the new property, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated property, you can click the **Retrieve** tab instead. The procedure for adding properties and mapping is the same.

2. Above **Unmapped SmartPlant properties** on the right side of the dialog box, click **New SmartPlant Property Definition** on the toolbar.



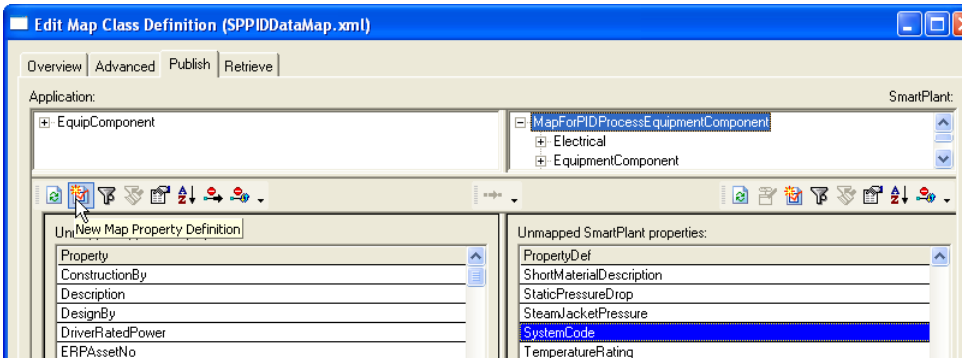
3. In the **New Property Definition** dialog box, define the new SmartPlant schema property, and click **OK** to create the property definition. For more information, see *Create a Property Definition* (on page 172).

TIPS

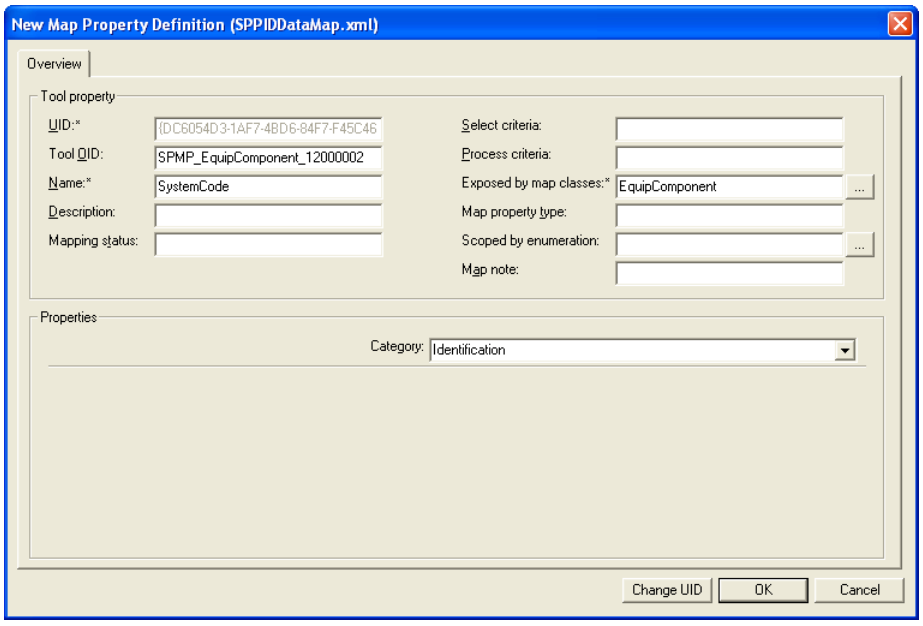
- In the **Scoped by property type** box, be sure to select **string** as the property type for the new string property.
- When you click **OK**, the new property appears in the **Unmapped SmartPlant properties** table in the **Edit Map Class Definition** dialog box.

Add the New Property to the Tool Map Schema and Metadata

1. Above **Unmapped application properties** on the left side of the dialog box, click **New Map Property Definition** on the toolbar.



2. In the **New Map Property** dialog box, type the name and any other information that you want to define for the new map property definition, and click **OK** to create it.



TIPS

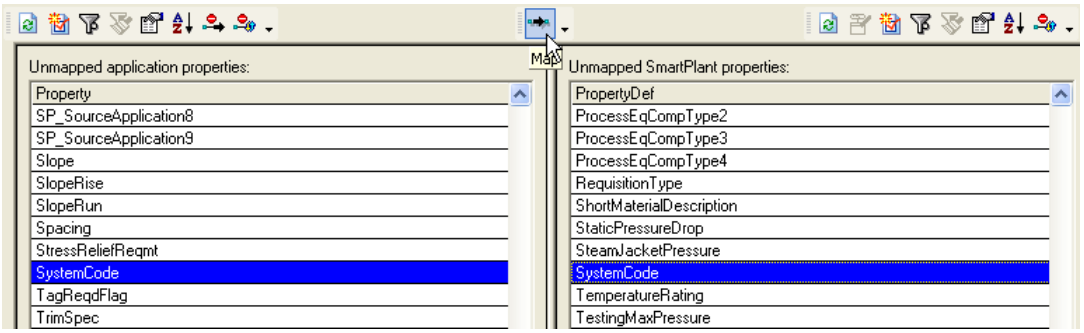
- By default, the authoring tool metadata adapter supplies the tool identifier (OID) and the name for the map property definition. You can change the default name, which is always a number, by selecting the existing number and typing the name you want.
- In the **Select criteria** box, you can define the criteria that the tool adapter uses to select or identify objects for this tool property definition. Some authoring tool metadata adapters define this information by default. See the mapping rules for your authoring tool for how to set the value for this property.
- In the **Process criteria** box, you can define the criteria that the adapter uses to process objects that are instances of this tool property definition. Some authoring tool metadata adapters define this information by default. See the mapping rules for your authoring tool for how to set the value for this property.

  For example, if the tool adapter must do a conversion on an enumerated list or the unit of measure for a property before the property is published, use the **Process criteria** box to specify the method used for the conversion, for example, EFAdapter.EnumConversions.
- By default, the software creates the relationship between the map class definition you selected to edit and the new map property definition. You can see which map class definition exposes the map property definition in the **Exposed by map classes** box.
- You can change the property category for the new map property definition in the **Category** box. Changing the property category displays other properties that you can define for the new map property definition.
- When you click **OK**, the new map property definition is added to the **Unmapped application properties** table in the **Edit Map Class Definition** dialog box.

IMPORTANT For some authoring tools, such as SmartPlant Instrumentation, changes to the metadata in the tool database are made automatically when you add the object to the tool map schema. For other authoring tools, changes are only saved when you click **File > Save > Tool Schema File** command in the Schema Editor to save the tool map schema. This allows you to cancel your changes when you are adding new tool properties, enumerated lists, or enumerated entries before you save the tool map schema. For more information, see *Mapping with Metadata Adapters* (on page 291).

Map Tool Property to the SmartPlant Schema Property

1. In the **Unmapped application properties** table, select the tool map property definition that you want to map.
2. In the **Unmapped SmartPlant properties** table, select the SmartPlant property definition that you want to map.
3. On the toolbar, click Map .



TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.
5. Click **OK** in the **Edit Map Class Definition** dialog box.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

Save the Tool Map Schema and Schema Extensions File

- Click **File > Save > All Modified Files** to save the SmartPlant schema extensions and the tool map schema. For some tools, saving the tool map schema also saves changes to the tool metadata database.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click **Yes** in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.

- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

Propagating the New Property to Other Authoring Tools Using Compare

- After you add new objects to the SmartPlant schema, do a compare with other authoring tool map schemas to identify changes that need to be made in the other authoring tools and then add new objects and define mapping in the **Map Environment**. For more information, see *Compare a Tool Map Schema to the SmartPlant Schema* (on page 385).

Create and Map New Properties with Enumerated Lists Starting in the SmartPlant Schema

The following procedure applies to authoring tools that have metadata adapters. For more information, see *Before You Begin Mapping for Tools with Metadata Adapters* (on page 328).

IMPORTANT When you add new properties starting in the SmartPlant schema, the authoring tool metadata adapter uses classes exposed in the tool map schema to determine where the property can be placed in the authoring tool metadata. In some cases, the tool map schema does not totally represent the tool data model, and because of this, classes available to add properties to in the authoring tool may not be available in the tool map schema. For example, if you want to add a property to the Plant Item table in the SmartPlant P&ID data dictionary so that all plant item types in SmartPlant P&ID inherit the property, you must define the new property in SmartPlant P&ID Data Dictionary Manager, not in the Schema Editor. However, if you want to define a new property on a map class that exists in the tool map schema, you can use the following procedure to do so.

Load the Tool Map Schema and Open a Map Class Definition

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
6. Click **OK** to set the configuration.

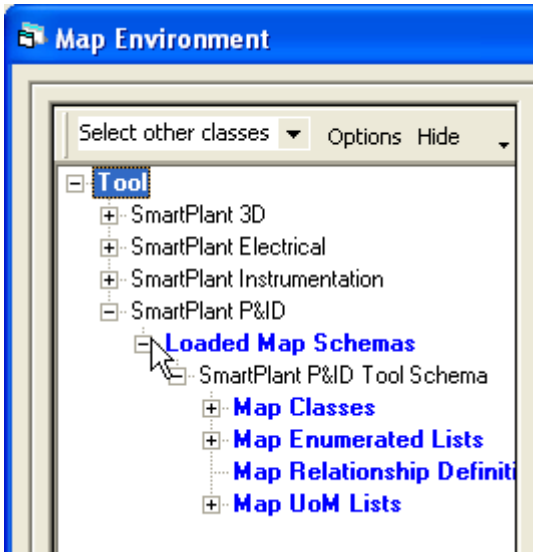
   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be

prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).
8. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
9. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.

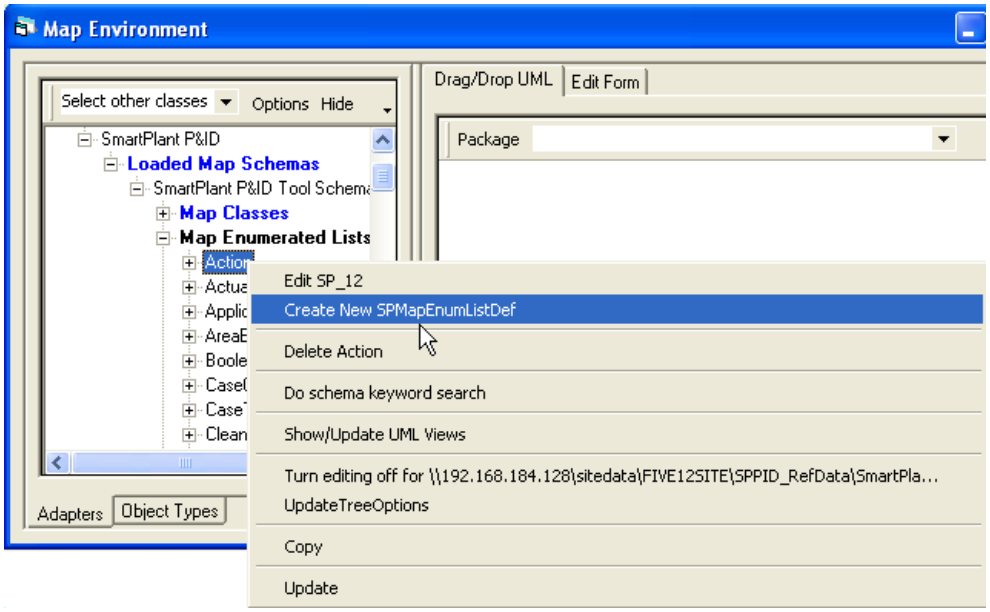


Add the New Enumerated List to the Tool Map Schema and SmartPlant Schema

Before you add a new map property definition to the authoring tool map schema, you should add the map enumerated list that scopes the property definition. Adding the map enumerated list definition also allows you to automatically create the equivalent enumerated list definition and entries in the SmartPlant schema. Before you create the list in the SmartPlant schema, you need to verify that the enumerated list or a list that has an equivalent meaning does not already exist.

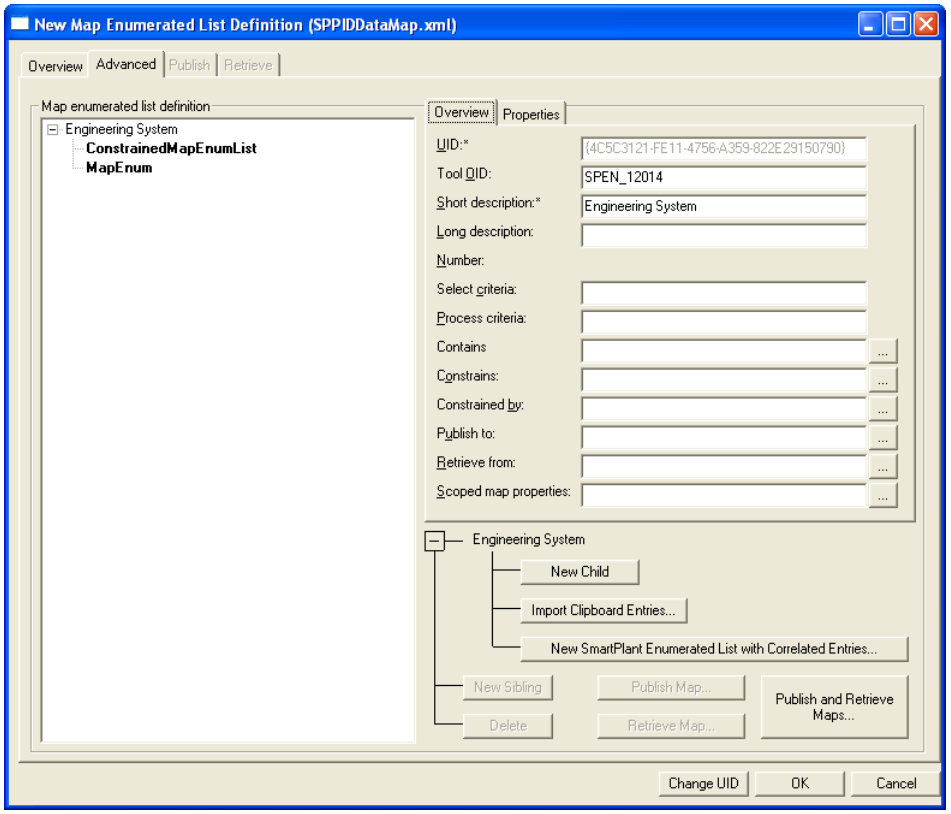
1. To add a new enumerated list, expand the **Map Enumerated Lists** node, and right-click any map enumerated list in the tree view.

2. On the shortcut menu, click **Create New SPMapEnumListDef**.



3. In the **New Map Enumerated List Definition** dialog box, type the name of the new map enumerated list definition in the **Name** box on the **Overview** tab.
4. Click the **Advanced** tab.

5. In the **Short Description** box, type a name for the new map enumerated list.



TIP By default, the authoring tool metadata adapter supplies the tool identifier (OID) and the short description for the map enumerated list. You can change the default name, which is always a number, by selecting the existing number and typing the name you want.

6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool enumerated list definition and its entries. See the mapping rules for your authoring tool for how to set the value for this property.

7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool enumerated list definition and its enumerations. See the mapping rules for your authoring tool for how to set the value for this property.

8. To create an enumerated list entry (MapEnum) for this map enumerated list, click **New Child** in the **New Map Enumerated List Definition** dialog box.

   TIP If you have a spreadsheet containing your enumerated list entries, you can copy them to the Clipboard and then add them to the enumerated list using the **Import Clipboard Entries** button. For more information, see *Import Enumerated List Entries from a Spreadsheet* (on page 174).

9. In the **Short description** box, type the name of the enumerated list entry.

   TIPS

   - By default, the authoring tool metadata adapter supplies the tool identifier (OID), the name, and the secondary numeric key for the map enumerated list entry. You can change the default name, which is always a number, by selecting the existing number

and typing the name you want. However, you should not modify the value in the **Number** box that is supplied by the metadata adapter.

- When you move out of the **Short description** box, the new enumerated list entry is automatically added to the **Map enumerated list definition** tree.

10. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool enumerated list entry. See the mapping rules for your authoring tool for how to set the value for this property.
11. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool enumerated list entry. See the mapping rules for your authoring tool for how to set the value for this property.
12. To create additional list entries, click **New Sibling**, and repeat steps 7-11 for each entry that you want to add.
13. In the **Map enumerated list definition** tree, click the name of the map enumerated list definition.
14. To create the equivalent list in the SmartPlant schema, click **New SmartPlant Enumerated List with Correlated Entries**.
15. Review the new SmartPlant schema enumerated list and entries in the **Created Enumerated List and Entries** dialog box, and then click **OK** to create the enumerated list.

    TIPS

    - You can right-click the list or list entries to make changes before you save the list and its entries in your schema extensions file.
    - The software creates the SmartPlant enumerated list definition and creates the publish map for the authoring tool map enumerated list definition and entries to the new enumerated list definition and entries in the SmartPlant schema.

16. Click **OK** in the **New Map Enumerated List Definition** dialog box.

    TIP The software prompts you to enter any information that is required by the metadata adapter, such as the long description. Required information varies by authoring tool.

Define Retrieve Mapping for the New Enumerated List

You can also define retrieve mapping for the new enumerated list. Not all enumerated lists require both publish and retrieve mapping. If you do not want to define retrieve mapping, skip to *Import Enumerated List Entries from a Spreadsheet* (on page 174).

1. Click the **Overview** tab.
2. Beside the **Retrieve from** box, click ... and select the SmartPlant enumerated list definition that maps to this tool enumerated list for retrieve.

   TIP When you define the SmartPlant schema enumerated list for retrieve, the **Retrieve** tab becomes available.

3. In the **Unmapped application enumerations** table, select the tool map enumerated list entry that you want to map.
4. In the **Unmapped SmartPlant enumerations** table, select the SmartPlant enumerated list entry that you want to map.
5. On the toolbar, click Map .

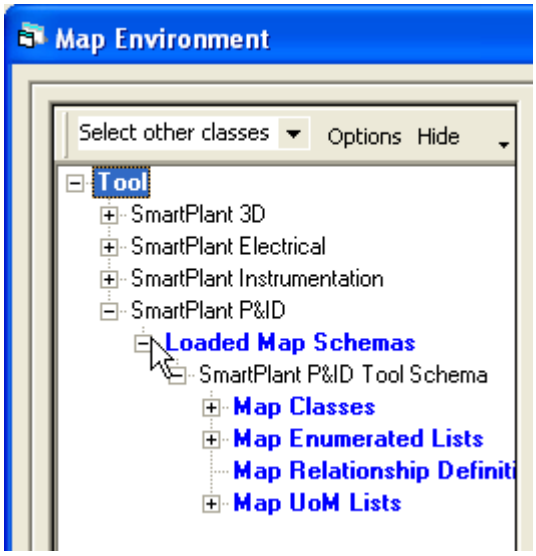
TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

6. Repeat steps 3-5 for each enumerated list entry in the tool map enumerated list definition.
7. Click **OK** in the **New Map Enumerated List Definition** dialog box.

Add the New Property to the SmartPlant Schema

After you add the new enumerated list and its entries to the tool map schema and the SmartPlant schema, you can add the property that is scoped by the new enumerated list.

1. In the **Map Environment**, expand the **Tool** node, and then expand the node for the appropriate authoring tool.
2. Under the authoring tool node in the tree view, expand **Loaded Map Schemas** and the **<Authoring Tool> Tool Schema** node to display contents of the authoring tool map schema.



3. To add a new property, expand the **Map Classes** node, and double-click the tool map class to which you want to add the new property.
4. If you want to start by defining publish mapping for the new property, click the **Publish** tab.

   If you only want to define retrieve mapping for the enumerated property, you can click the **Retrieve** tab instead. The procedure for adding properties and mapping is the same for both tabs.
5. Above **Unmapped SmartPlant properties** on the right side of the dialog box, click **New Property Definition** on the toolbar.
6. In the **New Property Definition** dialog box, define the new SmartPlant schema property, and click **OK** to create the property definition. For more information, see *Create a Property Definition* (on page 172).

   TIPS

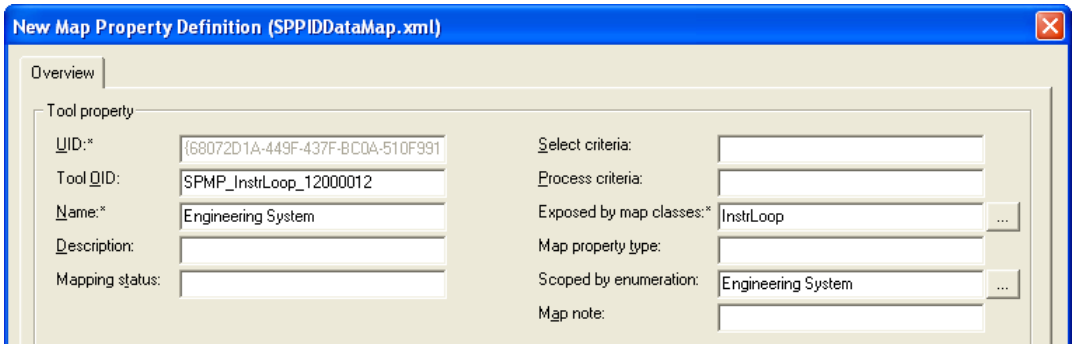
   - By default, the Schema Editor defines the interface definition that exposes the selected property definition based on the tool map class definition you selected. However, the

default is only a suggestion based on context, and you can change the interface definitions that expose this property definition in the **Exposed by interface definitions** box.

- In the **Scoped by property type** box, be sure to select the new enumerated list you created previously as the property type for the new property.
- When you click **OK**, the new property appears in the **Unmapped SmartPlant properties** table in the **Edit Map Class Definition** dialog box.

Add the New Property to the Tool Map Schema and Metadata

1. Above **Unmapped application properties** on the left side of the dialog box, click **New Map Property Definition** on the toolbar.
2. In the **New Map Property** dialog box, type the name and any other information that you want to define for the new map property definition, and click **OK** to create it.



TIPS

- In the **Scoped by enumeration** box, be sure to select the new enumerated list you created previously as the enumerated list for the new property.
- By default, the authoring tool metadata adapter supplies the tool identifier (OID) and the name for the map property definition. You can change the default name, which is always a number, by selecting the existing number and typing the name you want.
- In the **Select criteria** box, you can define the criteria that the tool adapter uses to select or identify objects for this tool property definition. Some authoring tool metadata adapters define this information by default. See the mapping rules for your authoring tool for how to set the value for this property.
- In the **Process criteria** box, you can define the criteria that the adapter uses to process objects that are instances of this tool property definition. Some authoring tool metadata adapters define this information by default. See the mapping rules for your authoring tool for how to set the value for this property.

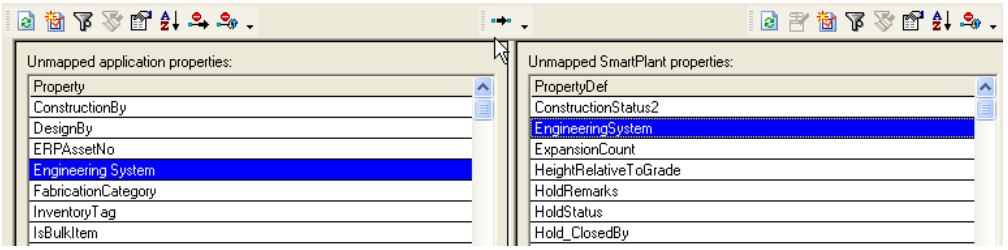
  For example, if the tool adapter must do a conversion on an enumerated list or the unit of measure for a property before the property is published, use the **Process criteria** box to specify the method used for the conversion, for example, EFAdapter.EnumConversions.
- By default, the software creates the relationship between the map class definition you selected to edit and the new map property definition. You can see which map class definition exposes the map property definition in the **Exposed by map classes** box.

- You can change the property category for the new map property definition in the **Category** box. Changing the property category displays other properties that you can define for the new map property definition.
- When you click **OK**, the new map property definition is added to the **Unmapped application properties** table in the **Edit Map Class Definition** dialog box.

IMPORTANT For some authoring tools, such as SmartPlant Instrumentation, changes to the metadata in the tool database are made automatically when you add the object to the tool map schema. For other authoring tools, changes are only saved when you click **File > Save > Tool Schema File** command in the Schema Editor to save the tool map schema. This allows you to cancel your changes when you are adding new tool properties, enumerated lists, or enumerated entries before you save the tool map schema. For more information, see *Mapping with Metadata Adapters* (on page 291).

Map Tool Property to the SmartPlant Schema Property

1. In the **Unmapped application properties** table, select the tool map property definition that you want to map.
2. In the **Unmapped SmartPlant properties** table, select the SmartPlant property definition that you want to map.
3. On the toolbar, click **Map** .



TIP The mapping you defined appears in the mapped objects table at the bottom of the dialog box.

4. If you want to define both publish and retrieve mapping for the enumerated list entry, click the **Retrieve** tab, and repeat steps 1-3 in this procedure.
5. Click **OK** in the **Edit Map Class Definition** dialog box.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

Save the Tool Map Schema and Schema Extensions File

1. Click **File > Save > Tool Schema File**. For some tools, saving the tool map schema also saves changes to the tool metadata database.
2. To save the changes to your schema extensions file, click **File > Save > Schema File**.

NOTES

- When you close the Schema Editor, the software prompts you to save your connection information, user interface options, and loaded tool map schemas to a session file. Click

Yes in the message box to save the session file so that you can use it to return to this working environment. However, you should not open a session file after launching the CMF file from the Desktop Client.

- If you save changes to the CMF file or a tool map schema but did not validate the changes before exiting, the software displays an information message to remind you that the file contains changes that have not been validated. Click **OK** to dismiss the message.
- If you make changes to the SmartPlant schema, you must load the changes into the SmartPlant Foundation database.

Propagating the New Property and Enumerated List to Other Authoring Tools Using Compare

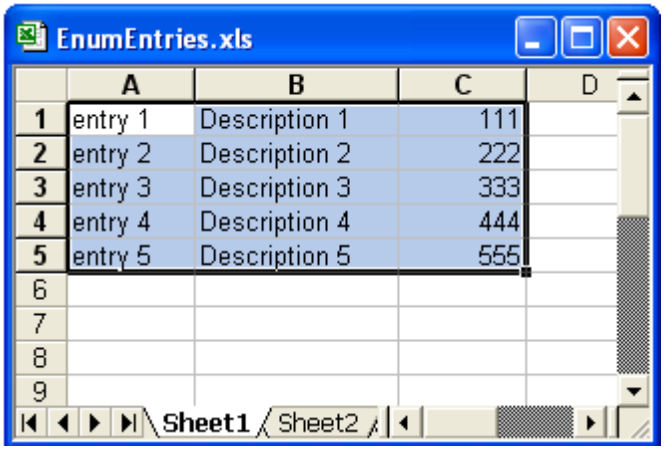
- After you add new objects to the SmartPlant schema, do a compare with other authoring tool map schemas to identify changes that need to be made in the other authoring tools and then add new objects and define mapping in the **Map Environment**. For more information, see *Compare a Tool Map Schema to the SmartPlant Schema* (on page 385).

Import Enumerated List Entries from a Spreadsheet

1. In Microsoft Excel, create a spreadsheet containing your enumerated list entries.

   TIP In an authoring tool map schema, you can also import enumerated list entries into a tool map enumerated list and units of measure into a tool map unit of measure list using this procedure.

2. In Excel, select the rows containing your enumerated list entries, and click **Edit > Copy**.

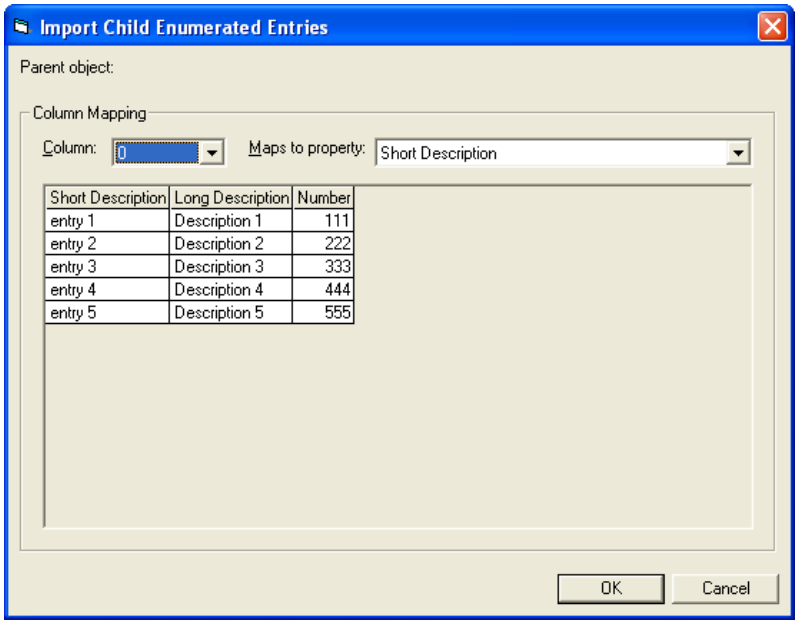


3. In the Schema Editor, create the enumerated list type to which you want to add the list entries. For more information, see *Create an Enumerated List* (on page 173) or *Create a Map Enumerated List Definition* (on page 437).
4. In the **New Enumerated List** dialog box, click **Import Entries from Clipboard**.

   TIP To import entries into a tool map enumerated list or map UoM list, click **Import Clipboard Entries** instead.

5. Verify that the default mapping between the columns in your copied spreadsheet and properties of enumerated list entries is correct. If this mapping is correct, you can skip to step 7 to import the entries.

6. If the default mapping is not correct, click the number for the column that you want to change the mapping for in the **Column** list. Numbering for the columns starts with zero, which is the first column in your spreadsheet.



7. In the **Maps to property** list, select the property that corresponds to the selected column in your copied spreadsheet.
8. When the mapping is correct, click **OK** to import the enumerated list entries.
9. Click **OK** in the **New Enumerated List** dialog box to save the new entries.

Compare a Tool Map Schema to the SmartPlant Schema

After you add new objects to the SmartPlant schema, you can compare the SmartPlant schema to authoring tool map schemas to determine whether equivalent objects in the tool map schema have been mapped. Comparing tool map schemas to the SmartPlant schema is especially useful when you add new objects in the process of defining mapping one tool and need to identify and copy those changes out to other authoring tools.

Load the Tool Map Schema and Compare

1. In the Desktop Client, set your scope for the applicable plant, and then find the CMF file.
2. Right-click the CMF file, and then click **Edit > Check Out**.
3. Click **OK** in the **Check Out** dialog box.
4. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
5. On the **Set Active Configurations** dialog box, select the schema version that you want to view.

6. Click **OK** to set the configuration.

   TIPS

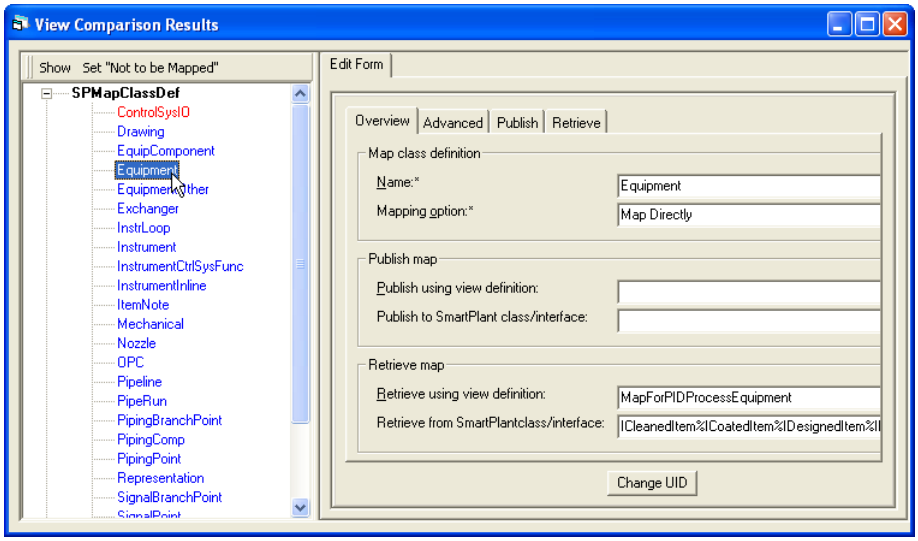
   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

7. In the Schema Editor, synchronize the authoring tool map schema for which you want to define mapping. For more information, see *Synchronize Tool Metadata with the Tool Map Schema* (on page 328).

8. Click **File > SmartPlant > Compare (Active Tool Schema) With Schema**.

   Results of the comparison appear in a tree view in the **View Comparison Results** window.



   TIPS

   - The following colors have special meaning in the **View Comparison Results** window.
     - Green - Fully mapped
     - Red - Not mapped
     - Blue - Incomplete mapping
   - The Schema Editor does not take retrieve mapping into account during comparison. After you resolve all the outstanding issues for publish mapping for an object, the object name changes to green in the tree view even if the retrieve mapping is not defined.

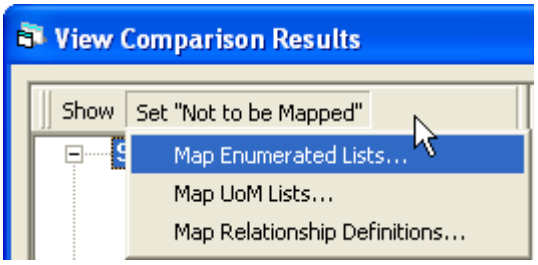
9. To review differences, expand the nodes in the tree view.

10. Click an object to review its publish and retrieve mapping on the **Publish** and **Retrieve** tabs in the **View Comparison Results** window.

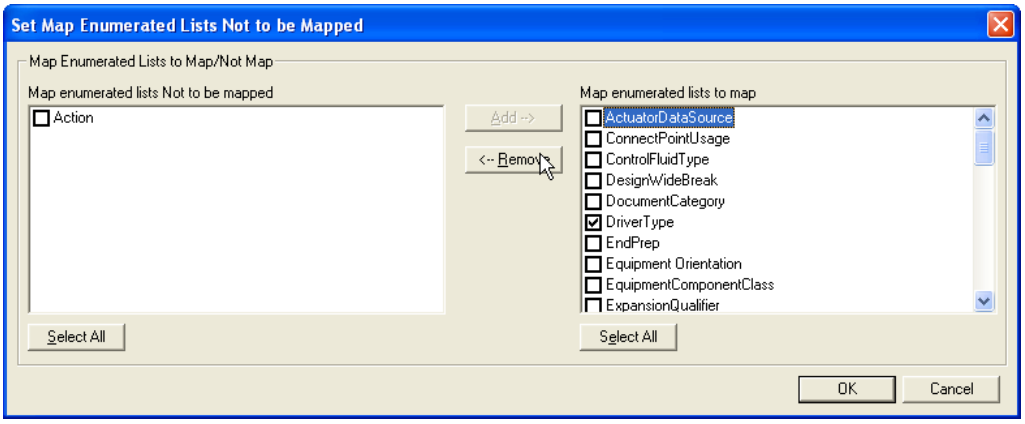
Mark Objects that You Do Not Want to Map

You can resolve differences between your tool map schema and the SmartPlant schema in a variety of ways. For example, you can mark properties or class definitions that you do not want to be mapped. These objects to not appear in the **Map Environment** or in the **View Comparison Results** tree.

1. Click **Set "Not to be Mapped"** and then click the type of object that you want to exclude from mapping.



2. In the **Map <object type> to map** list, select the objects that you want to exclude from mapping, and then click **Remove**.

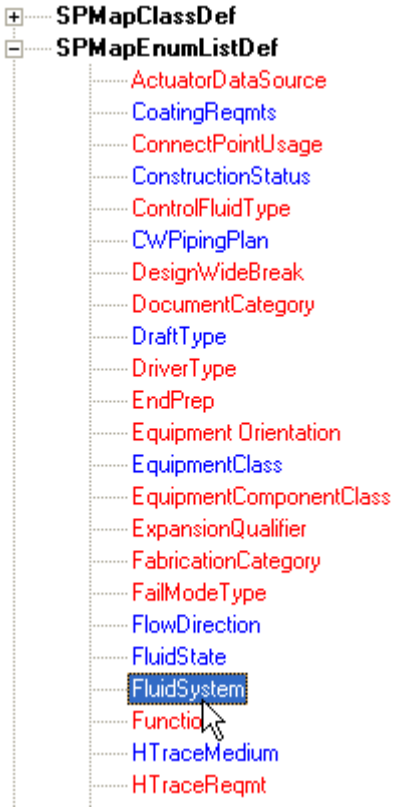


3. Click **OK** to save the changes.

NOTE If you decide to map the objects later, you can make them available for mapping by selecting the objects in the **Map <objects> Not to be Mapped** list and then clicking **Add**.

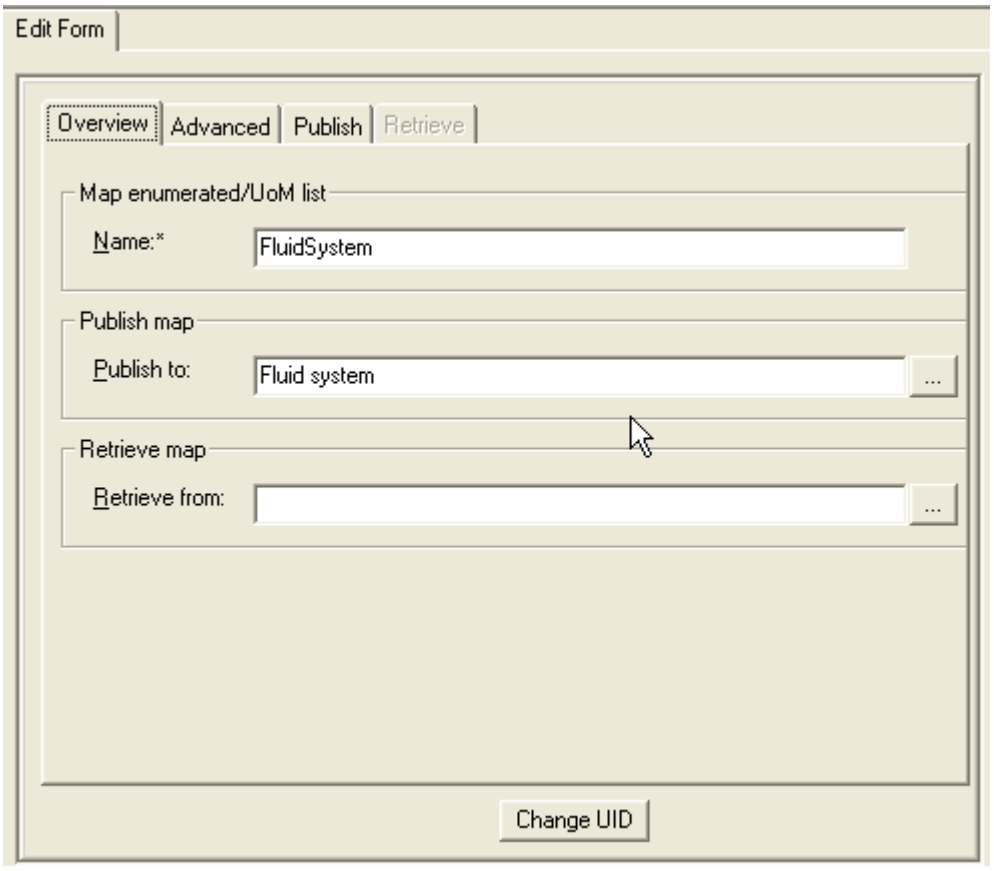
Map Tool Objects to SmartPlant Schema Objects

1. In the tree view, select the object that is unmapped or partially mapped.



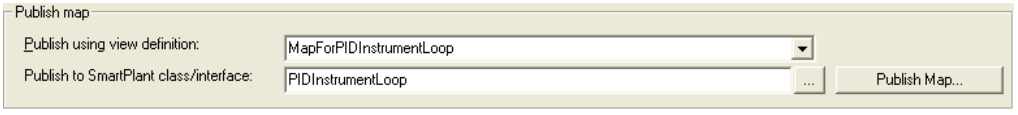
Information for the selected object appears on the **Edit Form** tab.

2. On the **Overview** tab, make sure that the publish to and retrieve from objects are defined properly.



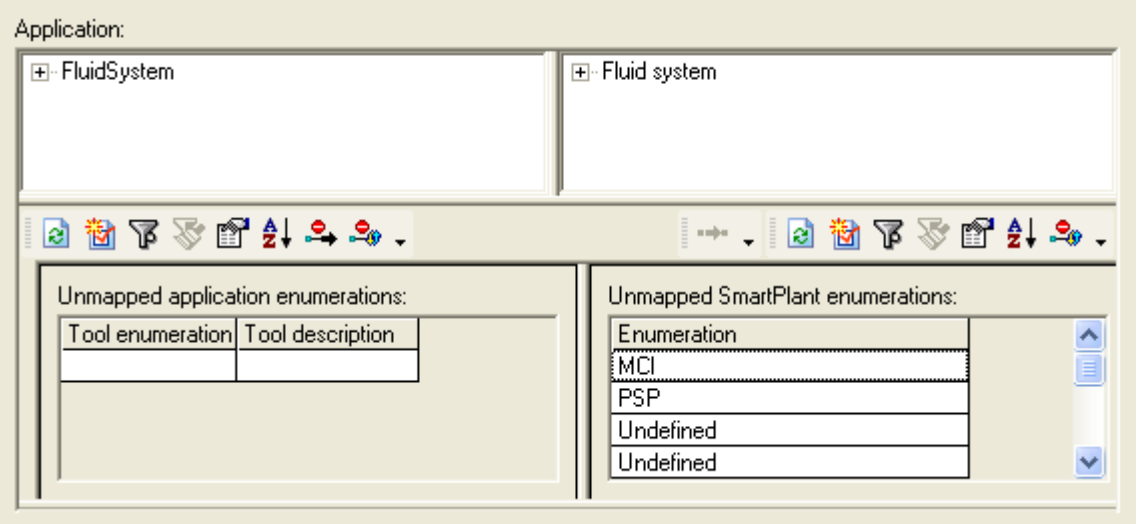
TIPS

- Map class definitions can be mapped using view definitions. View definitions allow you to see properties categorically. If a view definition does not exist for the interface definition that you want to map to, the Schema Editor generates a view definition automatically for mapping. The software does not save this view definition. In future releases, additional mapping view definitions will be delivered to make mapping easier.



- If a map class definition or an enumerated list definition have objects defined for both publish and retrieve, then the inconsistency could be in the map property definitions for the map class definition or the enumerated entries for the map enumerated list definition. To continue resolving inconsistencies, see step 3.

3. On the **Publish** and **Retrieve** tabs, review publish and retrieve mapping for map property definitions if you selected a map class definition or map enumerated entries if you selected a map enumerated list.



4. If map property definitions or map enumerated list entries are missing on the authoring tool side, click **New** on the toolbar and create the object.
5. If objects that need to be mapped are not mapped, select the objects that you want to map and click Map .

Save the Tool Map Schema

- Click **File > Save > Tool Schema File** to save the tool map schema. For some tools, saving the tool map schema also saves changes to the tool metadata database.

Map Publish and Retrieve Properties in the Map Environment

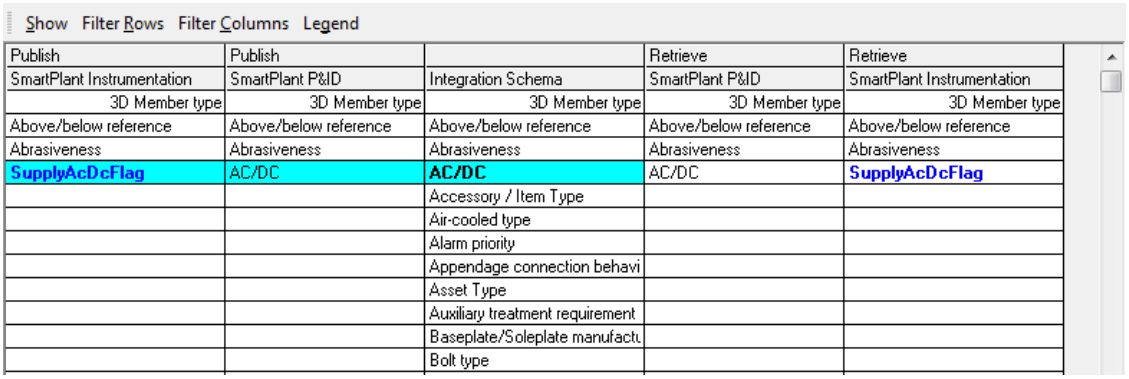
The **Integrated Map Environment** in the Schema Editor lets you map, view, and manage objects that are mapped into (publish) and out of (retrieve) the SmartPlant schema from tool map schemas. The Integrated Map Environment user interface visually represents tool map schemas and the SmartPlant schema in a table view to efficiently see the various schema definitions of different applications at one time. Color-coding provides a quick look into which schemas are completely mapped and which schemas require more mapping. The two-panel view consists of a left panel that displays all open tool map schemas in a table view and a right panel for detailed schema editing.

Mapping Table User Interface

The left panel of the Integrated Map Environment contains the **Mapping Table**, which is used to display objects and their current mapping status for all open tool map schemas. From this panel, you can modify entire enumerated lists. You can use the panel on the right to modify the individual enumerations within the enumerated lists. For example, using the Mapping Table, you can create an integration enumerated list from a map enumerated list, or create a map enumerated list from an integration enumerated list.

The auto-correlate command in the **Mapping Table** automatically correlates unmapped map enumerated lists with an existing integration enumerated list. You can also mark or unmark map enumerated lists as **"Not to be mapped"** individually or in bulk.

The **Show** command above the Mapping Table switches between **Map Enumerated Lists** and **Map UoM Lists** in the Mapping Table panel.



You can filter schema objects by tool schema and mapping status with the **Filter Rows** (mapping status) and **Filter Columns** (tool schema) commands. For more information, see *Filter the Mapping Objects in the Map Environment* (on page 394).

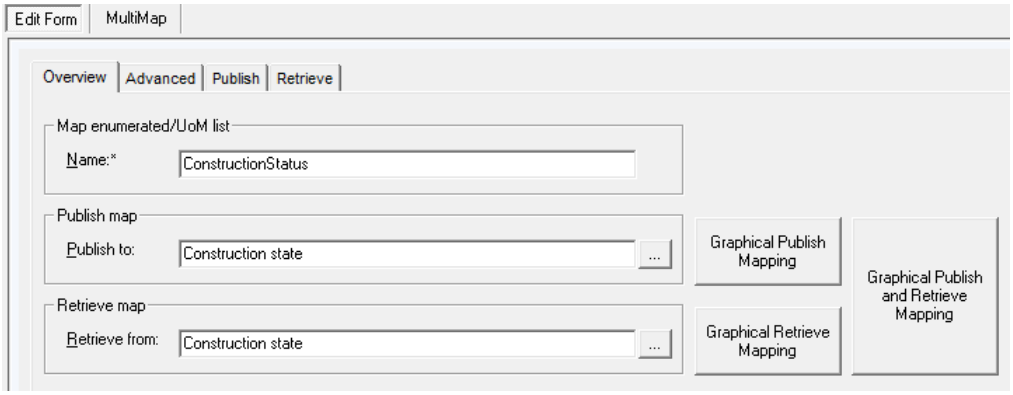
The different colors used in the Mapping Table indicate whether or not a value is mapped completely. For a full explanation of the color schemes, see *Map Environment Color Legend* (on page 395), or click **Legend** above the Mapping Table.

Edit Form and MultiMap User Interface

The right panel contains the **Edit Form** user interface and the **MultiMap** user interface. These interfaces are used for detailed graphical mapping and viewing. The commands in the right

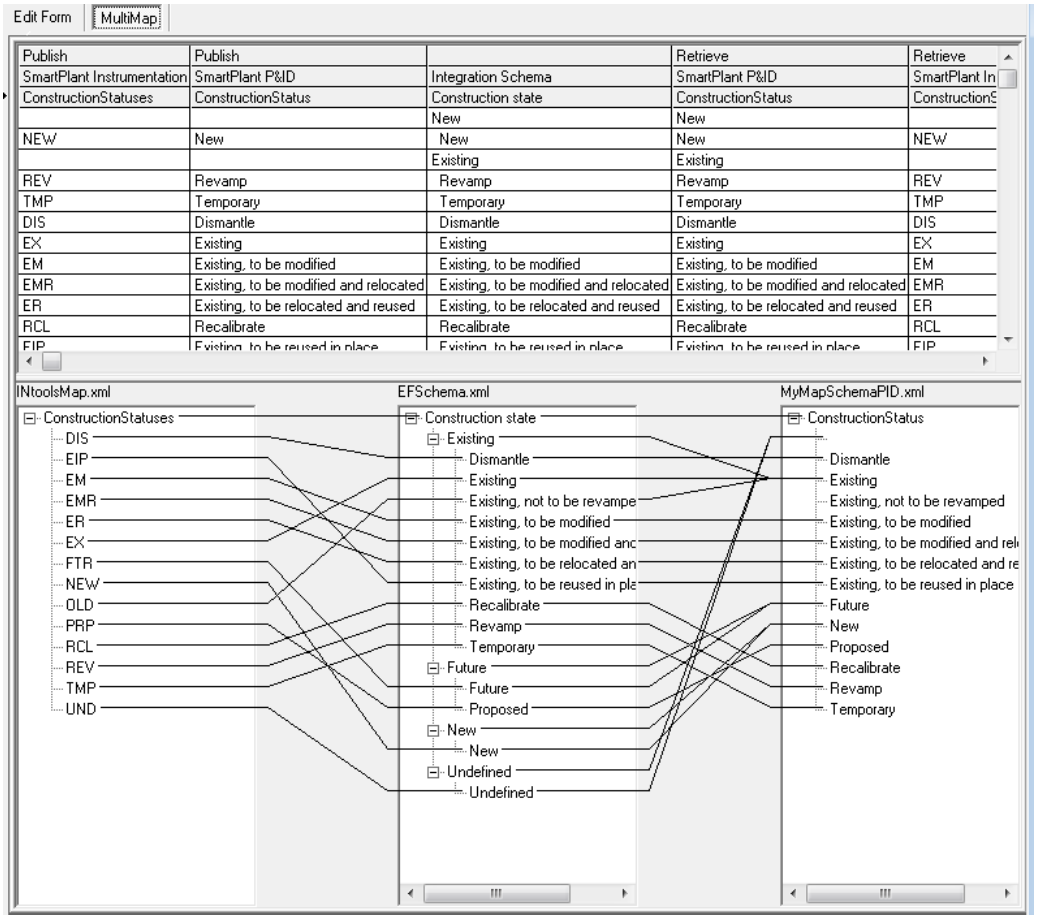
panel are similar to the commands in the Mapping Table, but work on an individual enumeration level instead of entire enumerated lists.

The **Edit Form** tab displays the same options available in the **Edit** command for the object that was last selected in the Mapping Table. The functionality available in the **Edit Form** depends on the type of object selected.

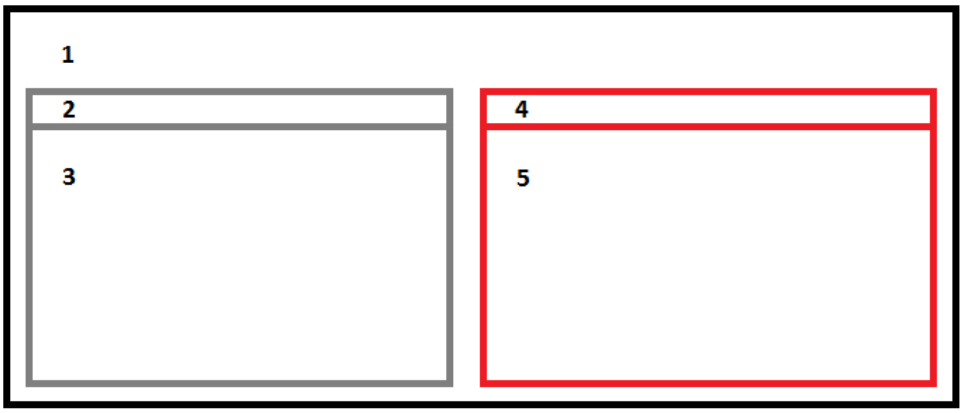


For more information on the **Edit Form** user interface, see *Map Publish and Retrieve Properties in a Graphical View* (on page 414).

The **MultiMap** tab contains a table similar to the Mapping Table, but displays the individual enumerations within the list selected in the Mapping Table. The area below the enumerations list contains a graphical display of the mapping with two or more columns to represent the publish, retrieve, and integration schema objects. These objects are connected by **Relationship Lines**, which visually represent the mapping relationships between the objects. You can use shortcut menu commands and drag-and-drop gestures to modify these relationships.



Integrated Map Environment User Interface



	Description
1	Schema Editor.
2	Mapping Table filter tools and legend.
3	Mapping Table.
4	**Edit Form** and **MultiMap** tabs.
5	Detailed Mapping through the **Edit Form** or **MultiMap** user interface. You change between these environments by clicking the **Edit Form** and **MultiMap** tabs above.

See Also

Create Mapping for Enumerating Lists in Map Environment (on page 395)
Create Mapping for Enumerated Lists in Graphical Mapping (on page 397)
Create Mapping for Enumerated Lists with MultiMap Editor (on page 399)
Create SmartPlant Schema Enumerated Lists from Mapping Table (on page 401)
Auto-correlate SmartPlant Schema Enumerated Lists in Mapping Table (on page 402)
Mark Individual Map Enumerated Lists as Not to be mapped (on page 405)
Mark Multiple Map Enumerated Lists as Not to be mapped (on page 406)

Filter the Mapping Objects in the Map Environment

You can use the **Show**, **Filter Rows**, and **Filter Columns** commands in the Integrated Map Environment to filter mapping objects by tool schema and mapping status.

1. Click **Show**, and select a mapping object type to view. For example, click **Show > Map Enumerated Lists**.
2. Click **Filter Rows** to show or hide mapping objects based on their mapping status. For example, unselect the **Mapped** and **<Not to be mapped>** options to only display mapping objects that are unmapped in both the integration and tool schemas.
3. Click **Filter Columns** to show or hide the publish or retrieve tool columns. For example, unselect the **Publish/SmartPlant P&ID** and **Retrieve/SmartPlant P&ID** options to only display the other tool mapping information.
4. Click **Legend** to display a legend that explains the colors used in the left panel.

Map Environment Color Legend

The left panel of the Integrated Map Environment displays the mapping of all open tool and integration schema objects. The color of these objects indicates different levels of mapping progress. For example, if an integration object is completely mapped between the publish and retrieve profiles of all open tool schemas, the objects are black.

- **Normal black text** - Mapping is complete.
- **Bold black text** - Mapping is incomplete.
- **Blue** - Some child tool schema objects are unmapped.
- **Magenta**- Some child SmartPlant schema integration objects are unmapped.
- **Red** - No mapping is completed.

Create Mapping for Enumerating Lists in Map Environment

The following topic provides an example of defining mapping for an enumerated list to multiple tool schemas from the Schema Editor.

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

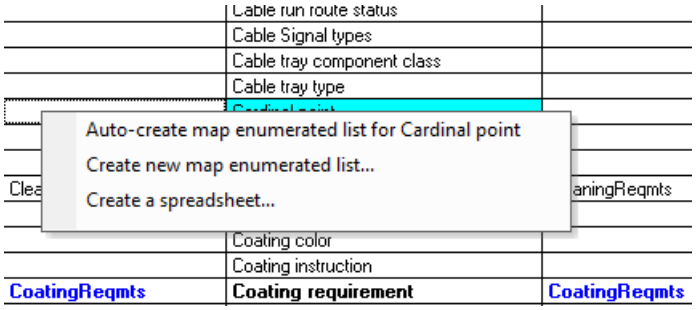
   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Tool Schema > Integrated Map Environment**.

   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

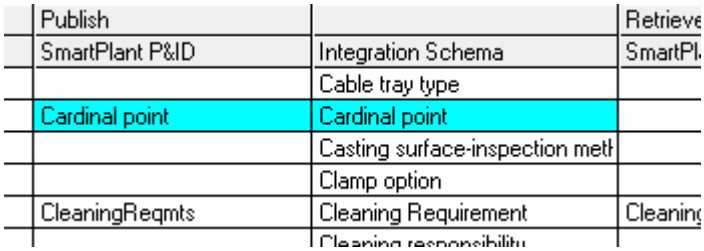
3. Right-click an empty **Publish** cell in the left panel and select **Auto-create map enumerated list for <object>**, where object is the schema object you selected. For example, right-click the empty cell in the **Cardinal point** row of the SmartPlant P&ID **Publish** column and select **Auto-create map enumerated list for Cardinal point**.



NOTE You can select **Create new map enumerated list** to manually create a new enumerated list with the **New Map Enumerated List Definition** dialog box. For more information, start with step 6 in *Create an Enumerated List* (on page 173).

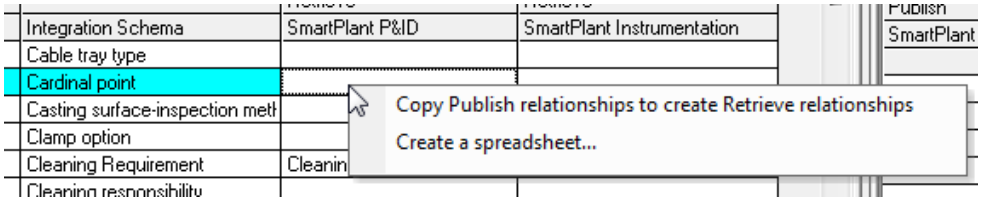
TIP If you select a multi-level enumerated list, a dialog box asks if you want an equivalent multi-level list created within the tool schema (map file) or if you want this multi-level list flattened to a single-level list in the tool schema. If you select a multi-level list, an equivalent hierarchy of enumerations (as existed in the original enumerated list in the integration schema) is created in the tool schema. If you select a flattened list, a single-level list is created in the tool schema.

The software creates a new map enumerated list for the integration enumerated list, a new map enumeration for each enumeration in the integration enumerated list, and maps all of the created map objects to the corresponding integration object.



4. Repeat step 3 on the corresponding tool schema **Retrieve** cell. For example, right-click the empty cell in the **Cardinal point** row of the SmartPlant P&ID **Retrieve** column and select **Copy Publish relationships to create Retrieve relationships**.

This creates the retrieve relationships corresponding to the publish relationships created in step 3.



5. Click the cell that you mapped in the previous step. For example, click either of the **Cardinal point** cells you created.

The mapping details for Cardinal point appear in the right panel.

TIP In the right panel, you can use the **Edit Form** or **MultiMap** tabs to create additional mapping.

6. Right-click any cell in the left panel, and click **Create a spreadsheet** to create a spreadsheet of the displayed tool and integration schema information.

Create Mapping for Enumerated Lists in Graphical Mapping

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

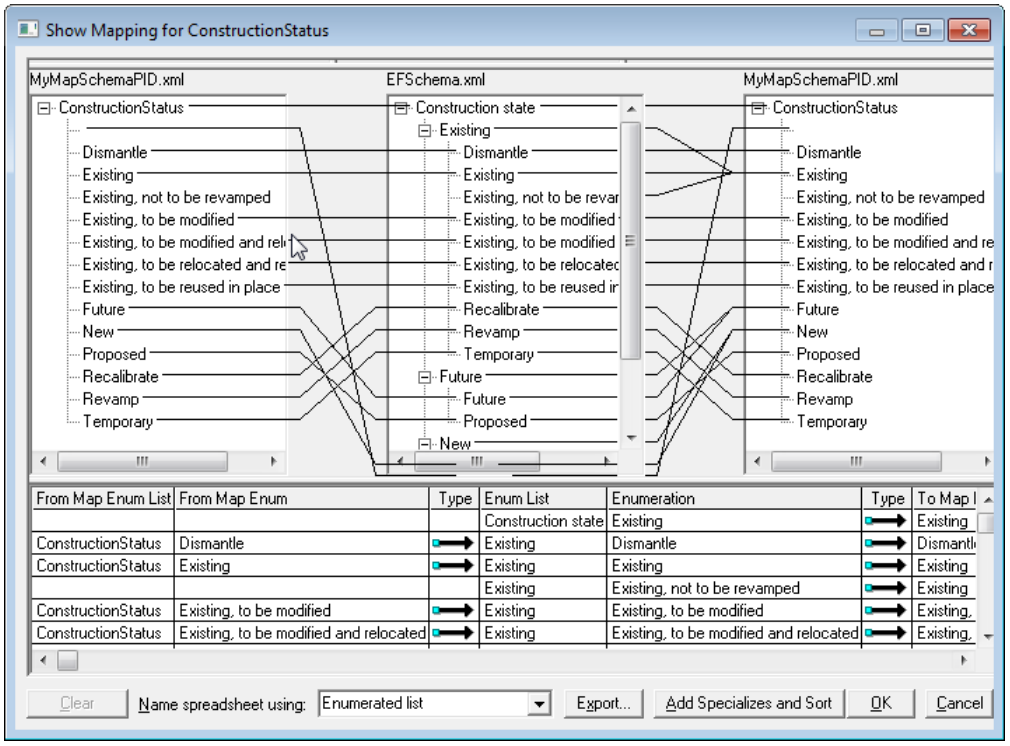
2. Click **Tools > Tool Schema > Integrated Map Environment**.

   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

3. Click a non-empty **Publish** cell in the left panel to display the **Edit Form** in the right panel. In this example, click the non-empty **Publish** cell on the same row as the **Construction state Integration schema** cell.

4. On the **Overview** tab, click the mapping option you want. For example, click **Graphical Publish and Retrieve Mapping** to map both publish and retrieve enumerations.

NOTE You can select **Graphical Publish Mapping** to only map enumerations for publish. You can select **Graphical Retrieve Mapping** to only map enumerations for retrieve.



5. Select an enumeration in the publish tree view (left) or the retrieve tree view (right) to create relationship for publish or retrieve. This example shows the creation of a publish relationship.
6. Drag your cursor toward the mapped-to SmartPlant tree (middle) and a **Relationship Line** appears.
7. Click the property in the mapped-to SmartPlant tree for which you want to create a publish relationship.

   NOTES

   - You can create a publish or retrieve relationship by selecting a mapped-to SmartPlant property as your starting point and dragging the relationship line to an enumeration in either the publish or retrieve tree.
   - You can automatically map enumerations for publish by right-clicking an enumeration and selecting one of the following shortcut menus:
     - **Auto-map enumeration (publish)** - Provides a list of the best matches for mapping between the selected map enumeration and existing objects in the SmartPlant Schema. Selecting an enumeration from this dialog box and clicking **OK** results in a relationship being created.
     - **Auto-create enumerations from selected map enumerations** - Creates a new enumeration and then creates a relationship between the selected map enumeration in the tree view and the new enumeration.
8. Click **OK**.

Create Mapping for Enumerated Lists with MultiMap Editor

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Tool Schema > Integrated Map Environment**.

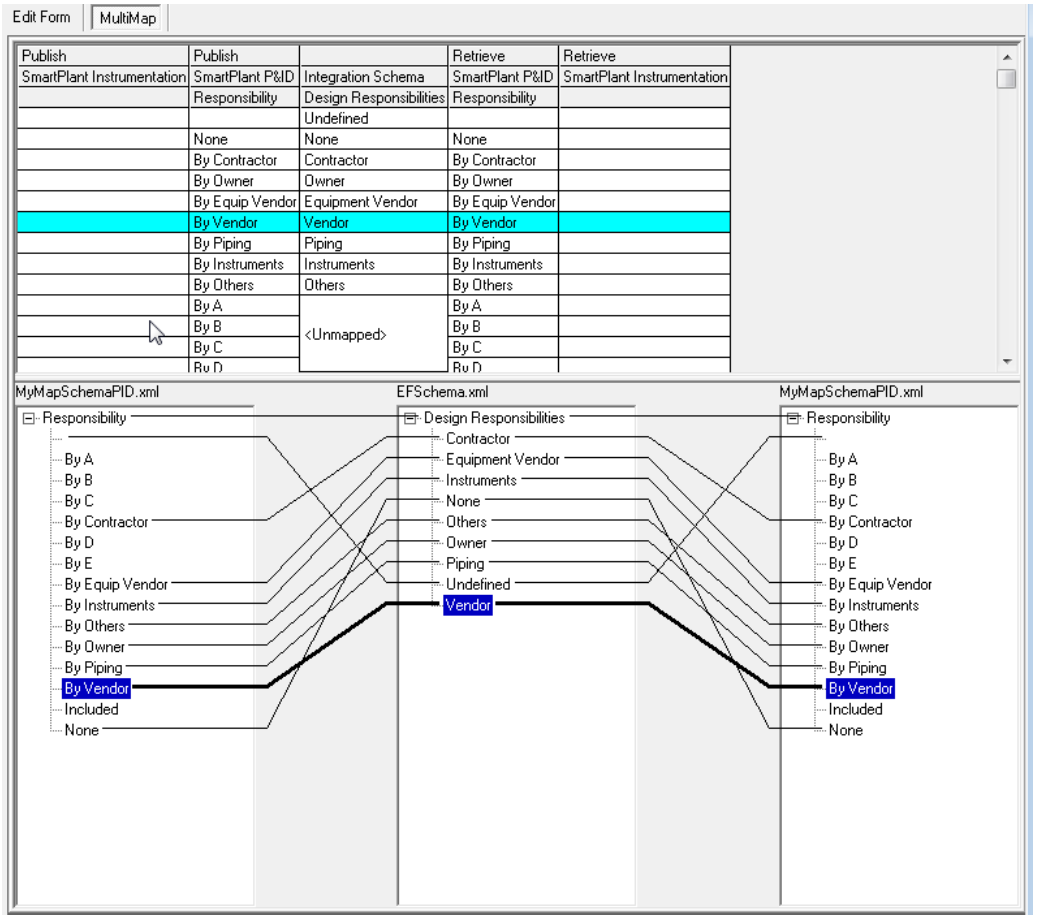
   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

3. Click a non-empty **Publish** cell in the left panel, and click **MultiMap** in the right panel. In this example, click the **Publish** cell on the same row as the **Design Responsibilities** cell of the **Integration Schema** column.

   The detailed mapping for the **Design Responsibilities** enumerated list appear in the **MultiMap** table in the right panel.

4. In the right panel under **MultiMap**, select an enumeration in a publish column (left) or the retrieve column (right).

The graphical mapping for the selected map enumerated list appears below the publish and retrieve columns.



5. Select a map enumeration object from either the publish or retrieve columns and drag-and-drop it on the appropriate object in the middle SmartPlant schema column. This creates a map relationship between the two objects, which is indicated with a **Relationship Line** between the objects.

   NOTE You can create a matching relationship by dragging and dropping the Integration Schema value to the tool integration value, or from the tool integration value to the Integration Schema value.

Create SmartPlant Schema Enumerated Lists from Mapping Table

The following topic provides an example of creating an enumerated list for the SmartPlant schema from a map enumerated list within a tool schema.

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

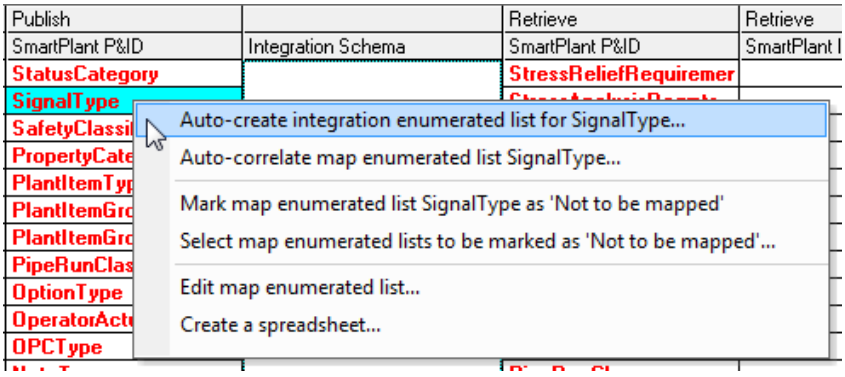
2. Click **Tools > Tool Schema > Integrated Map Environment**.

   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

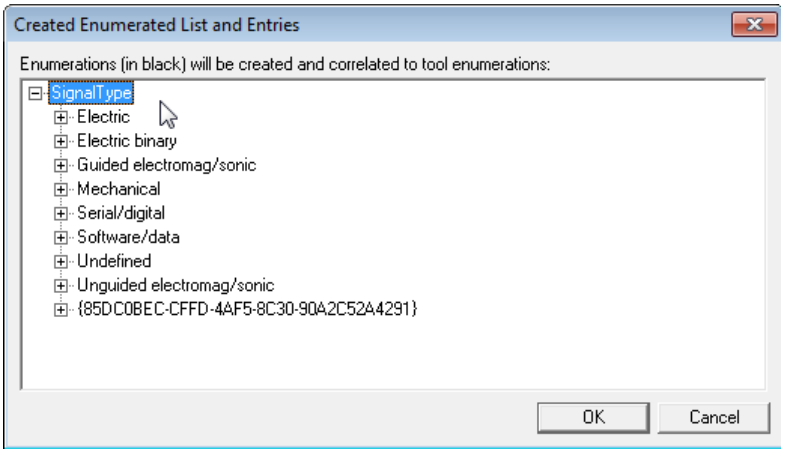
3. Right-click a **Publish** cell in the left panel that is displayed in red text. For example, right-click **SignalType** in the **SmartPlant P&ID Publish** column.

   TIP Tool schema objects not mapped to the SmartPlant schema are listed at the bottom of the table in red text. The **Integration Schema** column for these items displays **<Unmapped>** to indicate no mapping exists between the tool and SmartPlant schemas.

4. Select **Auto-create integration enumerated list for <enumerated list name>**. For example, select **Auto-create integration enumeration list for SignalType**.



5. Click **Yes** to create publish and retrieve relationships, or click **No** to only create publish relationships. In this example, click **Yes**.
6. Review the enumerations that will be created and correlated to tool enumerations in the **Created Enumerated List and Entries** dialog box, and click **OK**.



SignalType is removed from the section of unmapped enumerations and moved to the mapped section above. It appears in alphabetical order in the mapped objects list.

TIPS

- You can locate the enumeration you created above in the mapped objects section in alphabetical order.
- You can auto-create any other enumerations for other tools by using the **Auto-create map enumerated list for <enumerated list name>** (for Publish columns) and **Copy Publish relationships to create Retrieve relationships** (for Retrieve columns that have corresponding entries in the Publish column) right-click menus.

Auto-correlate SmartPlant Schema Enumerated Lists in Mapping Table

The following topic provides an example of auto-correlating the SmartPlant schema enumerated list objects from the tool schema.

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.

- The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

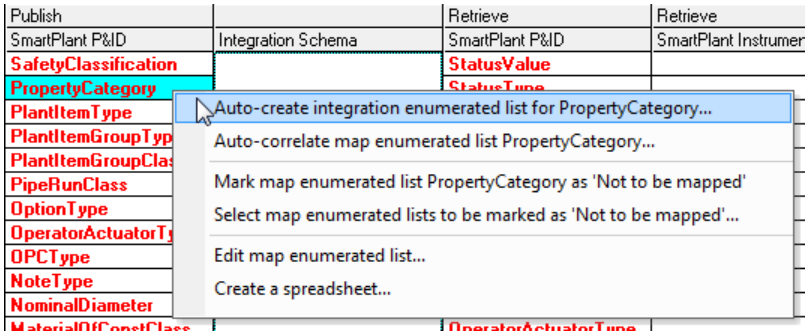
2. Click **Tools > Tool Schema > Integrated Map Environment**.

   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

3. Right-click a **Publish** cell in the left panel that is displayed in red text. For example, right-click **PropertyCategory** in the **SmartPlant P&ID Publish** column.

   TIP Tool schema objects not mapped to the SmartPlant schema are listed at the bottom of the table in red text. The **Integration Schema** column for these items displays **<Unmapped>** to indicate no mapping exists between the tool and SmartPlant schemas.

4. Select **Auto-correlate map enumerated list <enumerated list name>**. For example, select **Auto-correlate map enumerated list in PropertyCategory**.

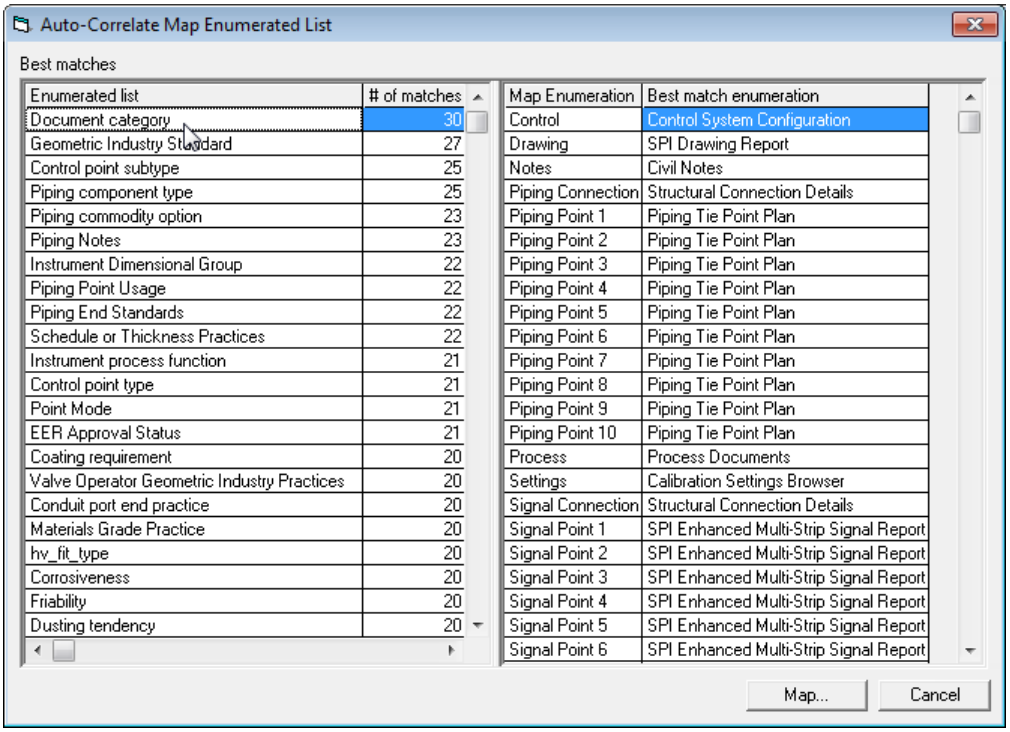


5. Select an enumerated list from the **Best matches** list that contains the enumerations you want to map.

   TIPS

   - The **Auto-Correlate Map Enumerated List** dialog box contains a list of **Best matches**. The **Enumerated list** column displays the enumerated lists sorted by the number of matches the software found. The **# of matches** column displays the number of matches each enumerated list contains.

- You can select different enumerated lists to view the potential mapping on the right side of the dialog box before you proceed with any mapping.

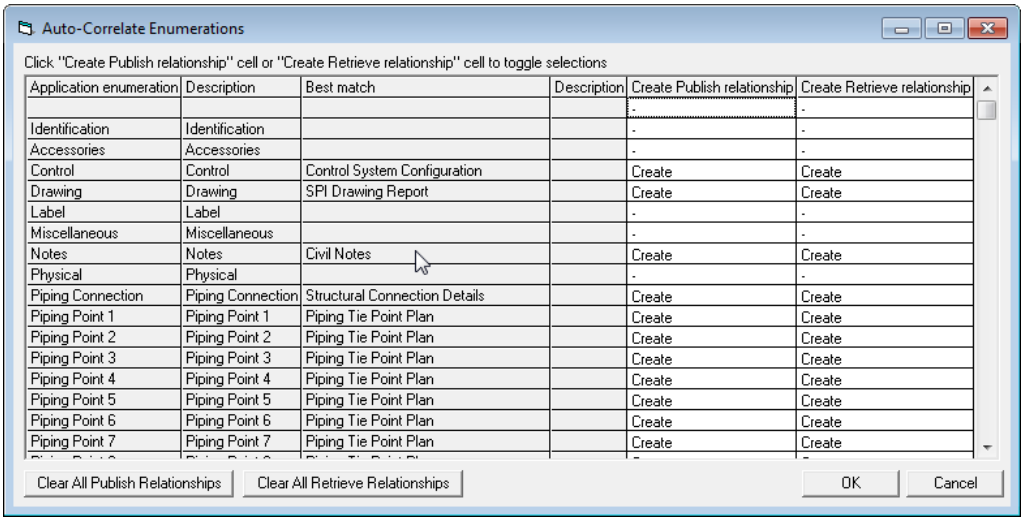


6. Click **Map** after you have selected an enumerated list that contains a match for the new enumeration. For example, select **Construction requirement** and click **Map**.
7. Click **Yes** to create publish and retrieve relationships, or click **No** to only create publish relationships. In this example, click **Yes**.
8. Review the information in the **Auto-Correlate Enumerations** dialog box and make changes to the publish and retrieve relationships if necessary.

TIPS

- Click any of the **Create Publish relationship** or **Create Retrieve relationship** cells to change the status of the relationship. These values can be **Create** or **Don't create**.
- You can change all of the values for the publish or retrieve relationships by clicking **Clear All Publish Relationships** or **Clear All Retrieve Relationships**.

- You cannot change cells that contain a dash. These enumerations have no matches, and will not be mapped in this workflow.



Application enumeration	Description	Best match	Description	Create Publish relationship	Create Retrieve relationship
				-	-
Identification	Identification			-	-
Accessories	Accessories			-	-
Control	Control	Control System Configuration		Create	Create
Drawing	Drawing	SPI Drawing Report		Create	Create
Label	Label			-	-
Miscellaneous	Miscellaneous			-	-
Notes	Notes	Civil Notes		Create	Create
Physical	Physical			-	-
Piping Connection	Piping Connection	Structural Connection Details		Create	Create
Piping Point 1	Piping Point 1	Piping Tie Point Plan		Create	Create
Piping Point 2	Piping Point 2	Piping Tie Point Plan		Create	Create
Piping Point 3	Piping Point 3	Piping Tie Point Plan		Create	Create
Piping Point 4	Piping Point 4	Piping Tie Point Plan		Create	Create
Piping Point 5	Piping Point 5	Piping Tie Point Plan		Create	Create
Piping Point 6	Piping Point 6	Piping Tie Point Plan		Create	Create
Piping Point 7	Piping Point 7	Piping Tie Point Plan		Create	Create

9. Click **OK** to create the relationships.

   StatusCategory is removed from the section of unmapped enumerations and moved to the mapped section above. It appears in alphabetical order in the mapped objects list.

   TIPS

   - You can locate the enumeration you created above in the mapped objects section in alphabetical order.
   - You can auto-create any other enumerations for other tools by using the **Auto-create map enumerated list for <enumerated list name>** (for Publish columns) and **Copy Publish relationships to create Retrieve relationships** (for Retrieve columns that have corresponding entries in the Publish column) right-click menus.

Mark Individual Map Enumerated Lists as Not to be mapped

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Tool Schema > Integrated Map Environment**.

TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

3. Right-click a **Publish** cell in the left panel that is displayed in red text. For example, right-click **LocationType** in the **SmartPlant P&ID Publish** column.

   TIP Tool schema objects not mapped to the SmartPlant schema are listed at the bottom of the table in red text. The **Integration Schema** column for these items displays **<Unmapped>** to indicate no mapping exists between the tool and SmartPlant schemas.

4. Select **Mark map enumerated list <enumerated list name> as 'Not to be mapped'**. For example, select **Mark map enumerated list LocationType as 'Not to be mapped'**.

   LocationType is removed from the section of unmapped enumerations and moved to the **Not to be mapped** section at the bottom of the list. It appears in alphabetical order under the unmapped objects list.

Mark Multiple Map Enumerated Lists as Not to be mapped

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Tool Schema > Integrated Map Environment**.

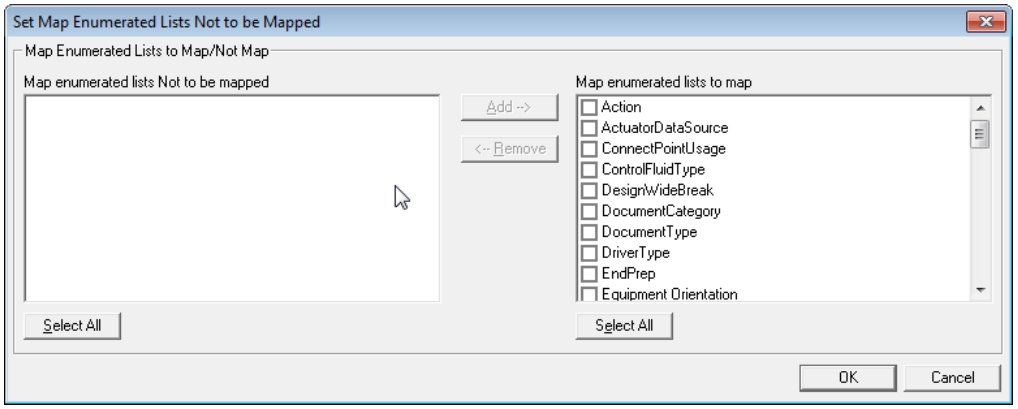
   TIP The Integrated Map Environment contains a left and right panel. The left panel contains the integration schema and all tool map schemas that you opened. The columns to the left of the **Integration Schema** column contain publish values, while the columns to the right of the **Integration Schema** column contain retrieve values for the various tool schemas. The right panel populates mapping details for any value you click on in the left panel. This environment allows you to see the integration and tool schemas as a whole and map them from a single environment.

3. Right-click a **Publish** cell in the left panel that is displayed in red text. For example, right-click **StatusCategory** in the **SmartPlant P&ID Publish** column.

TIP Tool schema objects not mapped to the SmartPlant schema are listed at the bottom of the table in red text. The **Integration Schema** column for these items displays **<Unmapped>** to indicate no mapping exists between the tool and SmartPlant schemas.

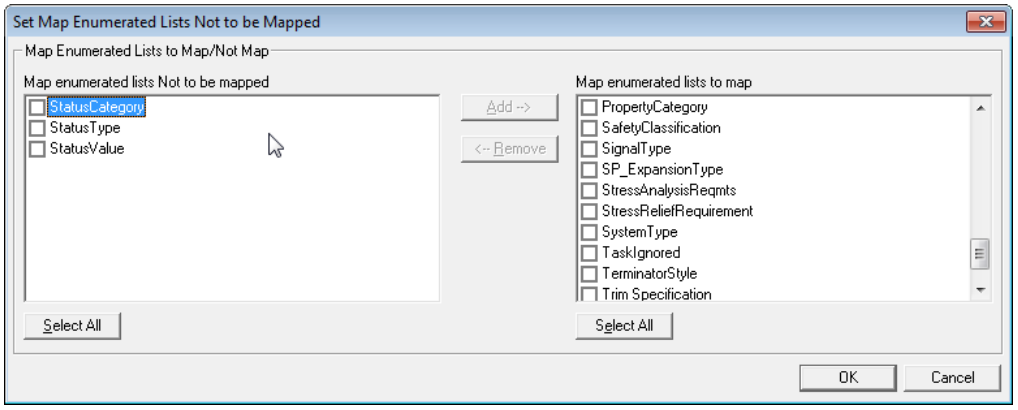
4. Click **Select map enumerated lists to be marked as 'Not to be mapped'**.

   The **Set Map Enumerated Lists Not to be Mapped** dialog box appears. The **Map enumerated lists Not to be mapped** section contains all map enumerated lists that are set to not be mapped, while the **Map enumerated lists to map** section contains all map enumerated lists from the unmapped section of the table, or all the schema objects that are red.



5. Select map enumerated lists from the **Map enumerated lists to map** section that you do not want to be mapped, and click **Remove** to move them to the Not to be mapped section.

   NOTE You can click **Select All** in either section to move all map enumerated lists from one side to the other.



6. Click **OK**.

   Any map enumerated lists that you set to **Not to be mapped** are removed from the unmapped section of the table and moved to the **Not to be mapped** section at the bottom of the table. They appear in alphabetical order in the Not to be **mapped** objects list.

Modify Reference Files with the Integrated Reference File Environment

The **Integrated Reference File Environment** in the Schema Editor lets you view and modify data in a set of schema and tool schema reference files. You can copy or replace data from a reference schema file into an active schema file, and copy or replace data from a reference tool schema file into an open tool schema file.

With this environment, you can copy data from schema and tool map files from a previous project into the schema and tool map files of a new project. This functionality allows you to deploy projects in less time by reusing configurations from other projects. For example, enumerated lists can be copied into a new project without requiring them to be manually re-entered.

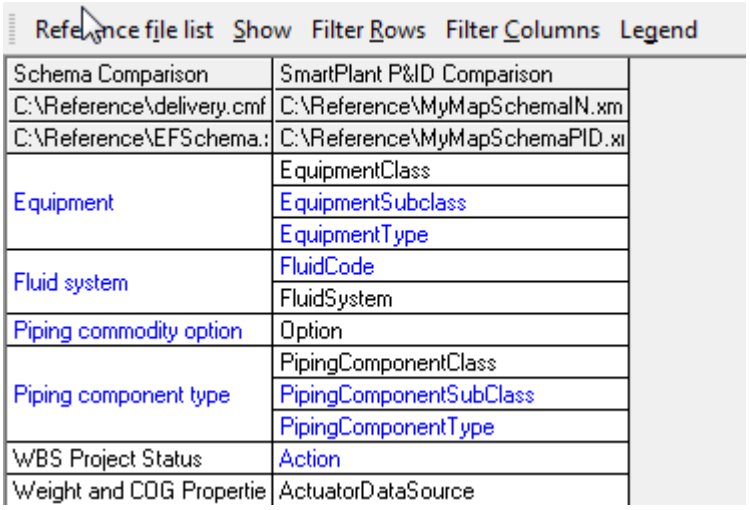
The software indicates whether an object only exists in the source file, only exists in the reference file, or if it exists in both. Objects that are included in both the source file and reference file are compared. If there is a difference between these files, you can update, create, or delete the source object based on the related reference object.

NOTE You need the related CMF file for any tool map files you want to reference. This file contains information essential to the copy functionality.

IMPORTANT The Integrated Reference File Environment does not support **Undo** and **Redo** actions.

Reference File Table User Interface

The Reference File Table User Interface, or left panel, displays the objects from the open schema files and tool map files. The **Schema Comparison** column shows the differences between the schema file and its reference schema files. The other columns display the comparison between the tool files that are open with the corresponding reference tool schema files.



Several commands are available above the table display.

The **Reference file list** command allows you to create, edit, view, or set the active reference file list. For more information, see *Configure the Reference File List* (on page 410).

The **Show** command allows you to display objects in the table by object type. You can choose to display class definitions, edge definitions, enumerated lists, units of measure lists, interface definitions, or relationship definitions in the Reference File Table.

The **Filter Rows** command allows you to show or hide objects based on their similarities or differences. You can also show or hide the file name rows at the top of the table with this command.

The **Filter Columns** command allows you to show or hide the tool comparison columns. The **Schema Comparison** column cannot be hidden, but any of the tool comparison columns can be hidden.

For more information on the **Show**, **Filter Rows**, and **Filter Columns** commands, see *Filter the Reference File Objects in the Integrated Reference File Environment* (on page 409).

The Reference File Table displays objects in different colors to indicate the level of difference between the two files. For more information, see *Integrated Reference File Environment Color Legend* (on page 410).

Shortcut Commands in the Reference File Table

Edit <target file object> - Opens the appropriate edit dialog box for the selected object.

View <reference file object> - Opens the appropriate view dialog box for the selected reference file object.

Select Replace <reference file object> - Opens the Selective Update dialog box, which includes the same functionality as the right panel of the Integrated Reference File Environment.

Create Excel spreadsheet - Creates an Excel spreadsheet of the data displayed in the Reference File Table.

Filter the Reference File Objects in the Integrated Reference File Environment

You can use the Show, Filter Rows, and Filter Columns commands in the Integrated Reference File Environment to filter objects by type, file differences, and file type.

1. Click **Show**, and select an object type to view. For example, click **Show > Enumerated lists**.
2. Click **Filter Rows** to show or hide objects based on their similarities or differences. You can also show or hide the file name rows located at the top of the Reference File Table. For example, unselect **In both schemas, no differences** to hide objects that exist in both schema files and are not different.
3. Click **Filter Columns** to show or hide any of the tool comparison columns. For example, if you opened more than one set of tool schema files to compare, you can hide the tools that you are not actively working on. To hide the SmartPlant P&ID comparison column, click **Filter Columns > SmartPlant P&ID**. To show the column, follow the same steps.
4. Click **Legend** to display a legend that explains the colors used in the Reference File Table.

Integrated Reference File Environment Color Legend

The Reference File Table displays a column for each set of mapping files open for comparison. It also displays the active schema files in the far left column. The color of the objects in this table indicates different levels of similarities. For example, if an object in the SmartPlant P&ID column is blue, it exists in both map files, but there is at least one difference between them.

Schema files

- **Red** - The object only exists in the target schema.
- **Green** - The object only exists in the reference schema.
- **Blue** - The object exists in both schemas, but is different.
- **Black** - The object exists in both schemas, and is identical.

Tool mapping files

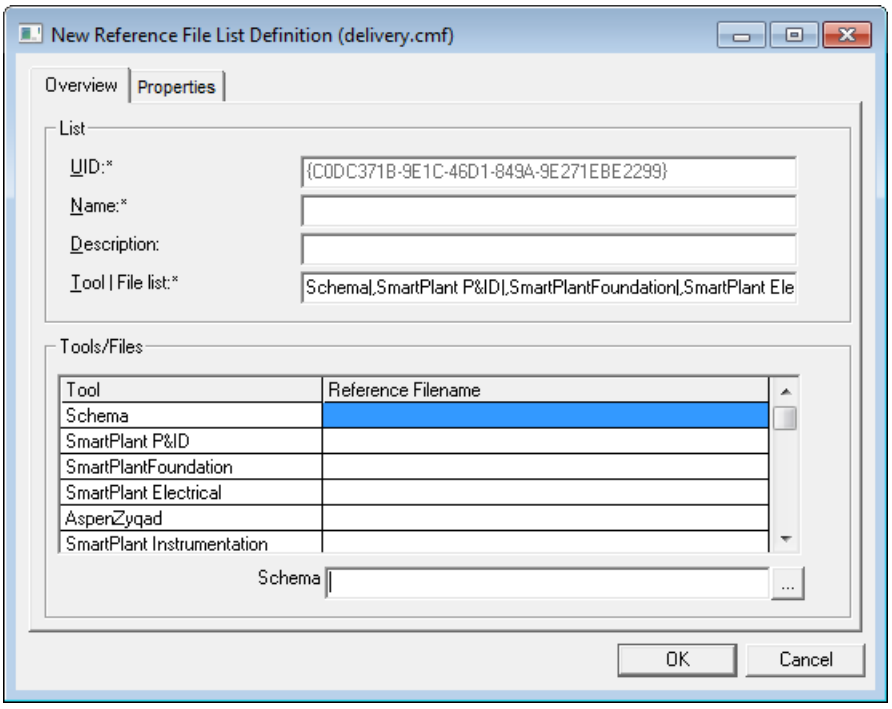
- **Red** - The object only exists in the target map file.
- **Green** - The object only exists in the reference map file.
- **Blue** - The object exists in both map files, but is different.
- **Black** - The object exists in both map files, and is identical.

Configure the Reference File List

Before you open the Integrated Reference File Environment, you need a reference file list. Follow the procedures below to create, edit, and set the active reference file list.

Create a new reference file list

The **Create** command is used to create a new reference file list object that specifies the files used as the reference file for each of the tools.



1. Click **Reference file list > Create** in the Reference File Table.
2. Enter a name in the **Name** field. For example, enter **ReferenceFileList01**.
3. Enter a description for the reference file list in the **Description** field.
4. In the **Tools/Files** section, click the empty cell next to the **Schema** cell.
5. Click the Browse button next to the **Schema** field below the table.
6. Browse to the schema file, and click **Open**.
7. Repeat steps these for each tool map file that you want to reference, and click **OK**.

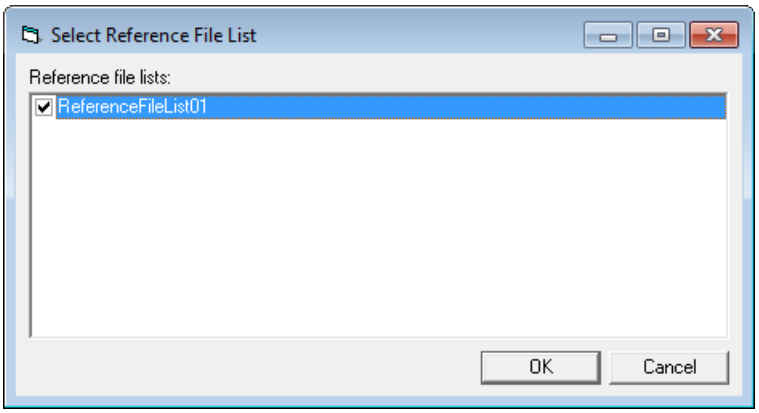
Edit an existing reference file list

The **Edit** command allows you to edit the active reference file list object. You can use this command to modify the path of the reference files used in the active list.

1. Click **Reference file list > Edit** in the Reference File Table.
2. Click on a file name in the **Reference Filename** column. For example, click on the file name for the **SmartPlant P&ID** row.
3. Click the Browse button next to the **SmartPlant P&ID** box.
4. Browse to the correct file, and click **Open**.
5. Click **OK**.

Set the active reference file list

The **Set active** command allows you to set the active reference file list object. You can use this command to switch between reference files.



1. Click **Reference file list > Set active** in the Reference File Table.
2. Select a reference file list in the **Select Reference File List** dialog box, and click **OK**.

Copy Objects from Reference File to Target File

The following topic provides an example of copying object properties from an older SmartPlant P&ID map file (the reference file) to a new SmartPlant P&ID map file (the target file) in order to speed up a new project.

NOTES

- You must check out the CMF file from Desktop Client before starting this procedure.
- A reference file list is required before entering the Integrated Reference File Environment. For more information on creating a reference file list, see the *Create a New Reference File List* section of *Configure the Reference File List* (on page 410).

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **Tools > Tool Schema > Integrated Reference File Environment**.

The Integrated Reference File Environment contains a left and right panel. The left panel contains a column that is a comparison of the schema file with its reference schema file and a column for each open tool schema file with its comparison to the corresponding reference tool schema file. The right panel contains a detailed comparison of any object you click in the left panel. From the right panel, you can select individual objects from the comparison and use them to add, update, or delete from the specific object selected in the left panel.

For example, you can add individual enumerations (if they exist in the reference file but not in the target file), delete enumerations (if they exist in the reference file but not in the target file), and update enumerations (if they exist in both the reference and target files, but are different).

3. Select a reference file list from the **Select Reference File List** dialog box, and click **OK**.

   NOTE If you do not have a reference file list, you can create one. For more information, see *Configure the Reference File List* (on page 410).

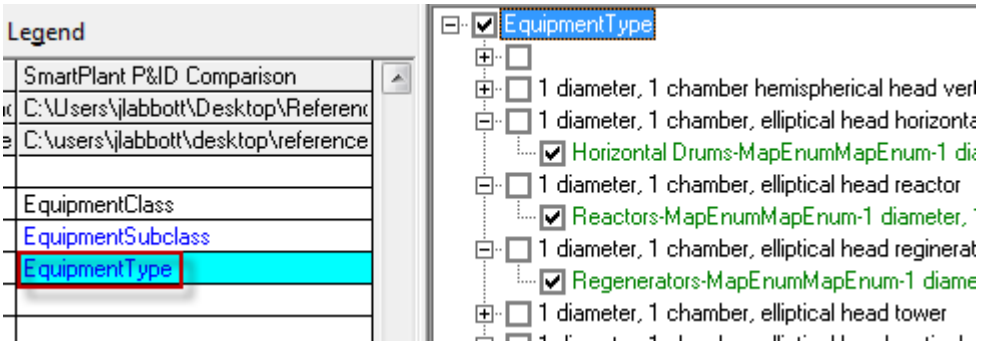
4. Select **Enumerated lists** from the **Show** menu.

5. Select a non-black object, or an object that has differences, from one of the tool map columns. For example, select the **EquipmentType** object in the **SmartPlant P&ID Comparison** column, which is blue.

   The right panel displays the differences between the target and reference files for this object.

6. Select one or more object properties in the right panel, and click **Apply** to copy them to the target file.

   TIP Click an item in the right panel to open the **View/Compare Objects** dialog box.



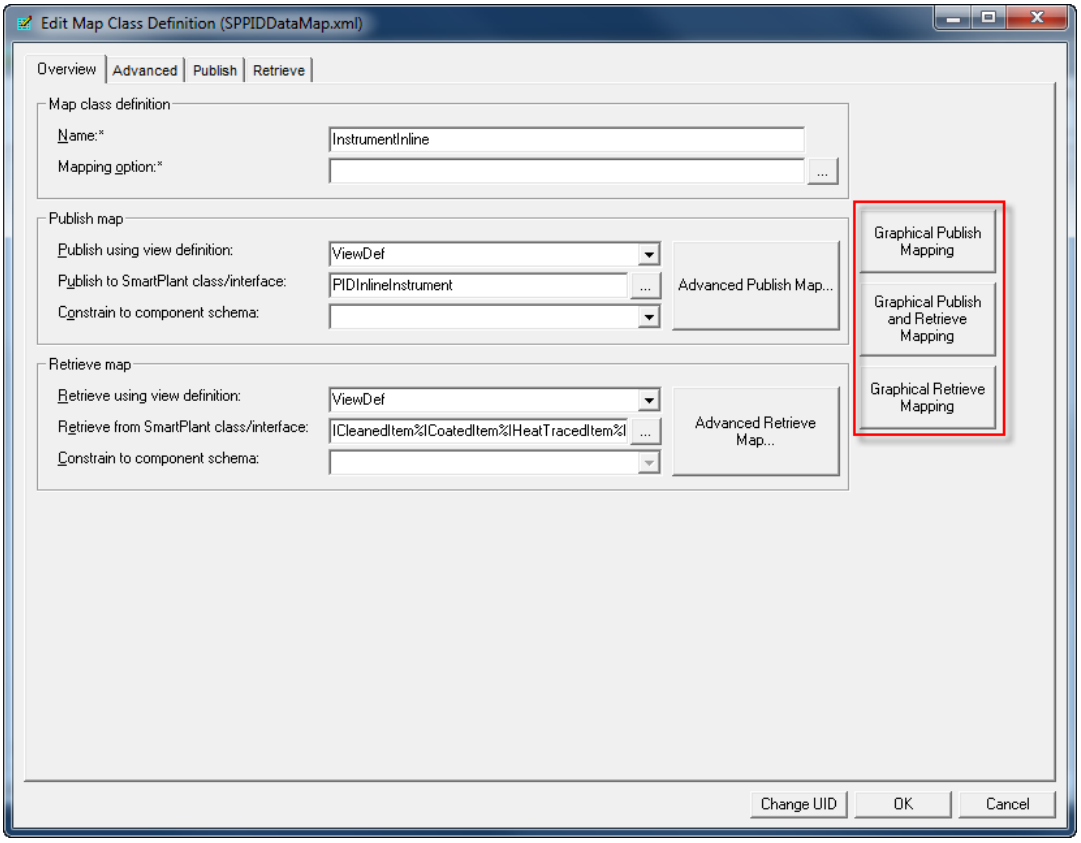
NOTES

- You can select the parent of an object in the hierarchical list to select all children under it. Objects that are black do not display a check mark if the parent is selected because they have no differences, but any non-black objects under the black object display a check mark. You may need to expand the item to see the check marks.These hidden relationships are automatically selected when a parent is selected.
- Click **Clear** to unselect all options in the right panel.
- Click **Create Report** to create a report of all differences displayed in the right panel. The report is created with your default text editor.
- Click **Show All** to display all objects, not just objects with differences. Click **Show Differences** to only display objects with differences.

Map Publish and Retrieve Properties in a Graphical View

The graphical mapping user interface in Schema Editor lets you easily map, view, and manage objects that are mapped into (publish) and out of (retrieve) the SmartPlant schema from tool map schemas. The graphical mapping user interface visually represents tool map schemas in a simple tree view to more easily see the various schema definitions of different applications and the relationships that have been mapped. If you are integrating with an application that supports map functions, the graphical mapping user interface allows the addition of map functions.

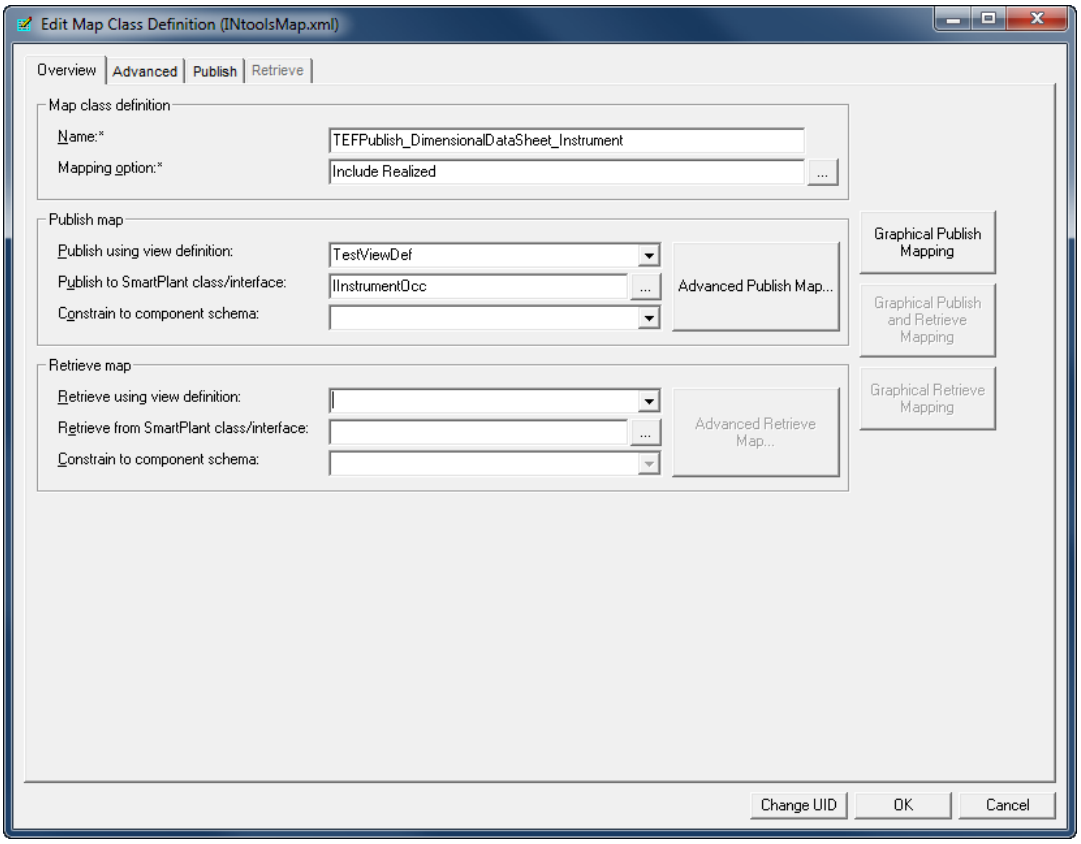
The **Graphical Publish Mapping**, **Graphical Publish and Retrieve Mapping**, and **Graphical Retrieve Mapping** buttons available from the **Overview** tab on the **Edit** window for map class definition can be used to map directly against the SmartPlant Schema.



When mapping to an interface definition for publish or from one or more interface definitions for retrieve, the view definition that is used includes all properties for all interfaces directly or indirectly implied by the mapped-to interface. This will likely include properties that are outside those used within a particular component schema and therefore properties that will not be published and/or retrieved.

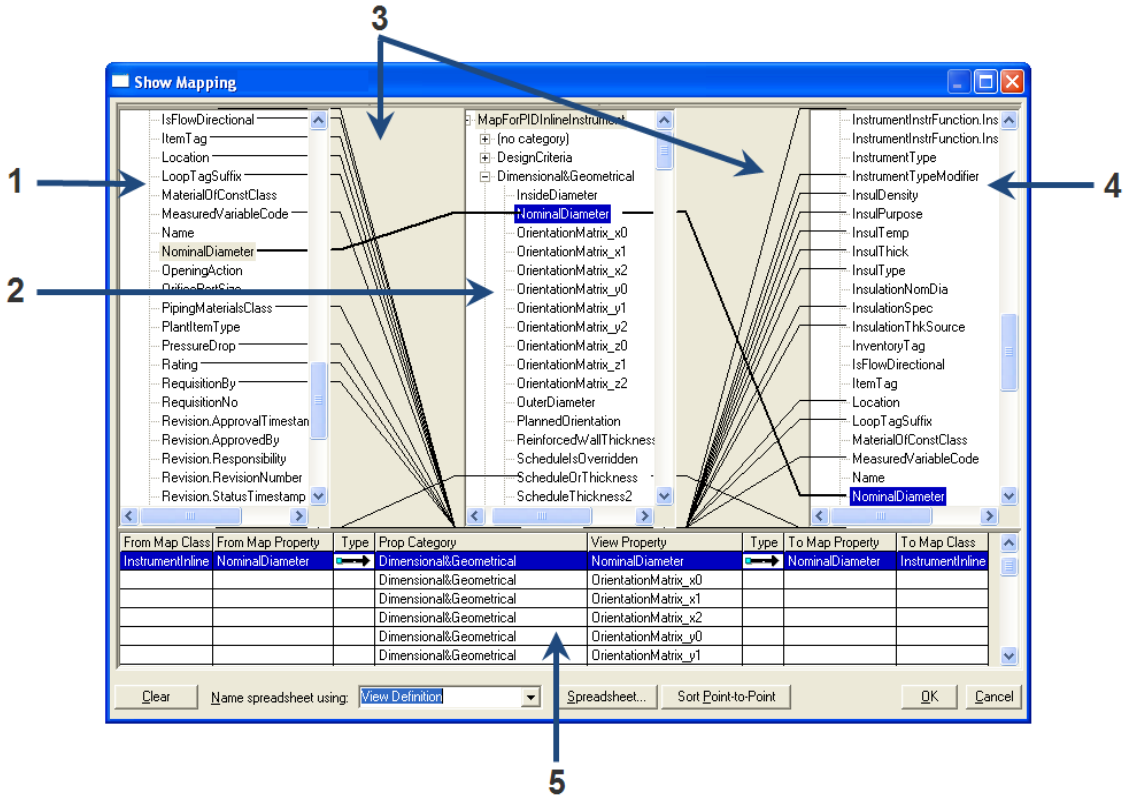
To limit the displayed properties to those that are on interfaces used within a particular component schema, that component schema should be identified though the **Constrain to component schema** list box. Specifying a component schema is optional for both publish and retrieve. However, if a component schema is specified, only those properties used within that component schema will be displayed through the **Publish** and **Retrieve** tabs as well as within the various graphical mapping displays.

The view definition name displayed in the **Publish using view definition** or **Retrieve using view definition** list boxes should update to reflect the view definition whenever a value is entered for the component schema though the **Constrain to component schema** list box.



The Graphical Mapping User Interface

Selecting the **Graphical Publish and Retrieve Mapping** for a map class definition displays the following window to allow mapping directly against the SmartPlant schema.



The graphical mapping user interface consists of:

1 Properties in the tool map schema that are available for publish

2 View definitions in the SmartPlant Schema to which publish and retrieve relationships can be added

3 The relationships lines indicated properties that have been mapped for publish or retrieve

4 Properties in the tool map schema that are available for retrieve

5 A tabular review that presents the same mapping information as selected in the tree view. The table view lets you easily reorganize data.

Create Mapping for Classes in Graphical Mapping

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

   TIPS

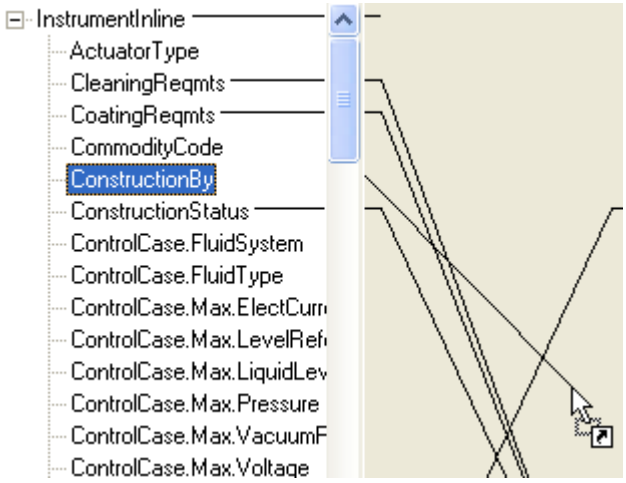
   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

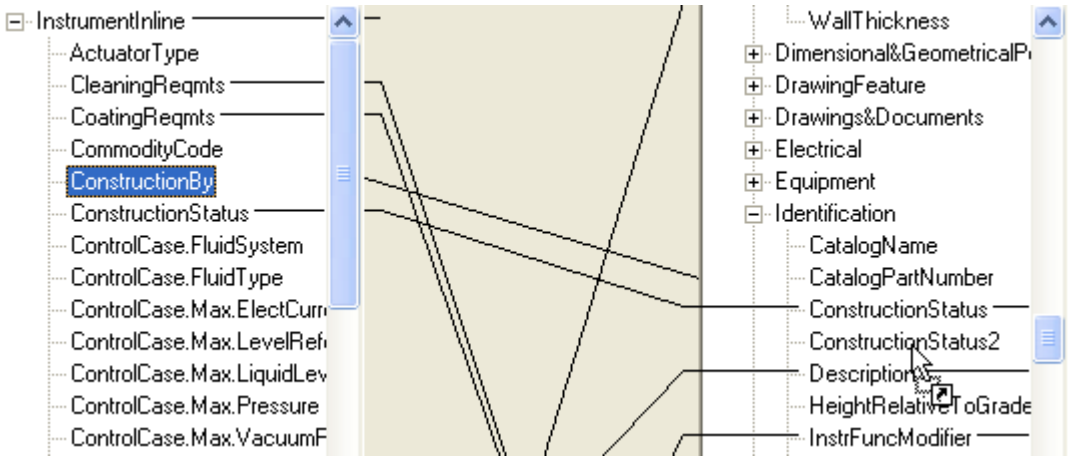
2. Click **View > Tool Schema**.
3. In the **SPFMapClass** tree view, right-click the class definition you want to map, then click **Edit <object>**. For example, from the tree view, right-click **InstrumentInline**, then click **Edit InstrumentInLine**.
4. On the **Overview tab**, click the mapping option you want. For example, click **Graphical Publish and Retrieve Mapping** to map both publish and retrieve properties.

   NOTE You can select **Graphical Publish Mapping** to only map properties for publish. You can select **Graphical Retrieve Mapping** to only map properties for retrieve.

   The **Show Mapping for** <*object*> window appears.
5. Select a property in the publish tree view (left) or the retrieve tree view (right) to create relationship for publish or retrieve. This example shows the creation of a publish relationship.
6. Drag your cursor toward the **View Property** tree and a Relationship Line appears.



7. Click the **View Property** in the middle tree view for which you want to create a publish relationship. A new relationship line should appear that connects the starting map property with the ending property definition.



TIPS

- You can create a publish or retrieve relationship by selecting a view definition property (middle tree view) as your starting point and dragging the relationship line to a map property in either the publish or retrieve tree.
- You can automatically map properties for publish by right-clicking a property in the publish tree and selecting one of the following from the shortcut menu:
  - **Auto-map property (publish)** - Provides a list of the best matches for the selected map property in the SmartPlant Schema. Selecting a property from this dialog box and clicking **OK** results in a relationship being created.
  - **Auto-create properties from selected map properties** - Creates a new property definition, adds the property definition to a view definition, and then creates a relationship between the selected map property in the tree view and the new property.
- You can automatically map properties for retrieve by right-clicking a property in the retrieve tree and selecting **Auto-map property (retrieve)** from the shortcut menu. **Auto-map property (retrieve)** provides a list of the best matches for the selected map property in the SmartPlant Schema. Selecting a property from this dialog box and clicking **OK** results in a relationship being created.

Create Mapping for Enumerated Lists in Graphical Mapping

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).

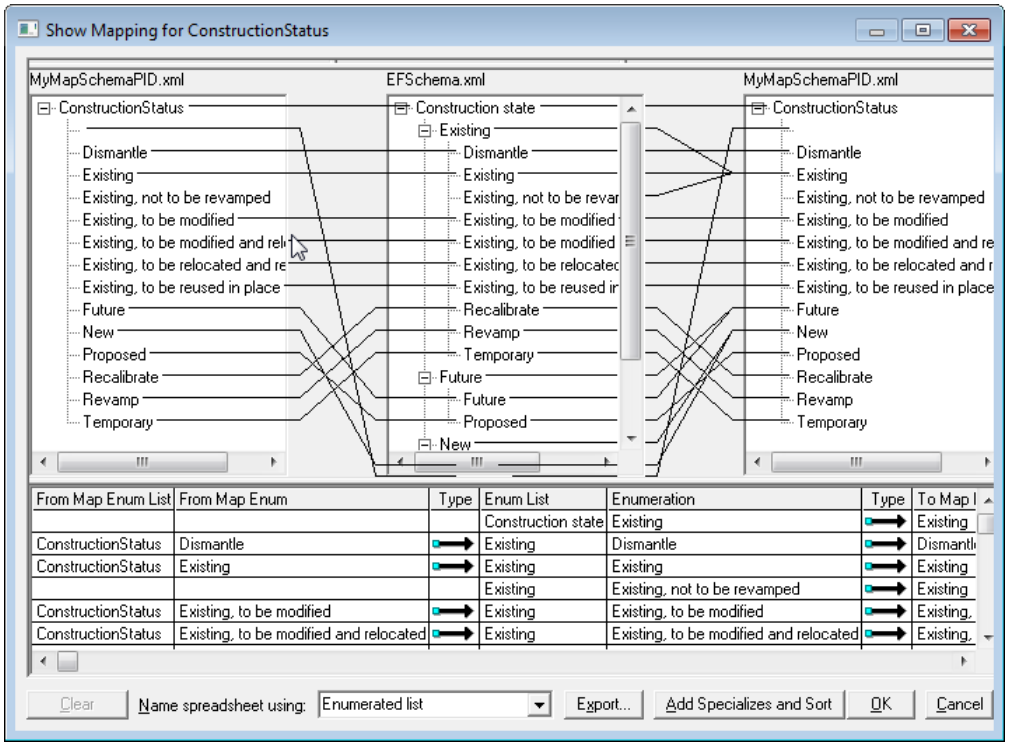
   TIPS

   - If you have checked the SmartPlant schema CMF file out of the Desktop Client, launched the Schema Editor, synchronized the tool map schema and the tool metadata, and are working with the tool's metadata adapter to define mapping, the authoring tool map schema is already open in the Schema Editor.
   - Before you can open a tool map schema, you must open a schema file.

     You can also open a file configuration or session file that contains the SmartPlant schema and the tool map schema files that you want to view.
   - The tool map schema file for each authoring tool is installed with the authoring tool. For more information about authoring tool map files and their locations, see your authoring tool documentation.

2. Click **View > Tool Schema**.
3. In the **SPMapEnumListDef**, right-click the enumerated list you want to map, then click **Edit** <*object*>. For example, right-click **ConstructionStatus**, then click **Edit ConstructionStatus**.
4. On the **Overview** tab, click the mapping option you want. For example, click **Graphical Publish and Retrieve Mapping** to map both publish and retrieve enumerations.

   NOTE You can select **Graphical Publish Mapping** to only map enumerations for publish. You can select **Graphical Retrieve Mapping** to only map enumerations for retrieve.



5. Select an enumeration in the publish tree view (left) or the retrieve tree view (right) to create relationship for publish or retrieve. This example shows the creation of a publish relationship.
6. Drag your cursor toward the mapped-to SmartPlant tree (middle) and a **Relationship Line** appears.
7. Click the property in the mapped-to SmartPlant tree for which you want to create a publish relationship.

   NOTES

   - You can create a publish or retrieve relationship by selecting a mapped-to SmartPlant property as your starting point and dragging the relationship line to an enumeration in either the publish or retrieve tree.
   - You can automatically map enumerations for publish by right-clicking an enumeration and selecting one of the following shortcut menus:
     - **Auto-map enumeration (publish)** - Provides a list of the best matches for mapping between the selected map enumeration and existing objects in the SmartPlant Schema. Selecting an enumeration from this dialog box and clicking **OK** results in a relationship being created.
     - **Auto-create enumerations from selected map enumerations** - Creates a new enumeration and then creates a relationship between the selected map enumeration in the tree view and the new enumeration.
8. Click **OK**.

Create Mapping for Units of Measure in Graphical Mapping

1. Open the tool map schemas that you want to view. For more information, see *Open a Tool Map Schema* (on page 98).
2. Click **View > Tool Schema**.
3. In the **SPMapUoMListDef**, right-click the unit of measure list you want to map, then click **Edit** <*object*>. For example, right-click **FlowRate, Mass**, then click **Edit FlowRate, Mass**.
4. On the **Overview** tab, click the mapping option you want. For example, click **Graphical Publish and Retrieve Mapping** to map both publish and retrieve properties.

   NOTE You can select **Graphical Publish Mapping** to only map properties for publish. You can select **Graphical Retrieve Mapping** to only map properties for retrieve.
5. Select a map unit of measure in the publish tree view (left) or the retrieve tree view (right) to create a relationship for publish or retrieve. This example shows the creation of a publish relationship.
6. Drag your cursor toward the mapped-to SmartPlant tree (middle) and a **Relationship Line** appears.

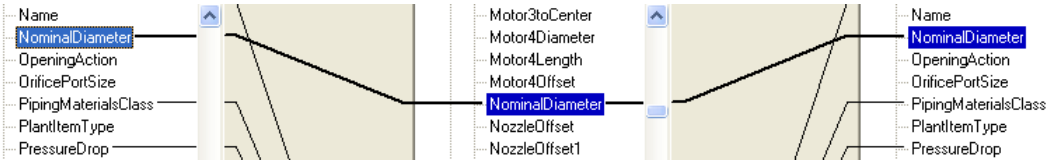
7. Click the map unit of measure in the mapped-to SmartPlant tree for which you want to create a publish relationship.

   NOTES

   - You can create a publish or retrieve relationship by selecting a mapped-to SmartPlant map unit of measure as your starting point and dragging the relationship line to a mapped unit of measure in either the publish or retrieve tree.
   - You can automatically map units of measure for publish by right-clicking a map unit of measure and selecting one of the following shortcut menus:
     - **Auto-map UoMs (publish)** - Provides a list of the best matches for the selected map unit of measure in the SmartPlant Schema. Selecting a unit of measure from this dialog box and clicking **OK** results in a relationship being created.
     - **Auto-create UoMs from selected map UoMs** - Creates a new unit of measure and then creates a relationship between the selected map unit of measure in the tree view and the new unit of measure.

View Mapping Relationships in Graphical Mapping

- Click a mapped property in any tree view to see the existing relationship.



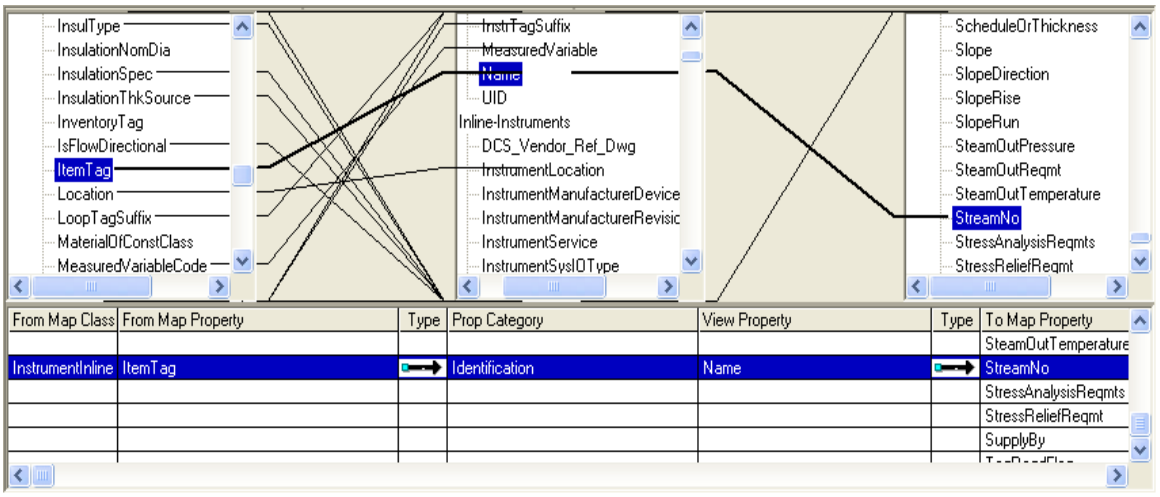
  The Relationship Line appears bold. The view property and related mapped property are automatically highlighted.

View and Map Properties in the Graphical Mapping Table

The view at the bottom of the graphical mapping user interface is a tabular representation of the same information as the tree and relationship views at the top.

You can sort the information in the tabular view by selecting a cell in the header row.

If you select a property from the table, the same property and any relationship lines will be highlighted in the graphical views.



The following shortcut menus are available by right-clicking the various cells within the table.

- **Edit** <*From Map Property*> - Allows you to edit the property to be published from the tool map schema using the **Edit Map Property Definition** dialog box.
- **View** <*View Property*> - Allows you to edit the view property in the SmartPlant schema using the **Edit Property Definition** dialog box.
- **Edit** <*To Map Property*> - Allows you to edit the property to be retrieved from the tool map schema using the **Edit Map Property Definition** dialog box.
- **Delete MapPropertyToProperty relationship between** <*From Map Property*> **and** <*View Property*> - Removes the relationship between the property being published in the tool map schema and the SmartPlant schema.
- **Delete PropertyToMapProperty relationship between** <*View Property*> **and** <*To Map Property*> - Removes the relationship between the property being retrieved in the tool map schema and the SmartPlant schema.
- **Add/delete displayed from map properties** - Allows you to add or remove columns used to display information about map properties available for publish.
- **Add/delete displayed schema properties** - Allows you to add or remove columns used to display information about SmartPlant Schema properties.
- **Add/delete displayed to map properties** - Allows you to add or remove columns used to display information about map properties available for retrieve.
- **Sort map data** - Allows you to select the properties by which to sort the table.

  NOTE You can sort the table by selecting a cell in the header row. For example, selecting the View Property header will sort the contents of the table using the values within that column.
- **Repaint** - Updates the table to the default view and removes any sort criteria.

Filter Options for Graphical Mapping

You can filter the items displayed in the tree views to show only the properties you want to map.

In the publish (left) tree view, you can do the following to filter the display:

- **Set map properties to be ignored** - Sets the properties that are not to be used for either publish or retrieve and removes the properties from display in both the publish (left) and retrieve (right) tree views.
- **Set map properties not to be used for publish mapping** - Sets the properties that are not already mapped in the tool schema to not be used for publish and removes the properties from display in the publish (left) tree view.
- **Set view properties not to be used for publish mapping** - Sets the view properties in the SmartPlant schema that are not to be used for publish. The view definition properties (middle) tree view is not updated when this option is set.

In the view definition properties (middle) tree view, you can do the following to filter the display:

- **Filter using publish from <object>** - This option filters the display to not show view properties that have been defined to be filtered out for publish operations.
- **Filter using retrieve from <object>** - This option filters the display to not show view properties that have been defined to be filtered out for retrieve operations.

In the retrieve (right) tree view, you can do the following to filter the display:

- **Set map properties not to be used for retrieve mapping** - Sets the properties that are not already mapped in the tool schema to not be used for retrieve and removes the properties from display in the retrieve (right) tree view.
- **Set view properties not to be used for retrieve mapping** - Sets the view properties in the SmartPlant schema that are not to be used for retrieve. The view definition properties (middle) tree view is not updated when this option is set.

Display Options for Graphical Mapping

You can set options for the behavior and appearance of tree views in the graphical mapping user interface. If you right click within the tree window, you can do one of the following:

- **Resize tree** - Sets the width of the tree view. A separate window appears

  NOTE The height is adjusted using the separator bar between the graphical view and the tabular view.
- **Only show mapped properties**. - Displays only properties (or map properties) that have at least one mapped relationship in the tree view. If this command is selected, then the next popup menu will present the option to toggle the display back to showing all properties (or map properties).
- **Show all properties** / **Show all map properties** - Displays all properties (or map properties) in the tree view. This option only appears if **Only show mapped properties** is selected.
- **Turn publish/retrieve relationships off** - Removes all the lines for the tree control and no longer displays the mapping relationships.

- **Turn publish/retrieve relationships on** - Displays the relationship lines for publish or retrieve to show mapping relationships.
- **Turn header label on/off** - Turns on or off the label showing the source for the map class definition and the schema view definition. The label appears directly above the tree view.

If you right-click within the relationship area, you can do one of the following:

- **Show toolbar** - Displays the tool bar
- **Copy to Clipboard** - Copies the display within the graphical mapping user interface to the clipboard. The display can then be pasted into other programs, such as Microsoft Word.
- **Print Form** - Prints a copy of the graphical section of the mapping user interface to the default printer.
- **Align** - Aligns the tree views horizontally and vertically. This option should be used when the graphical mapping user interface is used in Layout mode.
- **Stop lines at control edge** - Stops the relationship lines at the edges of the tree view frame.
- **Extend lines to text** - Extends the relationship lines to the property names in the tree view.

Resize the Graphical Mapping Tree View

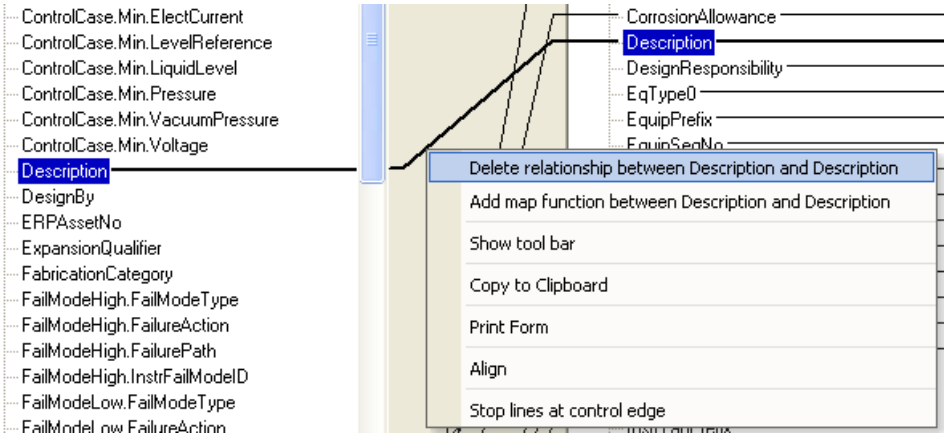
1. Right-click within a tree view window, then click **Resize tree**.

   A **Resize Tree** window appears.

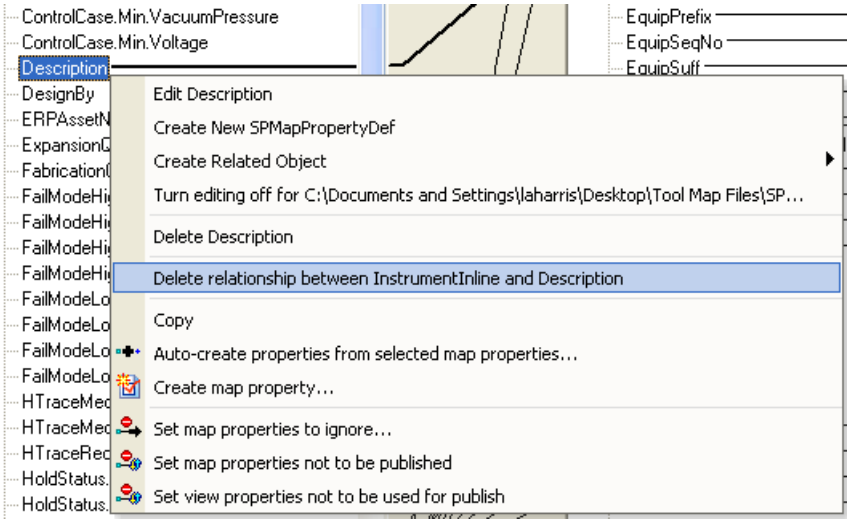
2. To change the width, point to the left or right window border. When the pointer changes into a horizontal double-headed arrow, drag the border to the right or left.
3. Click  when the window is the width you want. The tree view resizes to the selected width.

Delete Relationships for Graphical Mapping

- To delete the mapping relationship between two properties, right-click the **Relationship Line** between the two properties from which you want to remove the relationship, then click **Delete relationship between** <*property*> **and** <*property*>.



- To delete the relationship between an object and a property within an object; for example, a class definition in a tool map schema and a property within that class definition, right-click the property, then click **Delete relationship between** <*object*> **and** <*property*>.



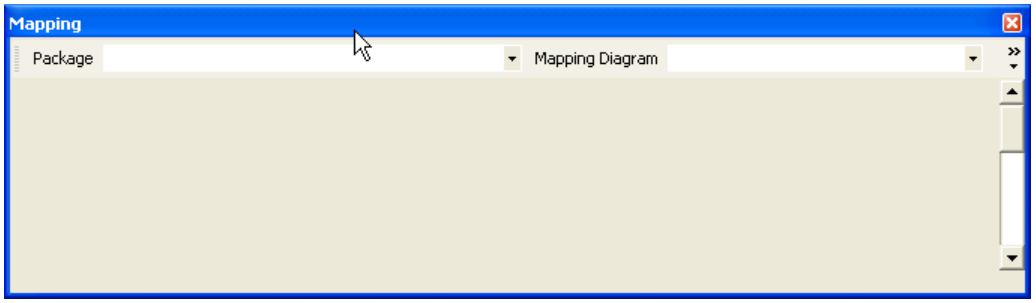
Create a Layout to Display Graphical Mapping Views

Layouts allow you to organize windows and views in the Schema Editor to help you work the way you want. To learn more, see *Layouts* (on page 69) in the *Schema Editor User's Guide.*

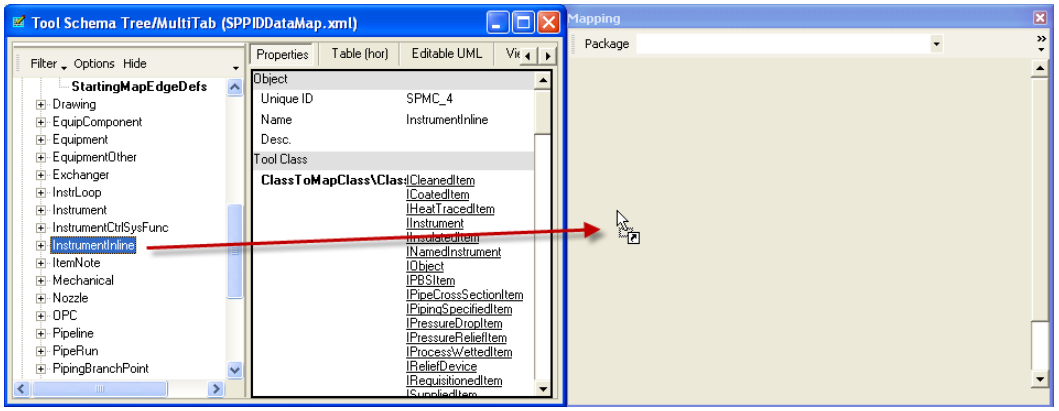
The following example explains how to create a layout view of the graphical mapping interface.

1. Open the active schema and the tool schemas you want to map.

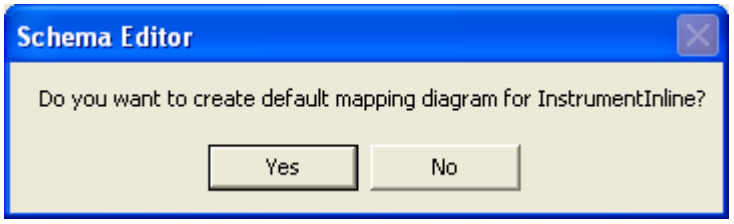
2. Click **Layout > Mapping**. The following window appears.



3. Drag an object, for example, the **InstrumentInline** map class definition, from the tool schema to the **Mapping** window.



4. In the **Schema Editor** message box, click **No** to create a customized mapping diagram.



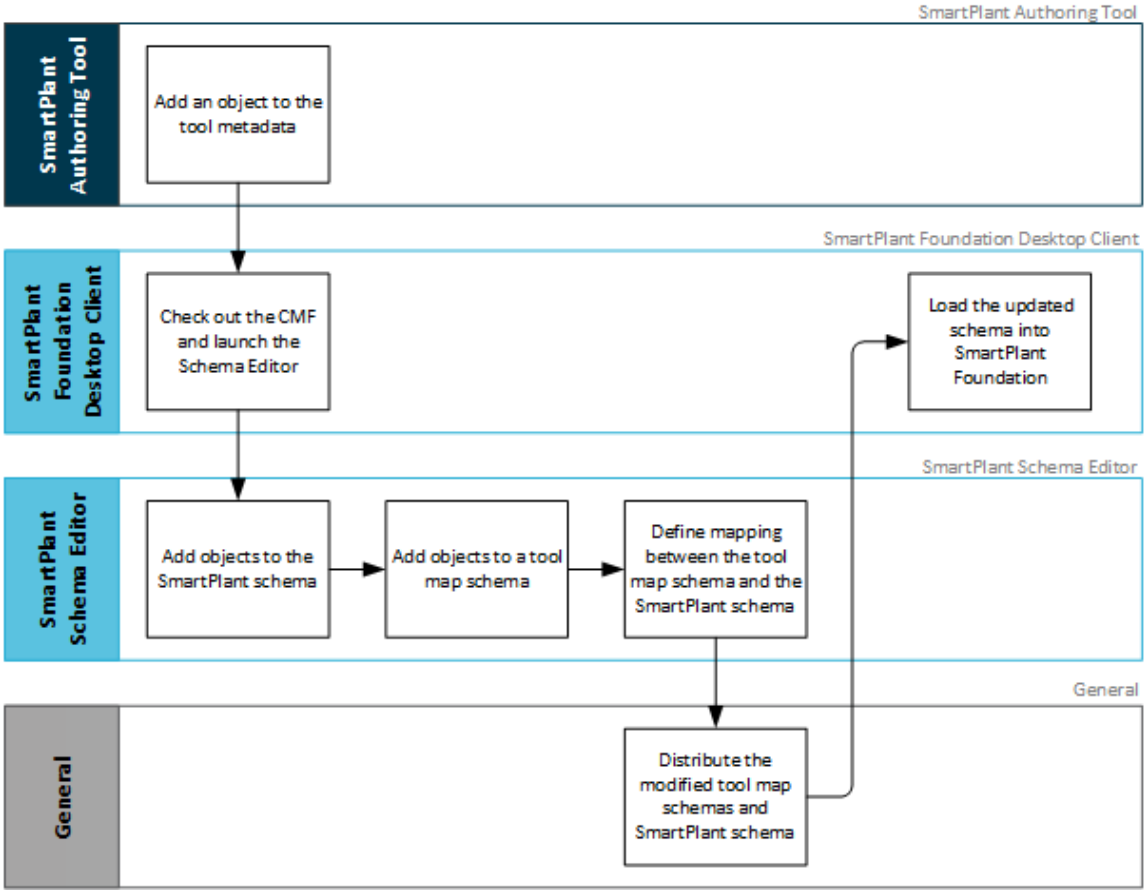
TIP To display the default mapping diagram for the selected object type, click **Yes** in the **Schema Editor** message box.

5. Define a display name and package for the new mapping diagram on the **Overview** tab on the **New Mapping Diagram** dialog box.
6. Click **OK** and the property tree for the object appears.
7. Move the tree view around as needed.
8. Right-click in a tree to change the display of the object.
9. Add additional objects to the **Mapping** window by browsing in the tool schema and dragging object from the tree view for the toll schema to the mapping diagram.
10. Add the view definition for the class.
    - Click **Layout >Schema**
    - Click **Filter > Select > Select Other Classes**, then select **Graphs/Views** from the list.

- Expand **ViewDef** in the tree view and browse to the view definition; for example, **PIDInlineInstrument_Map**.
- Drag **PIDInlineInstrument_Map** to the **Mapping** window.

Work Process for Mapping for Tools without Metadata Adapters

The following graphic provides an overview of the process for defining authoring tool mapping for tools that do not deliver metadata adapters. The graphic covers the work process for adding and mapping objects that do not already exist in the tool database, tool map schema, or SmartPlant schema. If the object already exists in the tool map schema or the SmartPlant schema, you can skip the steps that involve adding new objects in the work process.



What do you want to do?

- *Add an Object to the Tool Metadata* (on page 428)
- *Check out the CMF and Launch the Schema Editor* (on page 428)
- *Add Objects to the SmartPlant Schema* (on page 429)

- *Add Objects to a Tool Map Schema* (on page 429)
- *Define Mapping Between the Tool Map Schema and the SmartPlant Schema* (on page 441)
- *Distribute the Modified Tool Map Schema and SmartPlant Schema* (on page 462)
- *Load the Updated Schema into SmartPlant Foundation* (on page 462)

Understanding Mapping

Before you begin defining authoring tool mapping, you must understand the following:

- The SmartPlant schema model and how it works. For more information, see *SmartPlant Schema Overview* (on page 18) and *A Gentle Introduction to the SmartPlant Schema*, delivered with SmartPlant Foundation.
- How tool map schemas work and how they are used by the authoring tools. For more information, see *Tool Map Schemas* (on page 259).
- The authoring tool data model for which you are defining mapping. For more information, see your authoring tool documentation.
- The delivered tool map schema and mapping for the authoring tool. For more information see *Viewing Authoring Tool Mapping* (on page 307) and rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.
- Distributing the modified tool map schema to every computer that requires it. See *Distribute the Modified Tool Map Schema and SmartPlant Schema* (on page 462).

Add an Object to the Tool Metadata

If you want to define mapping between an authoring tool and the SmartPlant schema, the object that you want to map must first exist in the authoring tool metadata database. Any supporting objects, such as enumerated lists or units of measure, must also exist in the tool database so that the objects can be published and retrieved. For example, if you add a new property to your authoring tool database and values for that property are defined by an enumerated list, you must also define the enumerated list in the tool database.

NOTE The procedure for adding new objects to the tool database varies depending on the authoring tool you are using. For example, to add new objects to SmartPlant P&ID, you must use Data Dictionary Manager. For more information about adding new objects to the tool database, see the appropriate authoring tool documentation.

Check out the CMF and Launch the Schema Editor

If you need to add new properties, enumerated lists, list values or other objects to the SmartPlant Schema, you must first check it out of SmartPlant Foundation Desktop Client. Even if you do not need to extend the SmartPlant schema, you must open a copy of it to create mapping relationships.

- Check out the SmartPlant schema CMF file from the SmartPlant Foundation Desktop Client.

- Make sure your scope is set to the plant, right-click the new revision of the CMF file in SmartPlant Foundation, and click **Launch Schema Editor**. This provides the Schema Editor with the appropriate plant and SmartPlant Foundation site for connection.
- Select the schema version or versions that you want to view in the **Specify configurations to display** window of the **Active Configurations** dialog box.
- Click the **Advanced** button on the **Active Configurations** dialog box to view the **Set Configurations for new objects** window, where you can specify what version or versions of the schema changes should be written to. For example, to modify the 2016 schema, select the 2016 configuration for new objects.

NOTE By default, all configurations are checked to allow forward and backward compatibility between schema versions when creating new objects or modifying existing objects.

Add Objects to the SmartPlant Schema

The next step for mapping is to verify that the objects that you want to map exist in the SmartPlant schema. If the objects do not exist, you can create them in the SmartPlant schema using the Schema Editor. For more information about extending the delivered SmartPlant schema, see *Extending the SmartPlant Schema* (on page 146).

Add Objects to a Tool Map Schema

After you verify that the objects that you want to map exist in the SmartPlant schema, you can add any new objects that you want to map to the authoring tool map schema in the Schema Editor.

The authoring tool adapters define the rules for correlating objects in the tool map schema to objects in the tool database for publish and retrieve. For example, some adapters correlate using the name of the objects; others require that the UIDs from the tool database be defined in the tool map schema. For more information about authoring tool mapping, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or the tool documentation.

For more information about extending the SmartPlant schema, see *Extending the SmartPlant Schema* (on page 146).

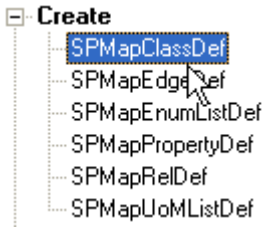
In general, you can add the following objects to a tool map schema and then define the mapping between those objects and the SmartPlant schema:

- Map class definitions (instances of SPMapClassDef)
- Map property definitions (instances of SPMapPropertyDef)
- Map relationship definitions (instances of SPMapRelDef)
- Map enumerated list definitions (instances of SPMapEnumListDef)
- Map enumerations (instances of SPMapEnumDef)
- Map unit of measure list definitions (instances of SPMapUoMListDef)
- Map units of measure definitions (instances of SPMapUoMDef)
- Map software definition (maximum of one instance of SPMapSoftware)
- Relationships between map objects and other map objects (map schema) and between map objects and SmartPlant objects (mapping)

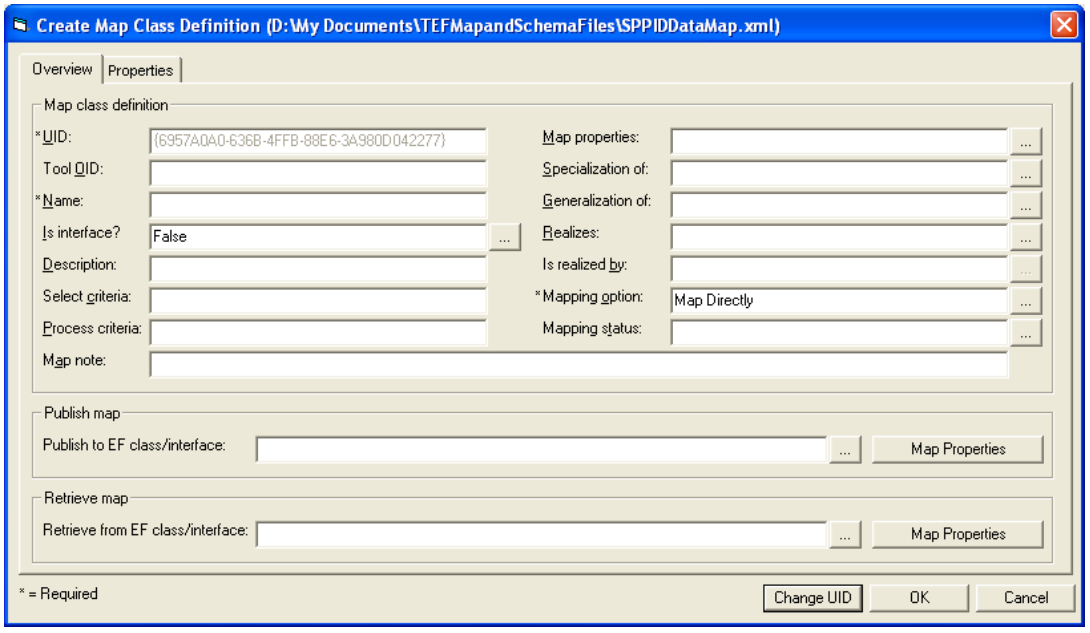
Create a Map Class Definition

1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**, and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in **Simple** mode. Click **Editor** in the **Form** list instead.
4. In the **Create** tree, double- click **SPMapClassDef**.



   TIP You can also create map class definitions by double-clicking the **SPMapClassDef** node in any tree view in any Schema Editor view.
5. In the **Name** box, type a unique name for the map class definition.



6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool class definition. See the mapping rules for your authoring tool for how to set the value for this property.
7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool class definition. See the mapping rules for your authoring tool for how to set the value for this property.

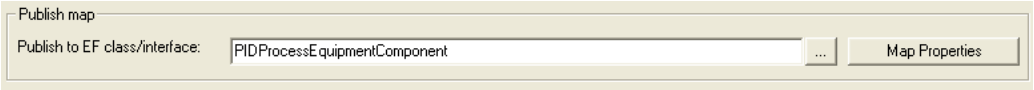
TIP For example, if you want SmartPlant P&ID to publish extended properties for equipment, such as process data properties, you can set the process criteria of the Equipment map class to EFAdapter.EqDataExchanger,True in the SmartPlant P&ID tool map schema.

8. To define the properties associated with the tool map class, click ... beside the **Map properties** box, and select the authoring tool map properties for the class definition.

   TIPS

   - If the properties that you want to associate with the map class do not already exist in the tool map schema, you can click **Create** in the **Possible MapProperties** dialog box to create them. For more information, see *Create a Map Property Definition* (on page 433).
   - As a general rule, when you define map class definitions for a tool class/interface model, the **Map properties** box should be blank; in a typical class/interface model, only interfaces (not classes) expose properties.

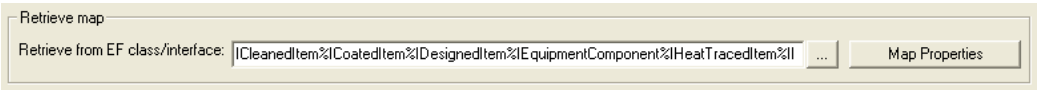
9. Under **Publish map**, click ... beside the **Publish to SmartPlant Class/Interface** box, and select the SmartPlant schema class definitions or interface definitions to which you want to map for publish operations.



   TIPS

   - Most mapping between map class definitions and the SmartPlant schema should be done using SmartPlant interface definitions.
   - If you map to interface definitions, you should map to the primary interface definitions for the SmartPlant objects that the tool class definition corresponds to.

10. Under **Retrieve map**, click ... beside the **Retrieve to SmartPlant Class/Interface** box, and select the SmartPlant schema class definitions or interface definitions that map to the map class definition for retrieve operations.



   TIP For retrieval, mapping should be defined using interface definitions rather than class definitions.

11. If the map properties for this map class definition already exist in the tool map schema, define the publish and retrieve maps for the properties under **Publish map** and **Retrieve map**. For more information, see *Define Publish Mapping for Tool Properties* (on page 442) or *Define Retrieve Mapping for Tool Properties* (on page 447).
12. Click **OK** to save the map class definition.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

NOTES

- When you create a map class definition, the Schema Editor automatically assigns a UID to the map class definition. If you want to change the UID, click **Change UID**. See the mapping

rules for your authoring tool for information on UID values to use. Some application adapters will not work correctly if the values for the UID do not follow the rules for that application.

- The application schema model that you are using determines the value for the **Is interface?** option and indicates whether the **Specialization of**, **Generalization of**, **Realizes** and **Is realized by** boxes require a value for the map class definition.
  - If the application model is a simple class/property model, then **Is interface?** should be **False**, and the **Specialization of**, **Generalization of**, **Realizes**, and **Is realized by** boxes should be blank.
  - If the application model is a class inheritance model, **Is interface?** should be **False**, and the **Realizes** and **Is realized** by boxes should be blank. If the tool map class inherits from other tool classes, this inheritance should defined in the **Specialization of** box. Likewise, if other tool classes specialize this tool class, this should be reflected in the **Generalization of** box.
  - If the application model is a class/interface model, then both classes and interfaces are modeled as tool map class definitions. The **Is interface?** option for a particular map class definition should reflect whether that tool class refers to an interface (**True**) or a class (**False**).

    For both classes and interfaces, the **Specialization of** and **Generalization of** boxes should be blank. For interfaces, the **Realizes** box should be blank, and the **Is realized by** box should indicate the classes that realize this interface. For classes, the **Is realized by** box should be blank, and the **Realizes** box should indicate the interfaces realized by this class.
- The **Mapping option** box identifies the type of mapping that you can use for the map class definition. The possible values include:
  - **Don't Map**
  - **Map Directly** - The properties for the tool class or interface map directly to a SmartPlant schema class definition or interface definition. The map properties that can be used for mapping are defined for this tool class only. **Map Directly** is the default mapping type.

    You should use this option if the properties for the tool class map exactly the same regardless of the tool class that specializes them (class inheritance model) or the tool class that realizes them (class interface model).
  - **Include Generalized** - If the tool map schema is using a class inheritance model and the **Map directly** option is not satisfactory for the mapping, this option should be used. The map properties that can be used for mapping for this option are the map properties for this tool class and the map properties for any generalized class (including generalizations of generalized classes or the entire inheritance tree).

    This option allows different mapping for the properties of the generalized tool classes depending on the specialized tool class. For example, one specialized tool class can map the properties for a generalized tool class differently than another specialized tool class maps those same properties.

    If the mapping of one or more properties on a generalized map class is different depending on the specialized map class definition, then the specialized map class definitions should use this option.
  - **Include Realized** - If the tool map schema uses a class/interface model and the properties on a map interface (map class definition for which **IsInterface=True**) map differently depending on the map class definition that realizes that map interface, use

the **Include Realized** mapping option for the map class definitions that realize that map interface.

This option allows different mapping for the properties of a map interface based on the realizing map class definition. One realizing map class can map the properties for a realized map interface differently than another realizing map class maps those same properties.

Map interfaces (map class definitions for which **IsInterface=True**) should use the **Map Directly** option if one or more properties on that map interface always maps the same way regardless of the realizing map class definition. Use the **Don't Map** option if all the mapping is different depending on the realizing map class definition.

The **Include Realized** option should not be used with map interfaces. Only the map class definitions that realize the map interface should use this option.

Create a Map Property Definition

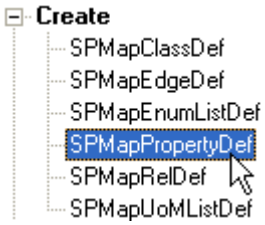
1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.

   TIP If you are creating a new map property definition in the **Map Environment** for an authoring tool that uses a metadata adapter, click **New Map Property Definition** on the **Publish** or **Retrieve** tab for the map class definition and skip to step 5.

2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**, and then click **OK**.

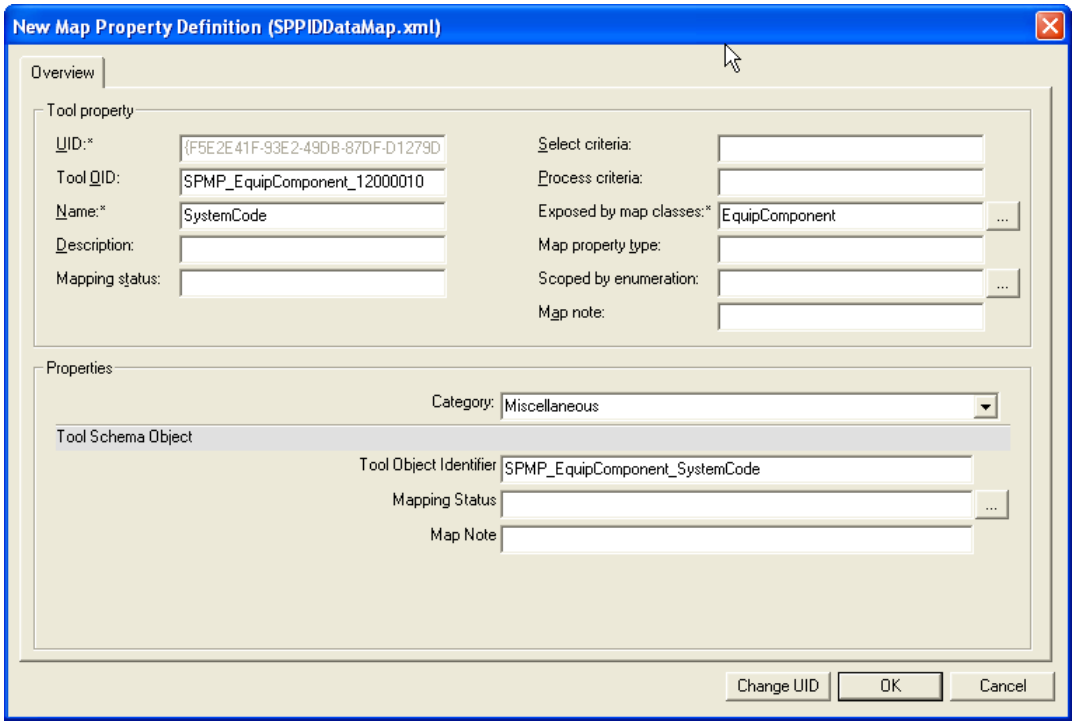
   TIP If you do not see the **View** tab, the **View** dialog box is in **Simple** mode. Click **Editor** in the **Form** list instead.

4. In the **Create** tree, double- click **SPMapPropertyDef**.



TIP You can also create map property definitions by double-clicking the **SPMapPropertyDef** node in any tree view in any Schema Editor view.

5. In the **Name** box, type a name for the map property definition.



TIP Unlike the SmartPlant schema where a property name can only be used once, property names can be reused in the tool map schema for different map classes.

6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool property definition. See the mapping rules for your authoring tool for how to set the value for this property.

7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool property definition. See the mapping rules for your authoring tool for how to set the value for this property.

   TIP For example, if the tool adapter must do a conversion on an enumerated list or the unit of measure for a property before the property is published to SmartPlant, use the **Process criteria** box to specify the method used for the conversion, for example, EFAdapter.EnumConversions.

8. To define the tool map class definitions that expose this map property definition, click … beside the **Exposed by map classes** box, and select the authoring tool class definitions.

   TIP If the map classes that you want to associate with the map property definition do not already exist in the tool map schema, you can click Create in the Possible MapClass Values dialog box to create them. For more information, see *Create a Map Class Definition* (on page 430).

9. If the map property definition has values defined by an enumerated list, click … beside the **Scoped by enumeration** box, and select the tool map enumerated list definition that scopes this tool map property definition.

   TIP If the map enumerated list that you want to associate with the map property definition does not already exist in the tool map schema, you can click **Create** in the **Possible**

MapEnumList Values dialog box to create it. For more information, see *Create a Map Enumerated List Definition* (on page 437).

10. Click **OK** to save the map property definition.

IMPORTANT Defining mapping for an object does not necessarily mean that you can publish and retrieve that object. The authoring tool adapter determines the objects that you can publish and retrieve. For more information, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

NOTE When you create a map property definition, the Schema Editor automatically assigns a UID to the map property definition. See the mapping rules for your authoring tool for information on UID values to use. Some application adapters will not work correctly if the values for the UID do not follow the rules for that application.

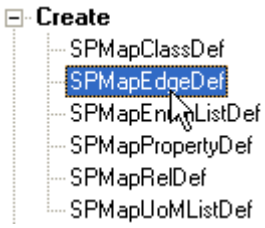
Create a Map Edge Definition

If you want to define many-to-one or many-to-many mapping between the tool map schema and the SmartPlant schema, you must create map edge definitions that traverse the relationship between two authoring tool class definitions. Map edge definitions allow you to traverse relationships between map class definitions to locate properties that you want to map.

Any map edge definitions defined as part of the tool map schema must be explicitly supported by the adapter for that tool. Unless the tool adapter has been defined to allow user definition of map edge definitions or the user defining the map edge definition has access to the tool adapter code and can modify it to recognize the new map edge definition, any map edge definitions defined by the user will not be recognized by the tool adapter and will not be supported for publish and retrieve operations. Therefore, new map edge definitions should only be created if the underlying tool adapter can either be modified or if the adapter supports user-added map edge definitions. See your authoring tool mapping rules for more information.

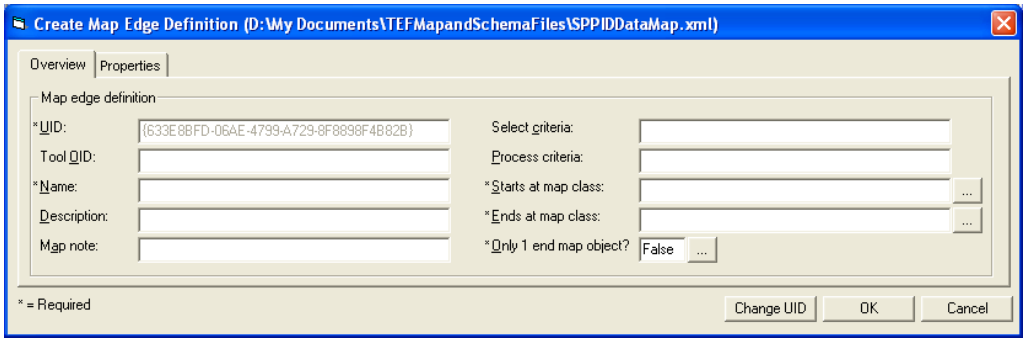
1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**, and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in **Simple** mode. Click **Editor** in the **Form list** instead.
4. In the **Create** tree, double-click **SPMapEdgeDef**.



TIP You can also create map edge definitions by double-clicking the **SPMapCEdgeDef** node in any tree view in any Schema Editor view.

5. In the **Name** box, type a unique name for the map edge definition.



6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool edge definition. See the mapping rules for your authoring tool for how to set the value for this property.
7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool edge definition. See the mapping rules for your authoring tool for how to set the value for this property.
8. To define the starting map class definition for the edge definition, click … beside the **Starts at map class** box, and select the starting tool map class.

   TIP If the map class definition that you want to use as the starting point for the map edge definition does not already exist in the tool map schema, you can click **Create** in the **Possible StartMapClassDef** dialog box to create it. For more information, see *Create a Map Class Definition* (on page 430).
9. To define the ending map class definition for the edge definition, click … beside the **Ends at map class** box, and select the ending tool map class.
10. If you to be able to map to both ends of the edge definition, click … beside the **Only 1 end map object?** box, and then select **True**.
11. Click **OK** to save the new map edge definition.

NOTES

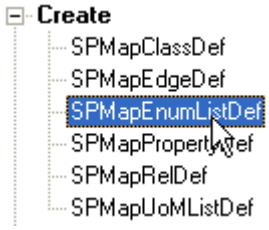
- The authoring tool adapter determines whether one-to-many, many-to-many, and many-to one mapping is supported. For more information, see the rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.
- When you create a map edge definition, the Schema Editor automatically assigns a UID to the map edge definition. If you want to change the UID, click **Change UID**. See the mapping rules for your authoring tool for information on UID values to use. Some application adapters will not work correctly if the values for the UID do not follow the rules for that application.

Create a Map Enumerated List Definition

When you create a map enumerated list definition, you must create the enumerated list definition and then define the enumerated list entries that the list contains.

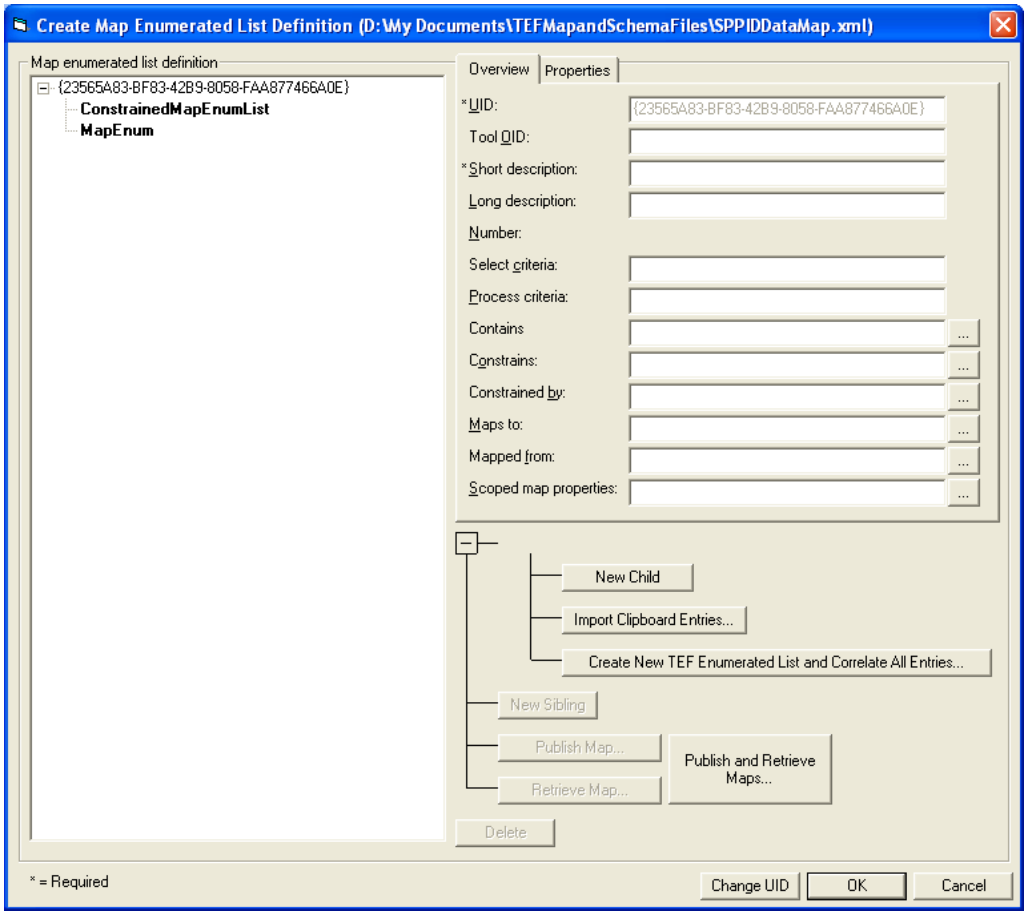
1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**, and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in **Simple** mode. Click **Editor** in the **Form** list instead.
4. In the **Tool Schema Editor UML** window, double-click **SPMapEnumListDef** in the **Create** tree.



TIP You can also create map enumerated list definitions by double-clicking the **SPMapEnumListDef** node in any tree view in any Schema Editor view.

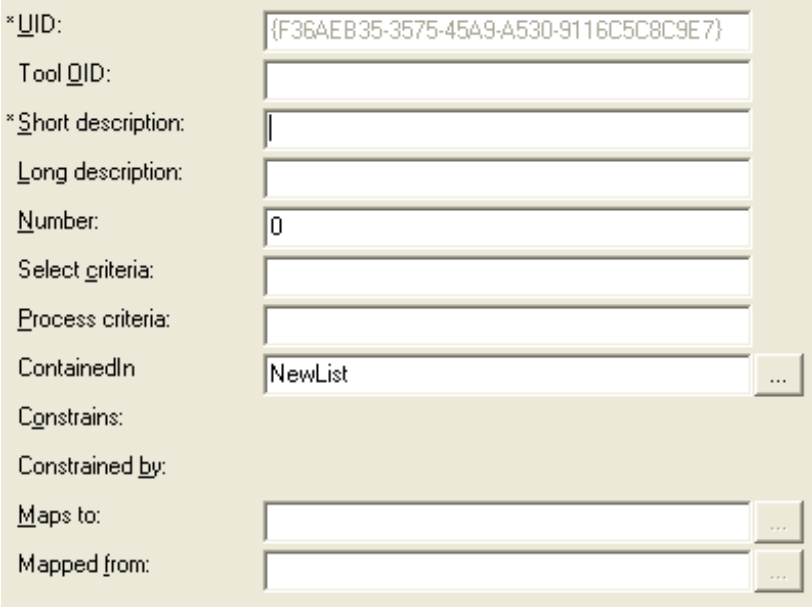
5. In the **Short description** box, type the name of the enumerated list.



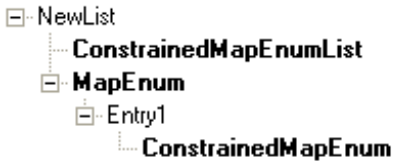
6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool enumerated list definition and its entries. See the mapping rules for your authoring tool for how to set the value for this property.
7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool enumerated list definition and its enumerations. See the mapping rules for your authoring tool for how to set the value for this property.
8. To define the SmartPlant schema enumerated list type that the tool enumerated list definition maps to for publish, click … beside the **Maps to** box, and select the SmartPlant enumerated list type.
9. To define the SmartPlant schema enumerated list type that maps to the tool map enumerated list definition for retrieve, click … beside the **Mapped from** box, and select the SmartPlant enumerated list type.
10. To define the tool map properties that this list defines values for, click … beside the **Scoped map properties** box, and select the map property definitions that are scoped by this map enumerated list.
11. To create an enumerated list entry (MapEnum) for this map enumerated list, click **New Child** in the **New Map Enumerated List Definition** dialog box.

TIP If you have a spreadsheet containing your list entries, you can copy them to the Clipboard and then add them to the list using the **Import Clipboard Entries** button. For more information, see *Import Enumerated List Entries from a Spreadsheet* (on page 174).

12. In the **Short description** box, type the name of the enumerated list entry.



TIP When you move out of the **Short description** box, the new enumerated list entry is automatically added to the **Map enumerated list definition** tree.



13. If a secondary numeric key identified the enumerated list entry, type the key in the **Number** box.
14. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool enumerated list entry. See the mapping rules for your authoring tool for how to set the value for this property.
15. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool enumerated list entry. See the mapping rules for your authoring tool for how to set the value for this property.
16. To create additional list entries, click **New Sibling**, and repeat steps 12-15 for each entry that you want to add.
17. To save your changes, click **OK**.

IMPORTANT For some authoring tools, you must change the system-generated UID so that the adapter can correlate the enumerated list and its entries with the appropriate enumerated list and entries in the tool database. For example, for SmartPlant P&ID, you must replace the system- generated UID for the map enumerated list and the list entries with UIDs from the

SmartPlant P&ID database. For more information, see the rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation. To change the system-generated UID, click **Change UID**.

NOTES

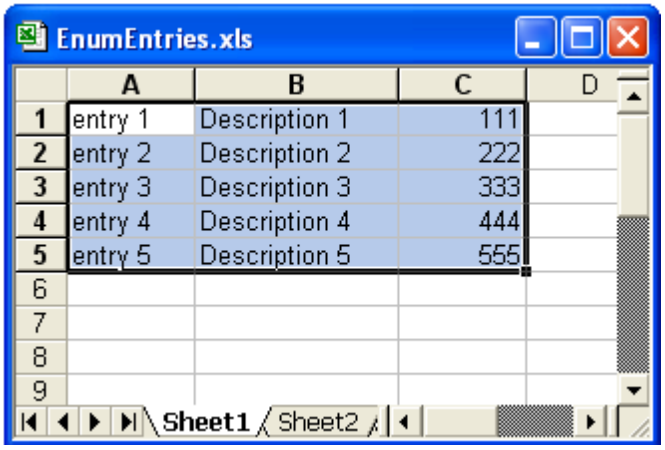
- If the map enumerated list constrains other enumerated lists or list entries in the authoring tool map schema, you can select the effected tool enumerated lists by selected the enumerated list name in the tree view, and then clicking ... beside the **Constrains** box. Constrains relationships define tool enumerated list hierarchies much like the contains relationships define enumerated list hierarchies in the SmartPlant schema.
- If the map enumerated list is constrained by another enumerated list, you can select the tool enumerated lists that constrains this one by selecting the enumerated list in the tree view, and then clicking ... beside the **Constrained by** box.
- To view properties for the map enumerated list or enumerated list entries, click the list or entry in the tree view.
- As you add enumerated list entries to the list, the software adds them to the **Contains** box for the map enumerated list definition.
- When you define enumerated list entries, the **ContainedIn** box is automatically populated with the map enumerated list name. However, you can change the map enumerated list for an enumerated list entry by clicking ... beside the **ContainedIn** box and selecting another map enumerated list. The enumerated entry is automatically moved to the selected map enumerated list.

Import Enumerated List Entries from a Spreadsheet

1. In Microsoft Excel, create a spreadsheet containing your enumerated list entries.

   TIP In an authoring tool map schema, you can also import enumerated list entries into a tool map enumerated list and units of measure into a tool map unit of measure list using this procedure.

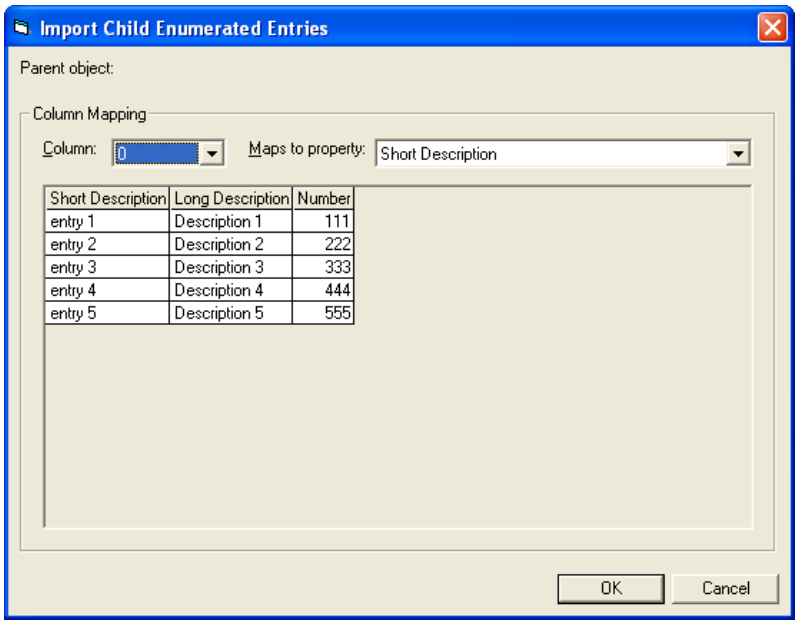
2. In Excel, select the rows containing your enumerated list entries, and click **Edit > Copy**.



3. In the Schema Editor, create the enumerated list type to which you want to add the list entries. For more information, see *Create an Enumerated List* (on page 173) or *Create a Map Enumerated List Definition* (on page 437).
4. In the **New Enumerated List** dialog box, click **Import Entries from Clipboard**.

TIP To import entries into a tool map enumerated list or map UoM list, click **Import Clipboard Entries** instead.

5. Verify that the default mapping between the columns in your copied spreadsheet and properties of enumerated list entries is correct. If this mapping is correct, you can skip to step 7 to import the entries.
6. If the default mapping is not correct, click the number for the column that you want to change the mapping for in the **Column** list. Numbering for the columns starts with zero, which is the first column in your spreadsheet.



7. In the **Maps to property** list, select the property that corresponds to the selected column in your copied spreadsheet.
8. When the mapping is correct, click **OK** to import the enumerated list entries.
9. Click **OK** in the **New Enumerated List** dialog box to save the new entries.

Define Mapping Between the Tool Map Schema and the SmartPlant Schema

After you define the objects that you want to map in the tool map schema and the SmartPlant schema, you can define the publish and retrieve mapping for those objects. Objects can have different mapping defined for publish and retrieve.

NOTE Some authoring tools may have additional steps for performing the mapping between the authoring tool and the SmartPlant schema. For mapping rules for each authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

Define Publish Mapping for Tool Properties

The following procedure is intended for authoring tools that do not deliver metadata adapters. To define publish mapping for tool properties, you must open the tool map class definition that exposes the properties that you want to map. If you have already opened your tool map schema and the map class definition that you want to define mapping for, you can skip to the *Define Map Properties for the Map Class Definition* section below and begin mapping.

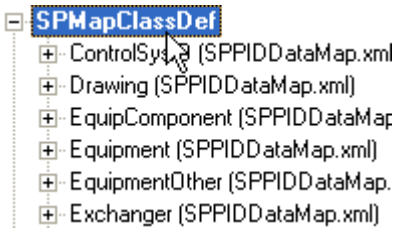
IMPORTANT

- While the Schema Editor provides flexibility for defining the types of mapping that are described in this section, the underlying authoring tool adapter ultimately defines whether the different types of mapping are supported. Therefore, you may be able to define mapping in the Schema Editor that is not supported by the authoring tool.
- Before you define any mapping other than straightforward one-to-one mapping that does not involve tool map edge definitions or SmartPlant schema edge definitions, check the limitations of the authoring tool adapter to determine what types of mapping are supported. Authoring tool programmers who are defining mapping for their authoring tools should be aware that they are responsible for supporting, with the assistance of the Schema Component API, any Schema Editor-defined mapping that they need.

Open the Tool Map Schema and Map Class Definition that You Want to Modify

1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, select one of the **Tree** views or **Editor** views and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
4. In the tree view, expand the **SPMapClassDef** node.



5. In the tree view, double-click the map class definition that exposes the properties that you want to map.

Define Interface Definitions and Class Definitions for Publish Mapping

1. In the **Edit Map Class Definition** dialog box, verify that all the properties associated with this map class definition are defined in the **Map properties** box.

   TIP If the mapping option is either **Include Generalized** or **Include Realized**, then the properties that can be mapped for this class definition will include all properties from the generalized or realized tool class definitions. These properties will not appear in the **Map properties** box but will be displayed when the mapping is defined.

2. If properties are missing, click ... beside the **Map properties** box, and select the authoring tool map properties for the class definition.

   TIP If the properties that you want to associate with the map class do not already exist in the tool map schema, you can click Create in the Possible MapProperties dialog box to create them. For more information, see *Create a Map Property Definition* (on page 433).

Define Interface Definitions and Class Definitions for Publish Mapping

1. In the **Edit Map Class Definition** dialog box verify that the SmartPlant schema classes definitions or interfaces definitions to which this tool class maps for publish are defined in the **Publish to SmartPlant Class/Interface** box.

   TIPS

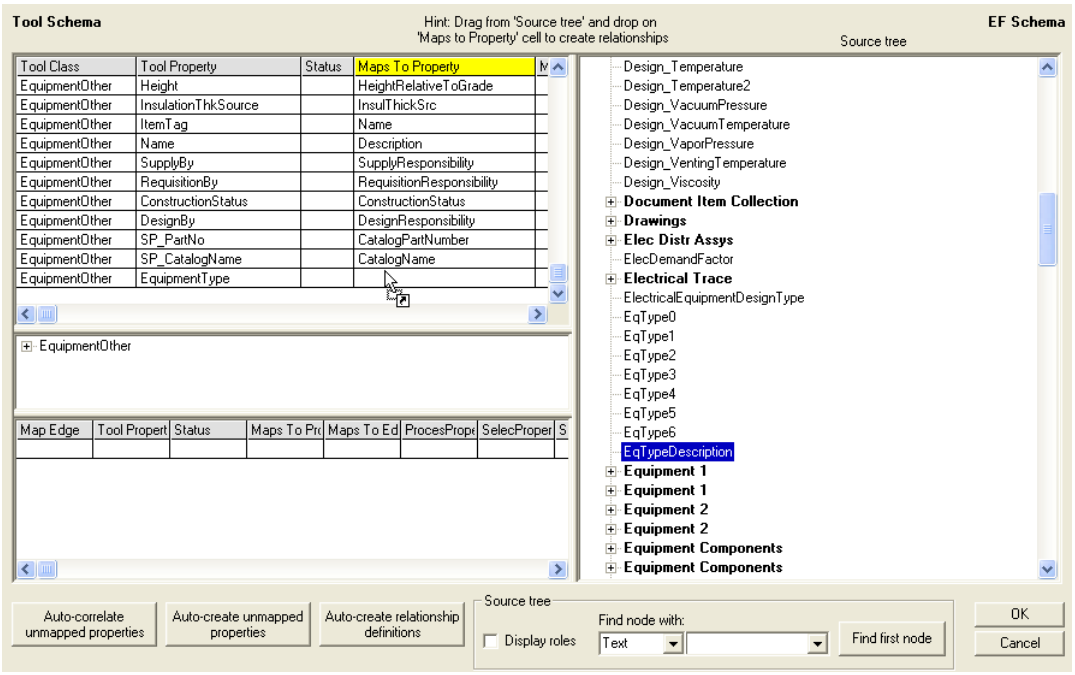
   - If you need to add or modify class definitions or interface definitions, click ... beside the **Publish to SmartPlant Class/Interface** box, and select the SmartPlant schema class definitions or interface definitions to which you want to map for publish operations.
   - Most mapping between map class definitions and the SmartPlant schema should be done using SmartPlant interface definitions instead of class definitions. The authoring tool adapter defines whether an application supports mapping to interface definitions instead of class definitions.
   - If you map to interface definitions, you should map to the primary interface definitions for the SmartPlant objects that the tool class definition corresponds to.
   - If the authoring tool does not publish the map class definition, you are not required to select a SmartPlant schema class definition or interface definition for mapping.

2. Under **Publish map**, click **Map Properties** to define property mapping for publish operations.
3. In the **Source** tree in the **Publish Map for Class Definition** window, expand the SmartPlant class definition or interface definition to which you want to map tool properties.

   TIP Nodes in bold text are edge definitions to other objects in the SmartPlant schema. Clicking + to expand the nodes displays the properties available across those edge definitions. If the underlying tool adapter does not support mapping across edge definitions (for example, one-to-many or many-to-many mapping), then any mapping defined using properties exposed by expanding these edge definitions will not actually be used and might cause the tool adapter to fail.

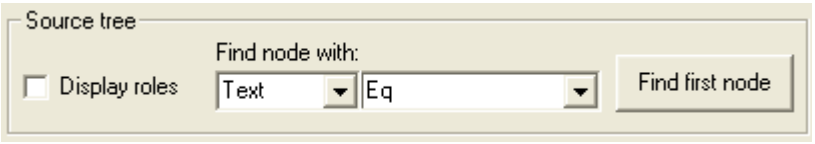
Define One-to-One Mapping for Published Properties

1. If you want to define one-to-one mapping between a property in the SmartPlant schema and a property in the tool, drag a property from the **Source** tree to the **Maps to Property** column for the tool property that you want to map.



TIPS

- You can map many properties from the SmartPlant schema to one property in the tool map schema by dragging and dropping more than one property onto the **Maps to Property** column for a single SmartPlant schema property. However, not all tool adapters support mapping one tool property to multiple SmartPlant properties.
- You can map multiple tool properties to one SmartPlant property by dragging and dropping one SmartPlant property multiple times onto the **Maps to Property** column for different tool properties. However, not all tool adapters support mapping one tool property to multiple SmartPlant properties.
- To find properties and edge definitions in the source tree, define search criteria under **Source tree** at the bottom of the window, and then click **Find first node**. You can click **Find first node** to go to the next node in the tree that matches the criteria you defined.

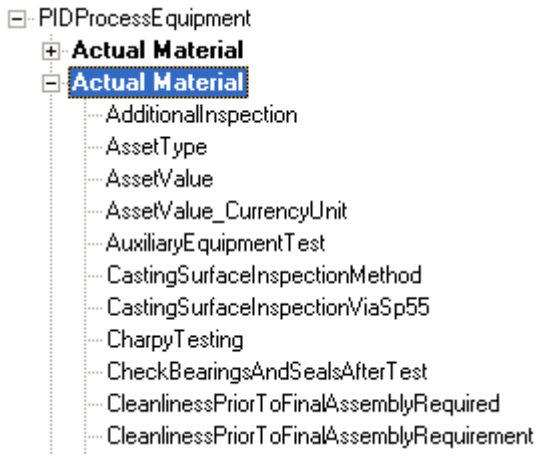


2. If you want to map a tool property to a relationship in the SmartPlant schema, drag an edge definition from the **Scope** tree to the **Maps to Property** column for the tool property that you want to map.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define One-to-Many Mapping for Published Properties

1. If you want to define one-to-many mapping between the SmartPlant schema and the tool, expand an edge definition in the **Source** tree.



TIPS

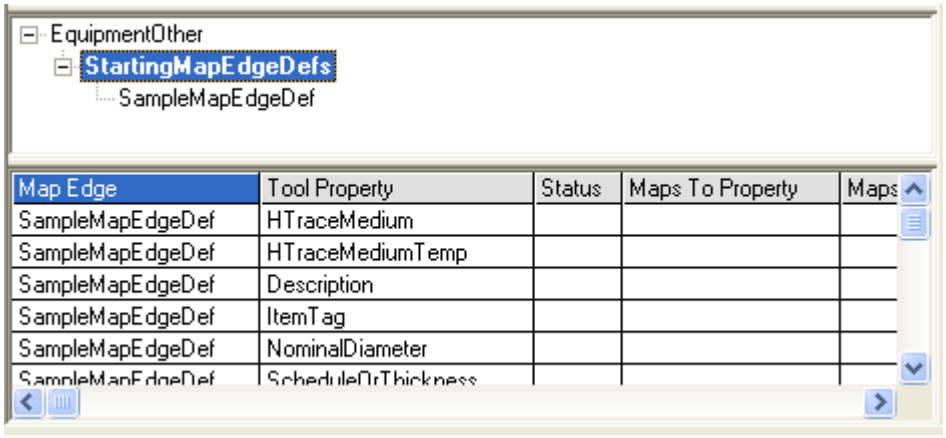
- Edge definitions in the **Source** tree are in bold text. When you expand an edge definition in the **Source** tree, the software displays the properties across that edge definition.
- In one-to-many mapping, one tool object corresponds to multiple SmartPlant objects.

2. Drag a property definition from beneath the expanded edge definition node in the **Source** tree to the **Maps to Property** column for the tool property that you want to map.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define Many-to-One Mapping for Published Properties

1. If you want define many-to-one mapping between the tool map schema and SmartPlant schema, expand the map class definition tree, and then expand the **StartingMapEdgeDefs** node to see the properties for the map edge definition in the table view at the bottom of the window.



TIP In many-to-one mapping, multiple tool objects correspond to one SmartPlant objects.

2. Drag a property from the **Source** tree to the **Maps to Property** column for the map edge definition property that you want to map.
3. If you want to define many- to-many mapping between the tool map schema and the SmartPlant schema, expand the map class definition tree, and then expand the **StartingMapEdgeDefs** node to see the properties for the map edge definition in the table view at the bottom of the window.

   TIP In many-to-many mapping, multiple tool objects correspond to multiple SmartPlant objects.
4. Drag a property definition from beneath an expanded edge definition node in the **Source** tree to the **Maps to Property** column for the map edge definition property that you want to map.

   TIP Edge definitions in the **Source** tree are in bold text. When you expand an edge definition in the **Source** tree, the software displays the properties across that edge definition.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define Mapping Automatically

1. Click the **Auto-correlate unmapped properties** to allow the Schema Editor to automatically map unmapped properties to properties in the SmartPlant schema.

   TIP The Schema Editor uses the automation software that you select to do this mapping. The automation software determines the rules used for mapping.
2. Click **Auto-create unmapped properties** to allow the Schema Editor to create SmartPlant schema properties that correspond to the unmapped tool properties and defines the mapping between the new SmartPlant schema properties and the tool properties.

   TIP The Schema Editor uses the automation software that you select to create these properties and do the mapping. The automation software determines the type information to create the properties and the rules used for mapping.
3. Click **Auto-create relationship definitions** to create relationship definitions in the SmartPlant schema based on your selections.

Save Publish Mapping for Properties

1. Click **OK** to save the publish mapping for the map class definition and return to the **Edit Map Class Definition** dialog box.
2. Click **OK** in the **Edit Map Class Definition** dialog box to save the updated map class definition.

NOTES

- Publish and retrieve mapping do not have to be the same for the map class definition or the map properties definitions.
- Not all properties are mapped for both publish and retrieve.

See Also

Create a Map Class Definition (on page 430)
Create a Map Edge Definition (on page 435)
Create a Map Enumerated List Definition (on page 437)
Create a Map Property Definition (on page 433)
Create a Map Unit of Measure List Definition (on page 457)
Add Objects to a Tool Map Schema (on page 429)
Define Mapping for a Tool Enumerated List (on page 452)
Define Retrieve Mapping for Tool Properties (on page 447)
Define Mapping for a Tool Unit of Measure List (on page 454)

Define Retrieve Mapping for Tool Properties

The following procedure is intended for authoring tools that do not deliver metadata adapters. To define publish mapping for tool properties, you must open the tool map class definition that exposes the properties that you want to map. If you have already opened your tool map schema and the map class definition that you want to define mapping for, you can skip to the Define Map Properties for the Map Class Definition section below and begin mapping.

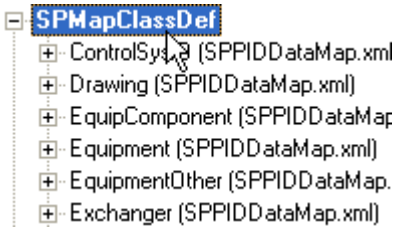
IMPORTANT

- While the Schema Editor provides flexibility for defining the types of mapping that are described in this section, the underlying authoring tool adapter ultimately defines whether the different types of mapping are supported. Therefore, you may be able to define mapping in the Schema Editor that is not supported by the authoring tool.
- Before you define any mapping other than straightforward one-to-one mapping that does not involve tool map edge definitions or SmartPlant schema edge definitions, check the limitations of the authoring tool adapter to determine what types of mapping are supported. Authoring tool programmers who are defining mapping for their authoring tools should be aware that they are responsible for supporting, with the assistance of the Schema Component API, any Schema Editor-defined mapping that they need.

Open the Tool Map Schema and Map Class Definition that You Want to Modify

1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, select one of the **Tree** views or **Editor** views and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
4. In the tree view, expand the **SPMapClassDef** node.



5. In the tree view, double-click the map class definition that exposes the properties that you want to map.

Define Map Properties for the Map Class Definition

1. In the **Edit Map Class Definition** dialog box, verify that all the properties associated with this map class definition are defined in the **Map properties** box.

   TIP If the mapping option is either **Include Generalized** or **Include Realized**, then the properties that can be mapped for this class definition will include all properties from the generalized or realized tool class definitions. These properties will not appear in the **Map properties** box but will be displayed when the mapping is defined.

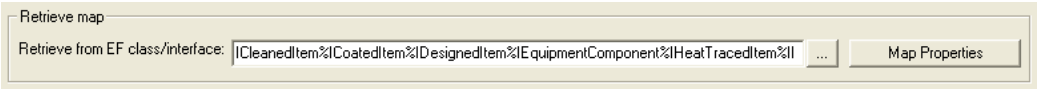
2. If properties are missing, click ... beside the **Map properties** box, and select the authoring tool map properties for the class definition.

   TIP If the properties that you want to associate with the map class do not already exist in the tool map schema, you can click **Create** in the **Possible MapProperties** dialog box to create them.

   Create a Map Property Definition (on page 433)

Define Interface Definitions and Class Definitions for Retrieve Mapping

1. In the **Edit Map Class Definition** dialog box, verify that the SmartPlant schema class definitions or interface definitions that map to the map class definition for retrieve operations are defined in the **Retrieve to SmartPlant Class/Interface** box.

   TIPS

   - If you need to add or modify class definitions or interface definitions, click ... beside the **Retrieve to SmartPlant Class/Interface** box, and select the SmartPlant schema class definitions or interface definitions that map to the map class definition for retrieve operations.
   - For retrieval, mapping should be defined using interface definitions rather than class definitions.

2. Under **Retrieve map**, click **Map Properties** to define property mapping for retrieve operations.

3. In the **Tool Schema** tree in the **Map for Retrieving** window, expand the tool map schema map class definition to which you want to map SmartPlant schema properties for retrieve.

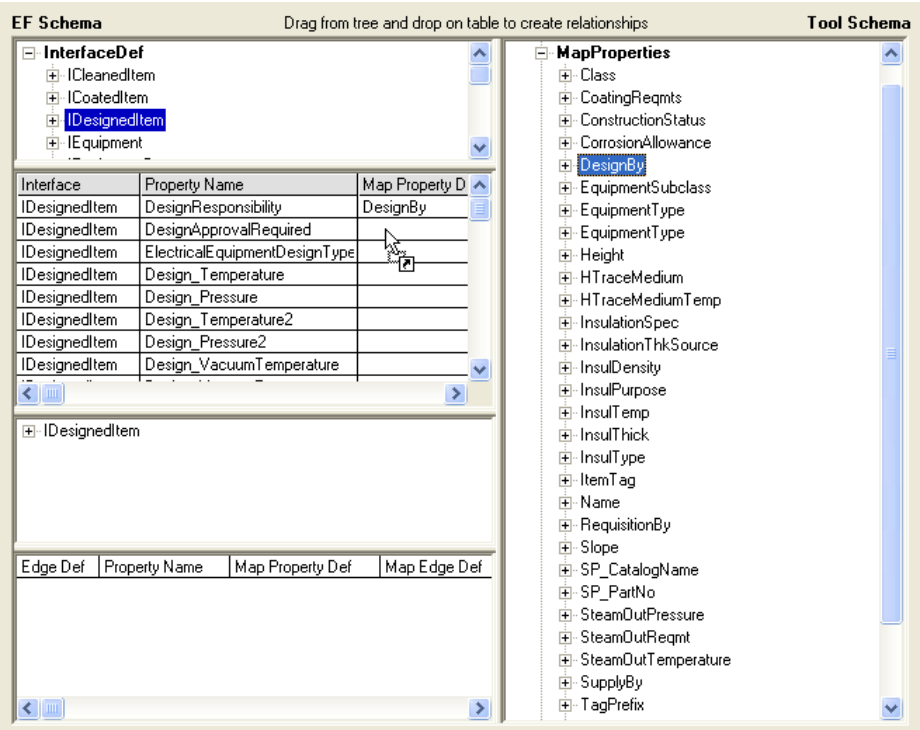
   

4. In the **SmartPlant Schema** tree, expand the **InterfaceDef** node.

   TIP The interface definitions that you see in the tree view are the ones you selected in the **Retrieve to SmartPlant Class/Interface** box in the **Edit Map Class Definition** dialog box. If an interface that you need does not appear in the tree, click **Cancel** and add it in the **Edit Map Class Definition** dialog box.

Define One-to-One Mapping for Retrieved Properties

1. If you want to do one- to-one mapping between a property in the SmartPlant schema and a property in the tool for retrieve, do the following:
   - In the top **SmartPlant Schema** tree, select the interface definition that exposes the property that you want to map for retrieve.
   - In the **Tool Schema** tree, expand the **MapProperties** node.
   - Drag a property from the **Tool Schema** tree to the **Map Property Def** column in the top table for the SmartPlant schema property that you want to map.



2. If you want to map to a tool property from a relationship in the SmartPlant schema, do the following:
   - In the top **SmartPlant Schema** tree, select the interface definition that exposes the property that you want to map for retrieve.
   - In the **Tool Schema** tree, expand the **StartingMapEdgeDefs** node.

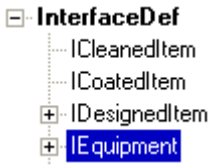


   - Drag an edge definition from the **Tool Schema** tree to the **Map Property Def** column in the top table for the SmartPlant schema property that you want to map.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define One-to-Many Mapping for Retrieved Properties

1. If you want to define one-to-many mapping between the SmartPlant schema and tool map schema, select the interface definition that is the starting point for the edge definition that you want to map for retrieve in the top **SmartPlant Schema** tree.



   TIP In one-to-many mapping, one SmartPlant object corresponds to multiple tool objects.

2. In the **Tool Schema** tree, expand the **StartingMapEdgeDefs** node.
3. Expand the map edge definition in the **Tool Schema** tree to display the tool properties across that map definition.
4. Drag a tool property exposed by a map edge definition to the **Map Property Def** column in the top table for the SmartPlant schema property that you want to map.

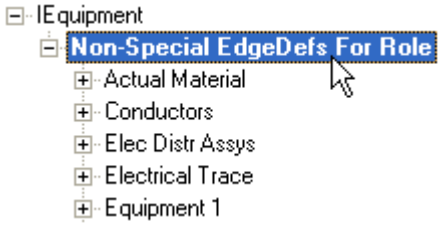
NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define Many-to-One Mapping for Retrieved Properties

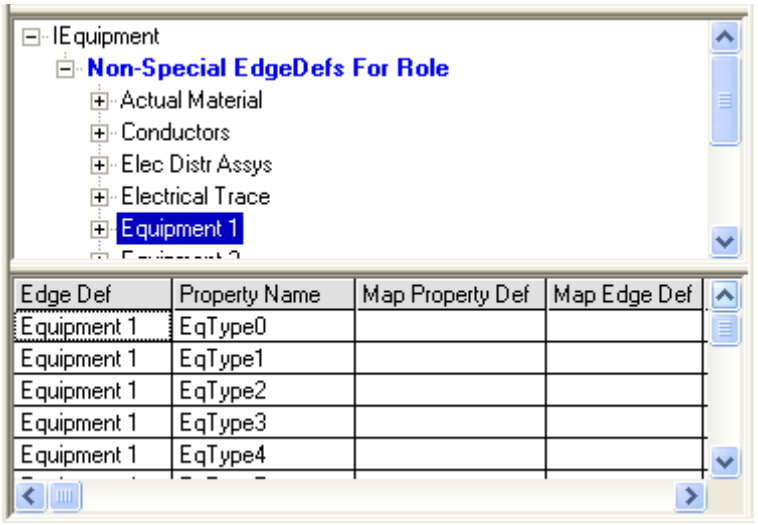
1. If you want to define many-to-one mapping between the SmartPlant schema and tool map schema, select the interface definition that is the starting point for the edge definition that you want to map for retrieve in the top **SmartPlant Schema** tree.

   TIP In many-to-one mapping, multiple SmartPlant objects correspond to one tool object.

2. In the bottom **SmartPlant Schema** tree, expand the interface definition that you selected in the top tree view.
3. Expand the **Non-Special EdgeDefs for Role** node.



4. Select the SmartPlant schema edge definition that you want to map in the tree view to display properties in the table below the tree view.



5. In the **Tool Schema** tree, expand the **MapProperties** node.
6. Drag a property from the **Tool Schema** tree to the **Map Property Def** column in the bottom table for the SmartPlant schema property that you want to map.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Define Many-to-Many Mapping for Retrieved Properties

1. If you want to define many-to- many mapping between the SmartPlant schema, and the tool map schema, select the interface definition that is the starting point for the edge definition that you want to map for retrieve in the top **SmartPlant Schema** tree.

   TIP In many-to-many mapping, multiple SmartPlant objects correspond to multiple tool objects.
2. In the bottom tree, expand the node for the interface definition you selected in the top tree view.
3. Expand the **Non-Special EdgeDefs For Role** node.
4. Select the SmartPlant schema edge definition that you want to map in the tree view to display properties in the table below the tree view.
5. In the **Tool Schema** tree, expand the **StartingMapEdgeDefs** node.
6. Expand the selected map edge definition in the **Tool Schema** tree to display the property across that map edge definition.
7. Drag a property definition from the expanded edge definition in the **Tool Schema** tree to the **Map Property Def** column in the top table for the SmartPlant schema property that you want to map.

NOTE Not all tool adapters support this special type of mapping. See the mapping rules for your authoring tool for more information.

Save Retrieve Mapping for Properties

1. Click **OK** to save the retrieve mapping for the map class definition and return to the **Edit Map Class Definition** dialog box.
2. Click **OK** in the **Edit Map Class Definition** dialog box to save the updated map class definition.

NOTES

- Publish and retrieve mapping do not have to be the same for the map class definition or the map properties definitions.
- Not all properties are mapped for both publish and retrieve.

See Also

Create a Map Class Definition (on page 430)
Create a Map Edge Definition (on page 435)
Create a Map Enumerated List Definition (on page 437)
Create a Map Property Definition (on page 433)
Create a Map Unit of Measure List Definition (on page 457)
Add Objects to a Tool Map Schema (on page 429)
Define Mapping for a Tool Enumerated List (on page 452)
Define Publish Mapping for Tool Properties (on page 442)
Define Mapping for a Tool Unit of Measure List (on page 454)

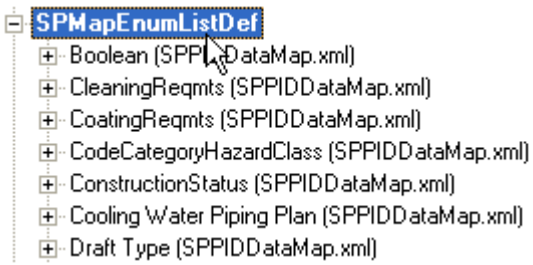
Define Mapping for a Tool Enumerated List

The following procedure is intended for authoring tools that do not deliver metadata adapters. If you have already opened your tool map schema and the map enumerated list definition that you want to define mapping for, you can skip to the Define Mapping for the Map Enumerated List section below and begin mapping.

Open the Tool Map Schema and Map Enumerated List Definition that You Want to Modify

1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, select one of the **Tree** views or **Editor** views and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.
4. In the tree view, expand the **SPMapEnumListDef** node.



5. In the tree view, double-click the map enumerated list definition that you want to define mapping for.

Define Mapping for the Map Enumerated List

1. In the **Edit Map Enumerated List Definition** dialog box, verify that the SmartPlant schema enumerated list type that this tool enumerated list maps to for publish is correct in the **Maps to** box.

   TIP Not all enumerated lists require both publish and retrieve mapping; therefore, the **Maps to** or **Maps from** options may be blank for some enumerated lists.

2. If the SmartPlant schema enumerated list type for publish is incorrect, click ... beside the **Maps to** box, and select the SmartPlant enumerated list type.

3. In the **Maps from** box, verify that the SmartPlant schema enumerated list type that maps to this tool enumerated list for retrieve is correct.

4. If the SmartPlant schema enumerated list type for retrieve is incorrect, click ... beside the **Mapped from** box, and select the SmartPlant enumerated list type.

Define Publish and Retrieve Mapping for the Map Enumerated List Entries

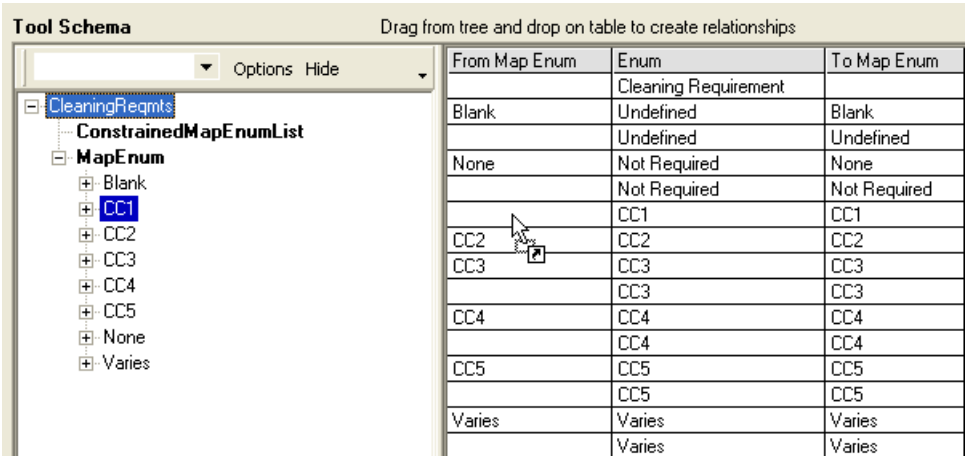
1. To define publish and retrieve mapping for the map enumerated list entries, click **Publish and Retrieve Maps** in the **Edit Map Enumerated List Definition** dialog box.

   TIP You are not required to define both publish and retrieve mapping for map enumerated list entries. If you want to define publish mapping only, click **Publish Map**. If you want to define retrieve mapping only, click **Retrieve Map**.

2. In the **Publish and Retrieve Maps for Tool Enumerated List** window, expand the map enumerated list in the **Tool Schema** tree.

3. Expand the **MapEnum** node to see the entries in the enumerated list.

4. To define the retrieve mapping for an enumerated list entry, drag the entry from the **Tool Schema** tree to the **From Map Enum** column for the SmartPlant schema enumerated entry that maps to the tool entry.



TIPS

- The table contains all entries in the SmartPlant schema enumerated list types that you are defined as mapping to or mapped from the tool enumerated list.
- Modifications that you make to the mapping have asterisks (*) around them, for example, *CC1*.

5. To define the publish mapping for an enumerated list entry, drag the entry from the **Tool Schema** tree to the T**o Map Enum** column for the SmartPlant schema enumerated entry that the tool entry maps to.

   TIP Publish and retrieve mapping for an enumerated list entry do not have to be the same.

6. Repeat steps 4 and 5 until all the entries that you want to map for publish and retrieve are mapped.
7. Click **X** in the title bar of the **Publish and Retrieve Maps for Tool Enumerated List** window to close the window and save your changes.
8. Click **OK** in the **Edit Map Class Definition** dialog box to save your mapping.

NOTES

- Publish and retrieve mapping does not have to be the same for map enumerated lists or their list entries.
- Not all enumerated list entries are mapped for both publish and retrieve.

See Also

Create a Map Class Definition (on page 430)
Create a Map Edge Definition (on page 435)
Create a Map Enumerated List Definition (on page 437)
Create a Map Property Definition (on page 433)
Create a Map Unit of Measure List Definition (on page 457)
Add Objects to a Tool Map Schema (on page 429)
Define Publish Mapping for Tool Properties (on page 442)
Define Retrieve Mapping for Tool Properties (on page 447)
Define Mapping for a Tool Unit of Measure List (on page 454)

Define Mapping for a Tool Unit of Measure List

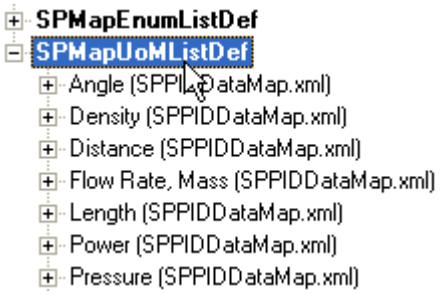
The following procedure is intended for authoring tools that do not deliver metadata adapters. If you have already opened your tool map schema and the map unit of measure (UoM) list definition that you want to define mapping for, you can skip to the Define Mapping for the Map Unit of Measure List section below and begin mapping.

Open the Tool Map Schema and Map UoM List Definition that You Want to Modify

1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, select one of the **Tree** views or **Editor** views and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Select the view that you want to use in the **Form** list instead.

4. In the tree view, expand the **SPMapUoMListDef** node.



5. In the tree view, double-click the map UoM list definition that you want to define mapping for.

Define Mapping for the Map Unit of Measure List

1. In the **Edit Map Unit of Measure List Definition** dialog box, verify that the SmartPlant schema enumerated list type that this tool UoM list maps to for publish is correct in the **Maps to** box.

   TIP Not all UoM lists require both publish and retrieve mapping; therefore, the **Maps to** or **Maps from** options may be blank for some UoM lists.

2. If the SmartPlant schema enumerated list type for publish is incorrect, click ... beside the **Maps to** box, and select the SmartPlant enumerated list type.
3. In the **Maps from** box, verify that the SmartPlant schema enumerated list type that maps to this tool UoM list maps for retrieve is correct
4. If the SmartPlant schema enumerated list type for retrieve is incorrect, click ... beside the **Mapped from** box, and select the SmartPlant enumerated list type.

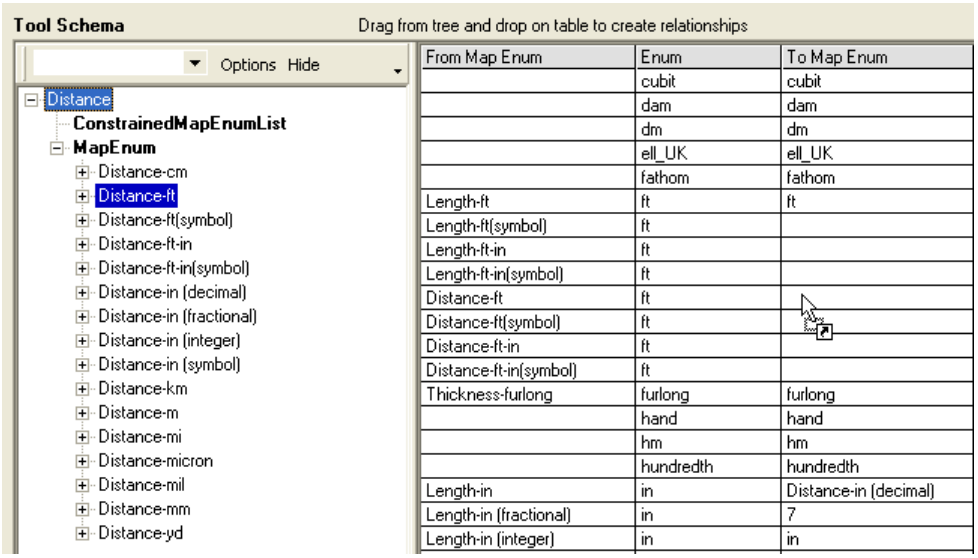
Define Publish and Retrieve Mapping for the Map Units of Measure

1. To define publish and retrieve mapping for the map UoMs, click **Publish and Retrieve Maps** in the **Edit Map Unit of Measure List Definition** dialog box.

   TIP You are not required to define both publish and retrieve mapping for map UoMs. If you want to define publish mapping only, click **Publish Map**. If you want to define retrieve mapping only, click **Retrieve Map**.

2. In the **Publish and Retrieve Maps for Tool Enumerated List** window, expand the map UoM list in the **Tool Schema** tree.
3. Expand the **MapEnum** node to see the UoMs in the UoM list.

4. To define the retrieve mapping for a UoM, drag the UoM from the **Tool Schema** tree to the **From Map Enum** column for the SmartPlant schema UoM that maps to the tool UoM.



From Map Enum	Enum	To Map Enum
	cubit	cubit
	dam	dam
	dm	dm
	ell_UK	ell_UK
	fathom	fathom
Length-ft	ft	ft
Length-ft(symbol)	ft	
Length-ft-in	ft	
Length-ft-in(symbol)	ft	
Distance-ft	ft	
Distance-ft(symbol)	ft	
Distance-ft-in	ft	
Distance-ft-in(symbol)	ft	
Thickness-furlong	furlong	furlong
	hand	hand
	hm	hm
	hundredth	hundredth
Length-in	in	Distance-in (decimal)
Length-in (fractional)	in	7
Length-in (integer)	in	in

TIPS

- The table contains all entries in the SmartPlant schema enumerated list types that you are defined as mapping to or mapped from the tool UoM list.
- Modifications that you make to the mapping have asterisks (*) around them, for example, *CC1*.

5. To define the publish mapping for a UoM, drag the entry from the **Tool Schema** tree to the **To Map Enum** column for the SmartPlant schema UoM that the tool UoM maps to.

   TIP Publish and retrieve mapping for a UoM do not have to be the same.

6. Repeat steps 4 and 5 until all the entries that you want to map for publish and retrieve are mapped.
7. Click **X** in the title bar of the **Publish and Retrieve Maps for Tool Enumerated List** window to close the window and save your changes.
8. Click **OK** in the **Edit Map Class Definition** dialog box to save your mapping.

NOTES

- Publish and retrieve mapping do not have to be the same for map UoM lists or the UoMs the list contains.
- Not all UoMs are mapped for both publish and retrieve.

See Also

Create a Map Class Definition (on page 430)
Create a Map Edge Definition (on page 435)
Create a Map Enumerated List Definition (on page 437)
Create a Map Property Definition (on page 433)
Create a Map Unit of Measure List Definition (on page 457)
Add Objects to a Tool Map Schema (on page 429)
Define Mapping for a Tool Enumerated List (on page 452)

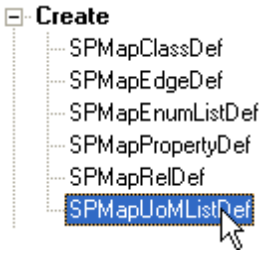
Define Publish Mapping for Tool Properties (on page 442)
Define Retrieve Mapping for Tool Properties (on page 447)

Create a Map Unit of Measure List Definition

When you create a map unit of measure (UoM) list definition, you must create the UoM list definition and then define the UoM list entries that the list contains. In the tool map schema, a UoM list is exactly like an enumerated list and can be mapped to an enumerated list in the SmartPlant schema.

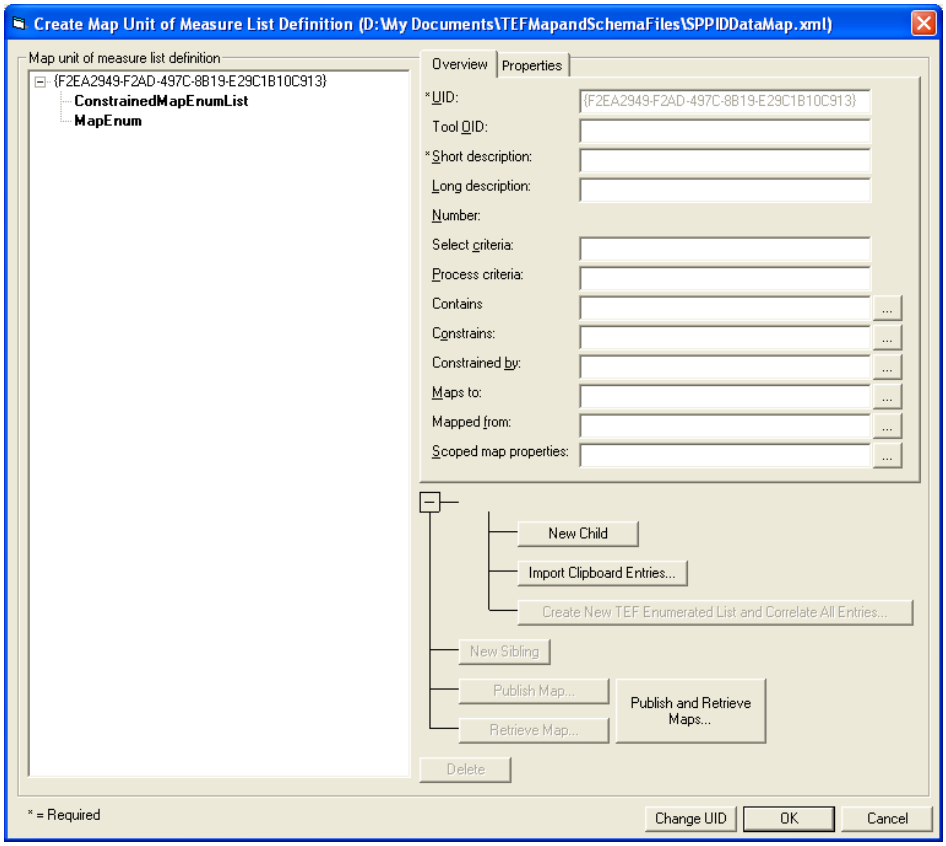
1. After you open the tool map schema that you want to edit in the Schema Editor, click **View > Tool Schema**.
2. Click **Options** on the **View Schema** dialog box.
3. On the **View** tab, click **Editor**, and then click **OK**.

   TIP If you do not see the **View** tab, the **View** dialog box is in Simple mode. Click **Editor** in the **Form** list instead.
4. In the **Tool Schema Editor UML** window, double-click **SPMapUoMListDef** in the **Create** tree.



TIP You can also create map unit of measure list definitions by double-clicking the **SPMapUoMListDef** node in any tree view in any Schema Editor view.

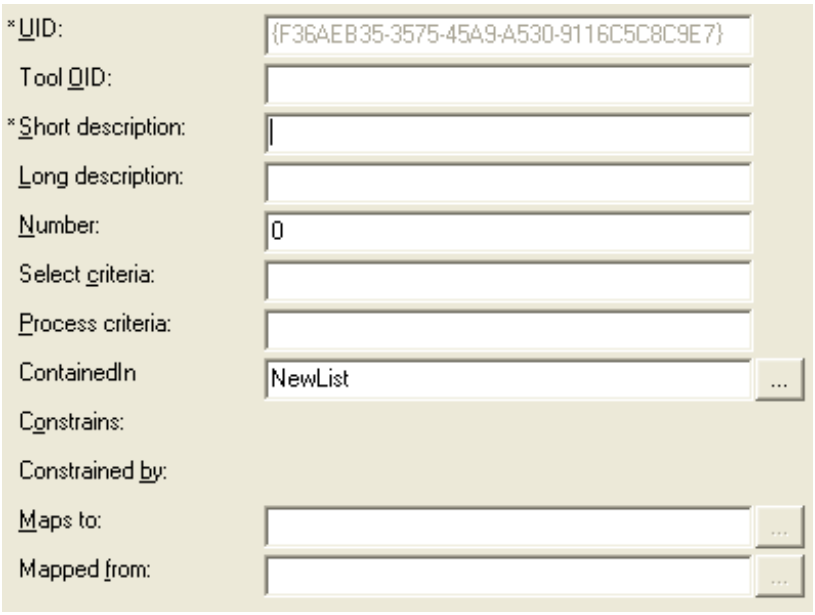
5. In the **Short description** box, type the name of the UoM list.



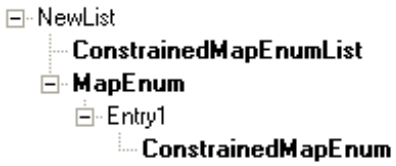
6. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool UoM list definition and its entries. See the mapping rules for your authoring tool for how to set the value for this property.
7. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool UoM list definition and its enumerations. See the mapping rules for your authoring tool for how to set the value for this property.
8. To define the SmartPlant UoM list that the tool UoM list definition maps to for publish, click … beside the **Maps to** box, and select the SmartPlant UoM list.
9. To define the SmartPlant UoM list that maps to the tool map UoM list definition for retrieve, click … beside the **Mapped from** box, and select the SmartPlant UoM list.
10. To define the tool map properties that this list defines UoMs for, click … beside the **Scoped map properties** box, and select the map property definitions that are scoped by this map UoM list.
11. To create a UoM list entry (MapEnum) for this map UoM list, click **New Child** in the **New Map Unit of Measure List Definition** dialog box.

    TIP If you have a spreadsheet containing your list entries, you can copy them to the Clipboard and then add them to the list using the **Import Clipboard Entries** button. For more information, see *Import Enumerated List Entries from a Spreadsheet* (on page 174).

12. In the **Short description** box, type the name of the list entry.



TIP When you move out of the **Short description** box, the new list entry is automatically added to the **Map unit of measure list definition** tree.



13. If a secondary numeric key identifies the UoM list entry, type the key in the **Number** box.
14. In the **Select criteria** box, define the criteria that the tool adapter uses to select or identify objects for this tool UoM list entry. See the mapping rules for your authoring tool for how to set the value for this property.
15. In the **Process criteria** box, define the criteria that the adapter uses to process objects that are instances of this tool UoM list entry. See the mapping rules for your authoring tool for how to set the value for this property.
16. To create additional list entries, click **New Sibling**, and repeat steps 12-15 for each entry that you want to add.
17. To save your changes, click **OK**.

IMPORTANT For some authoring tools, you must change the system-generated UID so that the adapter can correlate the enumerated list and its entries with the appropriate enumerated list and entries in the tool database. For example, for SmartPlant P&ID, you must replace the system- generated UID for the map enumerated list and the list entries with UIDs from the SmartPlant P&ID database. For more information, see the rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation. To change the system-generated UID, click **Change UID**.

NOTES

- To view properties for the map UoM list or UoM list entries, click the list or entry in the tree view.
- As you add UoM list entries to the list, the software adds them to the **Contains** box for the map UoM list definition.
- When you define UoM list entries, the **ContainedIn** box is automatically populated with the map UoM list name. However, you can change the map UoM list for a UoM list entry by clicking ... beside the **ContainedIn** box and selecting another map UoM list. The UoM entry is automatically moved to the selected map UoM list.

Create Temporary Smart 3D Enumeration Mapping Relationships

Smart 3D enumeration mapping is implicit, which means the Smart 3D enumerations are assigned an enumeration number for the corresponding entry in the SmartPlant schema. The Schema Editor does not display this mapping information by default. In order to view the Smart 3D enumeration mapping in the Schema Editor, you must create explicit and temporary relationships from these implicit relationships.

You can create the temporary relationships with the **Add container with implicit enumeration relationships** command. This command is located in the **Tools > Tool Schema File** menu of the Schema Editor.

IMPORTANT The mapping relationships created when using this command are not saved after you exit the Schema Editor. You must run this command each time you need to view the Smart 3D enumeration mapping relationships.

1. In the Schema Editor, click **Workflows > Overall Workflows**.
2. In the **Tool Schema** tab, click **Open** above the **Schema File** menu, and select a SmartPlant schema document.
3. In the **Tool Schema** tab, click **Open** above the **Tool Schema File** menu, and select a Smart 3D map schema file.
4. Click **Tools > Tool Schema File > Add container with implicit enumeration relationships** to create the temporary enumeration relationships between the SmartPlant schema and the Smart 3D map schema file.
5. Click **Tools > Tool Schema File > Integrated Map Environment** to view and modify the SmartPlant schema and Smart 3D map schema file relationships.

   NOTE You can view the mapping relationships in any of the Schema Editor mapping views.

Remove incomplete schema relationships

Every relationship must point to two objects: one at end 1 of the relationship, and another at end 2 of the relationship. If one or both of these objects are missing, the relationship is referred to as a dangling relationship. In most cases, a dangling relationship should not exist. If the software finds any dangling relationships, they should be deleted. You can view and delete dangling relationships with the **Process Dangling Relationship** command, which is located in the **Tools > Schema** and **Tools > Tool Schema File** menus.

In the following example, the SmartPlant schema is modified, but you can use the same process to modify a tool map schema.

1. Click **Find > Integration > Schema Documents**.

   IMPORTANT You must be logged on to the SmartPlant Foundation data warehouse database and have a schema administrator role to access commands on the **Integration** menu.

2. Type **CMF*** in the **Find** dialog box, and click **OK**.
3. Right-click the CMF file, and then click **Edit > Check Out**.
4. Click **OK** in the **Check Out** dialog box.
5. In the **New Items** window, right-click the new version of the CMF file, and select **Launch Schema Editor**.
6. On the **Set Active Configurations** dialog box, select the schema version that you want to view.
7. Click **OK** to set the configuration.

   TIPS

   - For more options, click **Advanced** on the **Set Active Configuration** dialog box.
   - By default, changes to the SmartPlant Schema are written to all versions managed by the CMF file. You can choose to write changes to only selected versions; however, that is not recommended.
   - If you choose to view one version of the schema (**Specify configurations to display** window), but write changes to all versions (**Set Configurations for new objects** window), an information window appears when you click **OK**, indicating that you will be prompted with every change to the schema to confirm that you want to write the changes to all versions. Click **OK** to continue.

   NOTE The CMF file must be checked in to SmartPlant Foundation before users can publish from an authoring tool.

8. Click **Tools > Tool Schema File > Process Dangling Relationships** to view and remove dangling relationships in the tool map schema file.

   NOTE Click **Tools > Schema > Process Dangling Relationships** to view and remove dangling relationships in the SmartPlant schema file.

9. In the **Process Dangling Relationships** dialog box, select the relationships to remove, and click **Delete**.
10. Click **OK** to delete the relationships.

Distribute the Modified Tool Map Schema and SmartPlant Schema

When the mapping is complete, you must distribute the updated authoring tool map schema to every computer that requires it. For more information about where the tool map schema is delivered for your authoring tool, see rules for your authoring tool in the *SmartPlant Mapping User's Guide* or in the tool documentation.

If modifications to the SmartPlant schema were required to define the mapping, you must also distribute the updated SmartPlant schema to every SmartPlant Foundation/integration server.

TIP By default, the SmartPlant schema (EFSchema.xml) is delivered with the SmartPlant Server component and is located in the ..\Program Files\CommonFiles\Intergraph\EFSchema folder. However, you may have a different version of the SmartPlant schema for each SmartPlant Foundation site.

Load the Updated Schema into SmartPlant Foundation

Loading the updated schema into SmartPlant Foundation involves the following high-level tasks:

- Checking the CMF file back into the Desktop Client.
- Running the **CMF File Manager** on the CMF file to create an XML file for loading.
- Importing the schema using the **Schema Import Wizard**.
- Regenerating the component schemas.

SECTION 13

Comparison

In the Schema Editor, you can compare schemas, tool map schemas, and data files with other files of the same type. Comparisons are valuable for seeing changes from one version of the schema to another, for seeing changes in data files as they are generated by the integrated authoring tools, and for seeing differences between the schemas from different authoring tools.

There are five types of comparisons in the Schema Editor:

- Full Comparison
- Selective Update
- Detection of Tombstones
- Comparison Using Views
- Integrated Reference File Environment

Full Comparisons

A full comparison compares every object, relationship, and property in one file with the file you select. When you do a full comparison, you can choose whether the software displays the full results of the comparison, the comparison instructions only, or both the comparison results and instructions.

After you perform a full comparison between files, you can view the modification instructions generated by the software. These modification instructions define the changes required to go from the objects in the file you selected for comparison to the objects in the original file. For example, you can see which objects would be deleted, which ones would be updated, and which ones would be created.

Selective Update

Selective update allows you to choose which objects to add, update, or delete after you have compared schemas, tool map schemas, and data files with other files of the same type. Because any addition, update, or deletion you make changes the file you are comparing, you need to be cautious about the selections you make and aware of their impact.

Detection of Tombstones

An alternative to doing a full comparison between two files is to detect tombstones. Tombstones are objects that have been deleted in one file that exist in another file. The Schema Editor identifies items as tombstones when they do not exist in the original file you opened in the Schema Editor but do exist in the second file you selected for comparison. After you detect tombstones, you can view the tombstones that were found by reviewing the comparison instructions for the two files.

Because tombstone detection does not involve a full comparison of two files, it typically runs faster than a full comparison.

Comparison Using Views

Comparison using views allows you to compare data based on a more user-centered view. Comparisons using views are most useful for comparing data files because the data is compared in terms of what a user might see in an authoring tool instead of a comparison based on the schema.

Comparison views are delivered as part of the schema, but you can create your own.

Integrated Reference File Environment

The **Integrated Reference File Environment** in the Schema Editor lets you view and modify data in a set of schema and tool schema reference files. You can copy or replace data from a reference schema file into an active schema file, and copy or replace data from a reference tool schema file into an open tool schema file.

With this environment, you can copy data from schema and tool map files from a previous project into the schema and tool map files of a new project. This functionality allows you to deploy projects in less time by reusing configurations from other projects. For example, enumerated lists can be copied into a new project without requiring them to be manually re-entered.

The software indicates whether an object only exists in the source file, only exists in the reference file, or if it exists in both. Objects that are included in both the source file and reference file are compared. If there is a difference between these files, you can update, create, or delete the source object based on the related reference object.

For more information, see *Modify Reference Files with the Integrated Reference File Environment* (on page 408).

See Also

Full Comparison: An Example (on page 465)
Detect Tombstones (on page 468)
Tombstone Comparison Results: An Example (on page 469)
Selectively Update (on page 471)
Modify Reference Files with the Integrated Reference File Environment (on page 408)

Compare Files

1. Open the schema, tool map schema, or data file to which you want to compare another file.
2. Click **File > Compare with File**, and then point to the type of file you want to compare with the currently open file.
3. To select the type of comparison that you want, do one of the following:
   - To do a full comparison and view comparison results only, click **Full Comparison > Display Results**.
   - To do a full comparison and view comparison instructions only, click **Full Comparison > Display Instructions**.
   - To do a full comparison and view both comparison results and instructions, click **Full Comparison > Display Both**.
   - To compare using a view definition, click **Using Views**.

- To compare and choose the objects to add, update, or modify, click **Selectively / Update.**

4. Browse to the file to which you want to compare the schema, data file, or tool map schema.
5. If the **View File Type Instructions** dialog box appears, click the view in which you want to display the comparison results, and then click **OK**.

   TIP If you do a full comparison and only want to view comparison results or if you do a comparison using views, you are not required to select the view in which you want to display results.

NOTES

- After you do a full comparison, you can review the comparison instructions that would be generated by the software if one file superseded the other. You can review these comparison instructions using the appropriate **Instructions** command on the **View** menu. For example, to view schema instructions, click **View > Schema Instructions > Tree/Table**.
- You can save instructions by clicking **File > Save > [FileType] Instructions**, where [FileType] is the type of file for which you want to save instructions. For example, to save data instructions, click **Data Instructions** on the **Save** submenu.
- If you selectively update objects, any addition, update, or deletion you make changes the file you are comparing. You need to be cautious about the selections you make and aware of their impact.

See Also

Full Comparison: An Example (on page 465)
Detect Tombstones (on page 468)
Comparison (on page 463)

Full Comparison: An Example

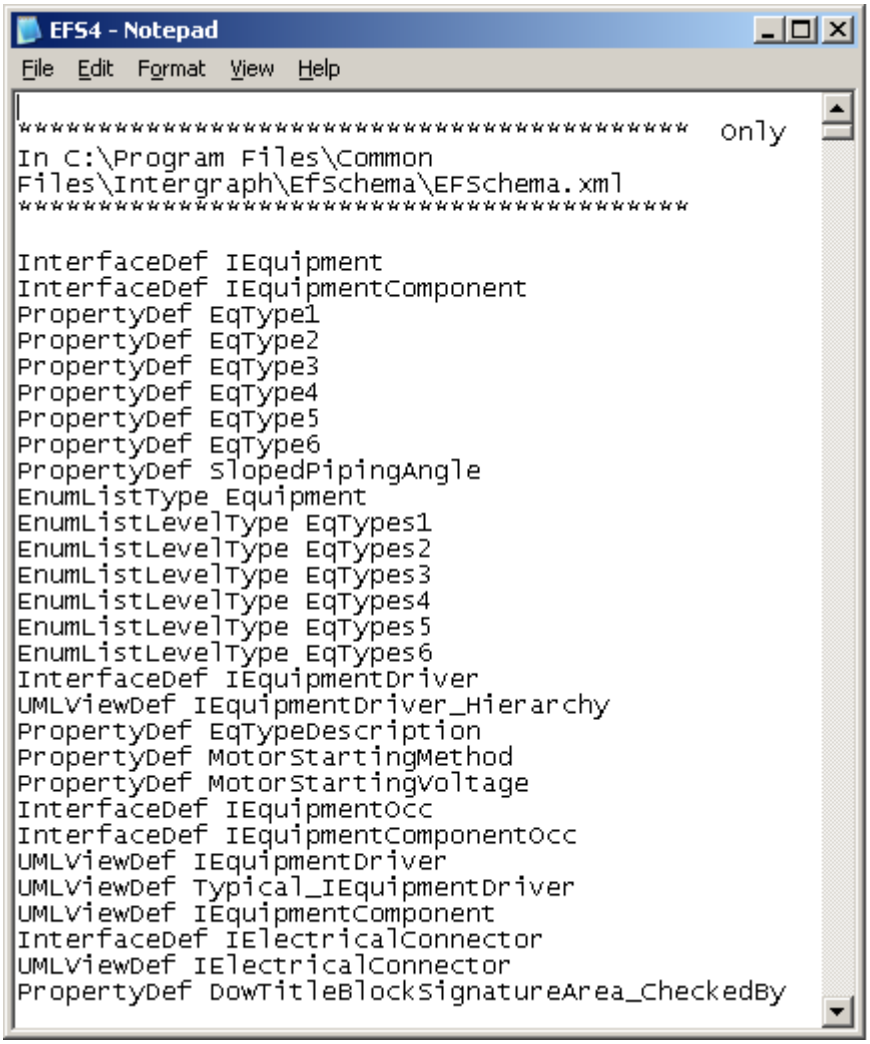
In the Schema Editor, you can compare schemas, tool map schemas, and data files. When you perform a full comparison, the Schema Editor compares every object, relationship, and property in the file that is already open in the Schema Editor with the objects, relationships, and properties in the second file you select.

When you do a full comparison of two files, you can choose whether you want to display the comparison results alone, the instructions, or both the comparison results and instructions.

Full Comparison Results

Comparison results appear in Notepad for viewing. The comparison results list the objects that appear in one file, but not the other, and the objects that are different in the two files.

For example, the following graphic shows what full comparison results might look like:



To perform a full comparison and review comparison results, do the following:

1. Open the file to which you want to compare another file.
2. Click **File > Compare with File** > **File Type > Full Comparison > Display Results**.
3. Browse to the file that you want to compare to the file you opened previously.
4. Select the folder where you want the software to store the comparison results file, and then click **X** to close the dialog box.

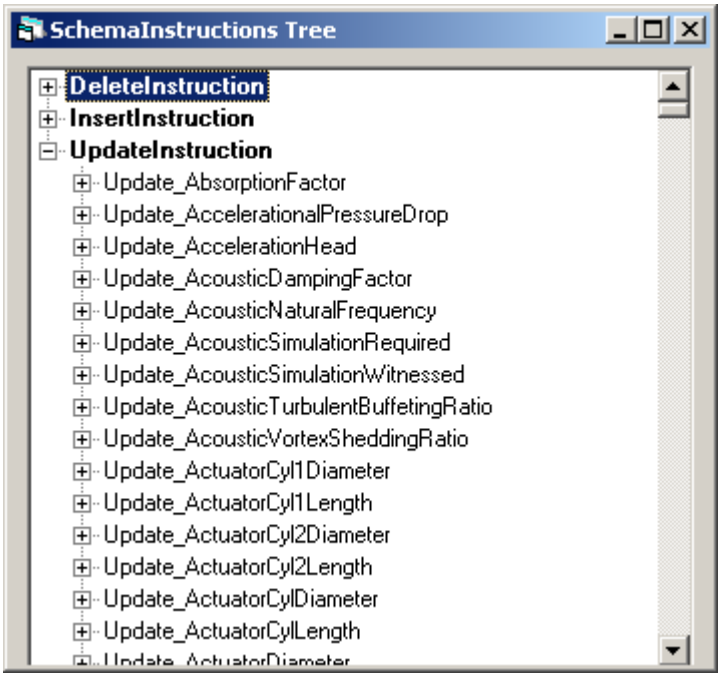
   When the comparison is complete, the results appear in Notepad.

NOTE After you do a full comparison, you can review the instructions that would be generated by the software if one file superseded the other. You can review these comparison instructions using the appropriate **Instructions** command on the **View** menu. For example, to view schema instructions, click **View > Schema Instructions > Tree/Table**.

Viewing Comparison Instructions

You can also choose to view the instructions generated when you compare two files. These instructions are a set of instruction objects that define the changes necessary to go from the file you selected for comparison to the file you opened first in the Schema Editor. These modification instructions define what would happen in one of the authoring tools when one file is superseded by the other. When you choose to view instructions, the **View File Type Instructions** dialog box appears to allow you to select the view in which you want to display the instructions.

For every object that exists in the second file that is not in the first file, the software creates a Delete instruction object. The software creates an Insert instruction object for every object that exists in the first file that is not in the second file. An Update instruction object is created for every object that exists in both files but that has differences in interfaces and/or properties.



To perform a full comparison and review comparison instructions, do the following:

1. Open the file to which you want to compare another file.
2. Click **File > Compare with File** > **File Type > Full Comparison > Display Instructions**.
3. Browse to the file that you want to compare to the file you opened previously.
4. In the **View File Type Instructions** dialog box, click the view in which you want to display the instructions.
5. Click **OK**.

NOTES

- You can also view instructions after a full compare by clicking **View > Schema Instructions**, **View > Data Instructions**, or **View > Tool Schema Instructions**.

- You can save instructions by clicking **File > Save > [FileType] Instructions**, where [FileType] is the type of file for which you want to save instructions. For example, to save data instructions, click **Data Instructions** on the **Save** submenu.

Viewing Both Comparison Results and Instructions

You can also choose to view both comparison results and instructions. When you view both, the comparison results appear in Notepad and the **View File Type Instructions** dialog box appears to allow you to select the view for the instructions.

To perform a full comparison and review both comparison results and instructions, do the following:

1. Open the file to which you want to compare another file.
2. Click **File > Compare with File** > **File Type > Full Comparison > Display Both**.
3. Browse to the file that you want to compare to the file you opened previously.
4. Select the folder where you want the software to store the comparison results file, and then click **X** to close the dialog box.

   When the comparison is complete, the results appear in Notepad.
5. In the **View File Type Instructions** dialog box, click the view in which you want to display the comparison instructions.
6. Click **OK**.

NOTES

- You can also view comparison instructions after a full compare by clicking **View > Schema Instructions**, **View > Data Instructions**, or **View > Tool Schema Instructions**.
- You can save instructions by clicking **File > Save > [FileType] Instructions**, where [FileType] is the type of file for which you want to save instructions. For example, to save data instructions, click **Data Instructions** on the **Save** submenu.

See Also

Comparison (on page 463)

Detect Tombstones

1. Open the schema, tool map schema, or data file in which you want to detect deleted items.
2. Click **File > Compare with File** and then point to the type of file you want to use as the basis for tombstone detection. For example, to compare two schemas, point to **File > Compare with File > Schema File**.
3. Click **Detect Tombstones** on the submenu.
4. Browse to the file to which you want to compare the schema, data file, or tool map schema.
5. After tombstone detection is complete, select the view that you want to use for the delete instructions on the **View** tab.

6. Click **OK**.

NOTES

- When you detect tombstones, only the delete instructions are displayed in the view you selected.
- You can save instructions by clicking **File > Save > [FileType] Instructions**, where [FileType] is the type of file for which you want to save instructions. For example, to save data instructions, click **Data Instructions** on the **Save** submenu.

See Also

Comparison (on page 463)
Tombstone Comparison Results: An Example (on page 469)

Detect New UIDs for Named Objects

You must open a schema or a file configuration that contains a schema, and open two different data file sets with the **Data File Set** and **Compare Data File Set** commands before using the **Detect New UIDs for Named Objects** command.

1. Click **File > Compare with File > Data File Set > Detect New UIDs for Named Objects**.
2. Browse to a results file in the **Filename** box.
3. Select a class or view map in the **Class/view map** drop-down box.
4. Click **OK**.

See Also

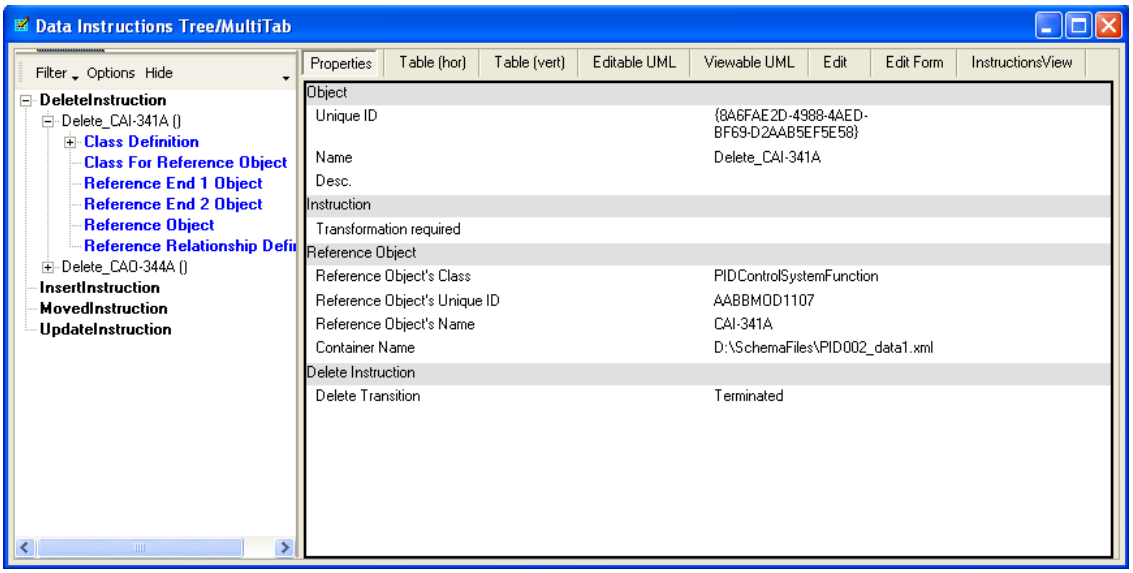
Open a Data File Set (on page 99)
Compare a Data File Set (on page 100)

Tombstone Comparison Results: An Example

In the Schema Editor, you can compare files to see only those things that exist in one file but that have been deleted in the other; these deletions are called tombstones. Because this type of comparison only checks for objects that exist in the second file you select that do not exist in the first file, tombstone detection typically runs faster than full comparisons.

After tombstone detection is complete, you can select the type of view that you want to use to view the tombstones on the **View** tab of the **View File Type Instructions** dialog box. In the view you select, you can only see Delete instructions for those objects that exist in the second file you selected, but not in the first file.

For example, when you compare two data files, you can view the Delete instructions in the **Data Instructions Tree/Table** window. In the following graphic, the Delete_E114A object is selected. The properties for the instruction object show the Class, UID, and Name of the object to be deleted.



To detect and view tombstones for a data file, do the following:

1. Open the data file to which you want to compare another file.
2. Click **File > Compare with File** > **Data File > Detect Tombstones**.
3. Browse to the data file that you want to compare to the file you opened previously.
4. In the **View File Type Instructions** dialog box, click the view in which you want to display the tombstones.
5. Click **OK**.

NOTE You can save instructions by clicking **File > Save > [FileType] Instructions**, where [FileType] is the type of file for which you want to save instructions. For example, to save data instructions, click **Data Instructions** on the **Save** submenu.

See Also

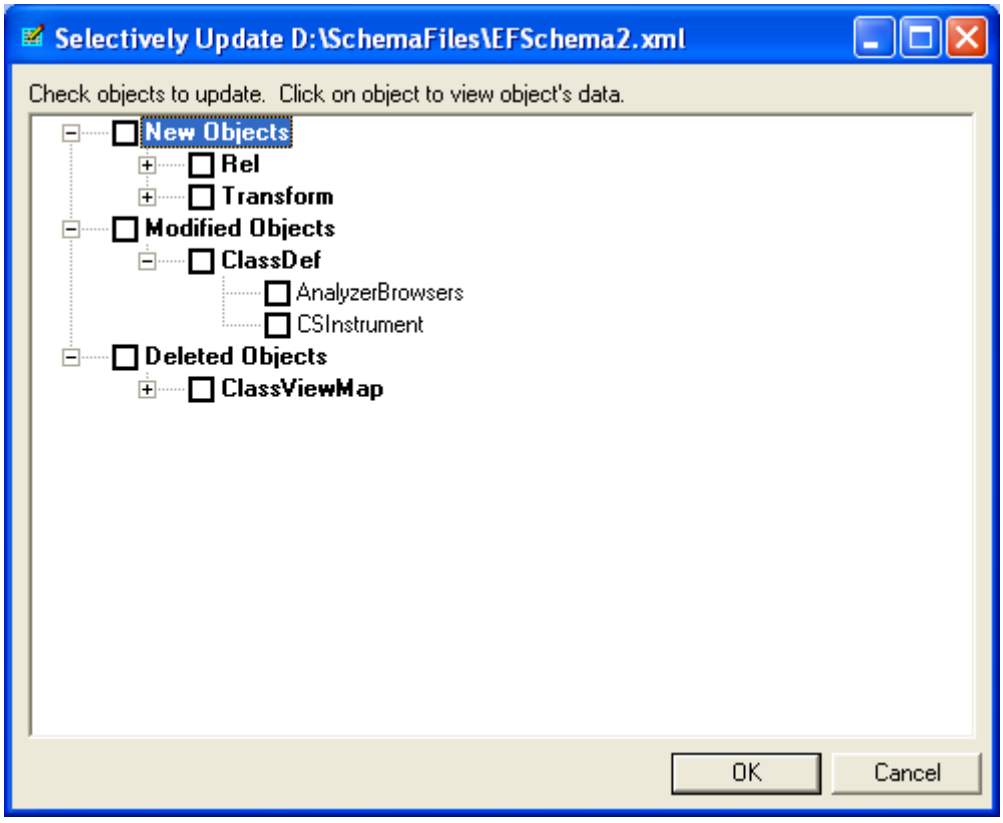
Comparison (on page 463)
Detect Tombstones (on page 468)

Selectively Update

Selectively Update is a very powerful tool for reconciling differences, but please use caution when using this tool because you can introduce errors into your schema.

When a comparison is performed with **Selectively Update**, the results of the comparison are presented in a **Tree** view, allowing you to choose which differences to resolve. This **Tree** view can contain nodes **New Objects**, **Modified Objects**, and **Deleted Objects** depending on the results of the comparison.

- The **New Objects** node, when expanded, identifies all the objects that exist in the compared with file (or configuration) that don't exist in the opened file (or configuration).
- The **Modified Objects** node, when expanded, identifies all the objects that exist in both files (or configurations) but have at least property that has different values.
- The **Deleted Objects** node identifies all the objects that exist in the opened file (or configuration) that don't exist in the compared with file (or configuration).



Selecting nodes from this tree control and then selecting **OK** will result in the changes for the selected nodes being made to the opened file (or configuration). Therefore, selecting all the nodes in this tree control and then selecting **OK** will result in the opened file being made to be the same as the compared with file.

By selectively choosing which modifications to apply, you can control which changes are actually made to the opened file (or configuration). Therefore, you can choose to take some subset of the differences and apply them to the opened file.

CAUTION Using **Selectively Update** improperly can introduce errors into the opened file. Therefore, you should be careful for any case where you select a subset of the full set of differences to be applied. The file (or configuration) should be validated after this command has been executed and any new problems that are found by validation should be resolved before proceeding further.

SECTION 14

Validate Schemas

In the Schema Editor, you can validate the schema, meta schema, tool map schemas, and data files. Although the Schema Editor provides commands to validate the schema, meta schema, and tool map schemas, these tasks are typically performed by software developers when changes are made to the SmartPlant schema and tool map schemas.

IMPORTANT Tool schemas must be validated using the Schema Editor before they are delivered and used with the authoring tool software. Error checking in the Schema Component assumes that the tool map schema validates.

Data validation is typically performed by software vendors who want to successfully communicate in an integrated environment. As data files are generated by authoring tools, they can be validated in the Schema Editor to make sure that they conform to the SmartPlant schema. You can set Schema Editor to automatically validate schemas or data files when objects are created are modified. This helps identify errors and allows you to correct them as immediately.

When you use the **Validate** commands on the **Tools** menu, the Schema Editor validates the selected file against the SmartPlant schema and a set of semantic rules. Validating a file performs a series of checks for potential problems with the data. If an authoring tool generates illegal data, the Schema Editor validation creates a list of errors. The Schema Editor automatically sets validation flag to indicate whether a set of objects have been successfully validated with no errors and no modification commands have been executed since the successful validation. This flag is set to True when a full validation is performed and no errors are detected. It is set to False by each and every command that modifies or adds any object in the set of objects.

After validation is complete, the Schema Editor displays validation errors in the **Validation Errors and Warnings** dialog box for your review. These errors help you diagnose problems with files. To help you more easily diagnose problems, you can filter the errors and warnings you see during validations using the **Set Validation Message Filtering** command.

See Also

Validate Files (on page 473)

Validate Files

1. On the **Tools** menu, point to the type of file that you want to validate.

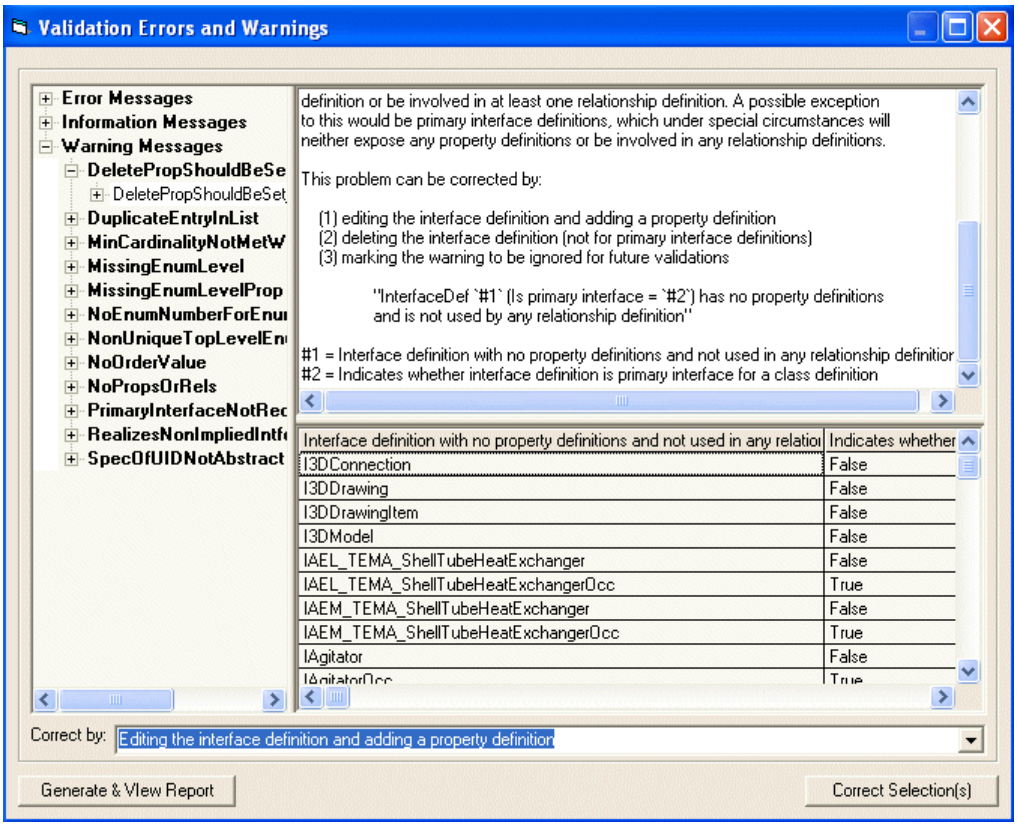
   NOTES

   - The schema file that you want to validate must be open.

- When you validate a schema file, the Schema Editor checks the file against the meta schema and a set of semantic rules. Validation results are displayed in a text editor.

TIPS

- You can select **Validate Schema/Display Results** on the toolbar to validate the active schema file and display validation results.
- Click **Tools > Another Schema > Validate All Schema Files > Display Results** to validated all opened schema files.
- To view error messages and warnings in Notepad, click **Generate & View Report**.



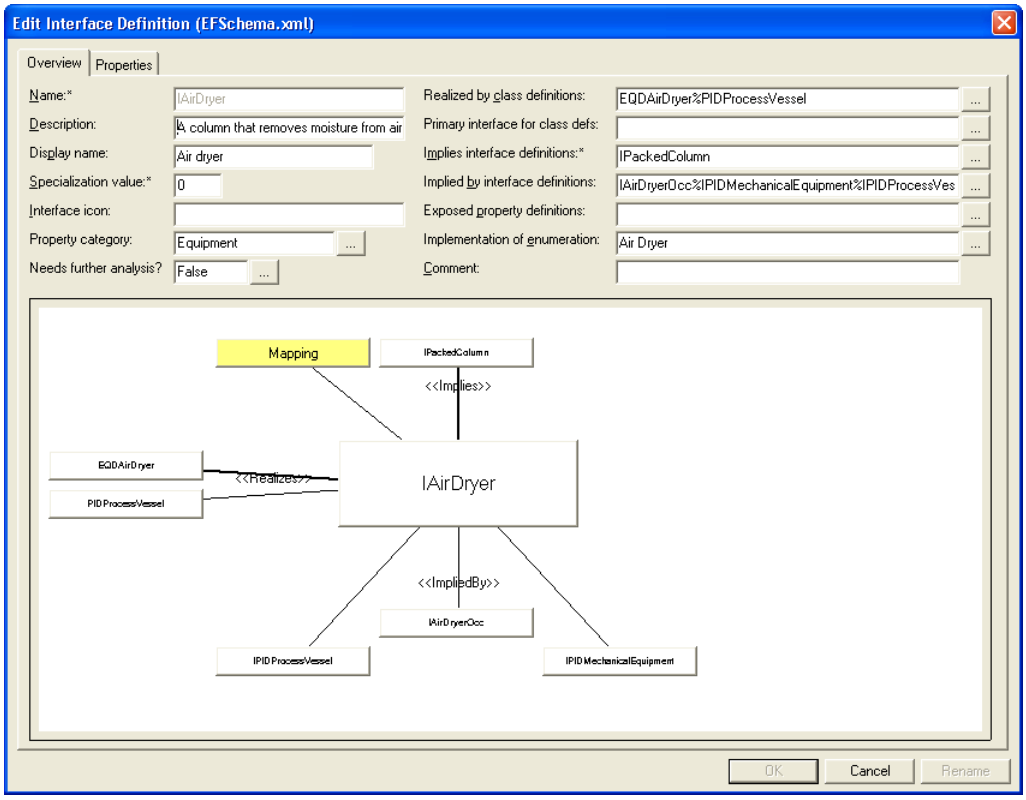
2. To display validation results, click **Validate > Validate/Display Results**.
3. On the **Validation Errors and Warnings**, use the **Tree View** to navigate the messages by type, group, or individually.

   Three types of messages appear in the tree view -- **Information Messages**, **Warning Messages**, and **Error Messages**.
4. Expand the type you want to view, then select a message.
   - If you select a group message, which appears **bold** in the **Tree View**, the message text appears in the upper right window and each object error related to that group appears in the **Error Table** in the lower right window.
   - If you expand the group message and select an object from its tree view, the message text appears in the upper right window and the object error related to that item appears in the **Error Table** in the lower right window.

5. To correct validation errors, select one or more items in the **Error Table**.
6. In the **Correct by** list, select the action you want to take, then click **Correct Selection(s)**.

   NOTES

   - To correct more than one object from the **Error Table**, hold the **Shift** key and select the consecutive objects you want to correct. You can only select consecutive objects.
   - If you select more than one object from the **Error Table**, the **Edit <object>** dialog box for the first object in the list appears first. When you click **OK**, the dialog box opens for the next object.
7. In the **Edit** <*object*> dialog box, make your corrections, then click **OK**.



8. When all corrections you want to make are complete, click **Close**.

NOTE If you want to filter particular types of errors out of the validation results, click **Tools > Set Validation Message Filtering**.

See Also

Validate Schemas (on page 473)

Automatic Validation

- Click **Edit > Turn AutoValidation On** to automatically validate the schema and data files.

Turning automatic validation on helps you identify actions he is performing that cause validation warnings and errors. If automatic validation is turned on and the user takes corrective steps to fix the problems as they arise, then the number of problems detected during full validation should be significantly less.

When automatic validation is turned on, any object created or modified during a command is validated when that command is completed. The validation uses many of the same checks used when a full validation of the schema is executed, but performs the check only against the set of objects created or modified during the command. The Schema Editor displays validation errors in the **Validation Errors and Warnings** dialog box for your review.

The use of automatic validation does not preclude the use of full validation since there are some checks that are only performed during full validation and the partial validation performed at the end of each command when automatic validation is on is insufficient to validate that the entire set of objects are error-free.

NOTE Turning on automatic validation may slow performance of the Schema Editor. To improve performance, turn off automatic validation and address validation problems with a full validation on the schema rather than at the end of each command.

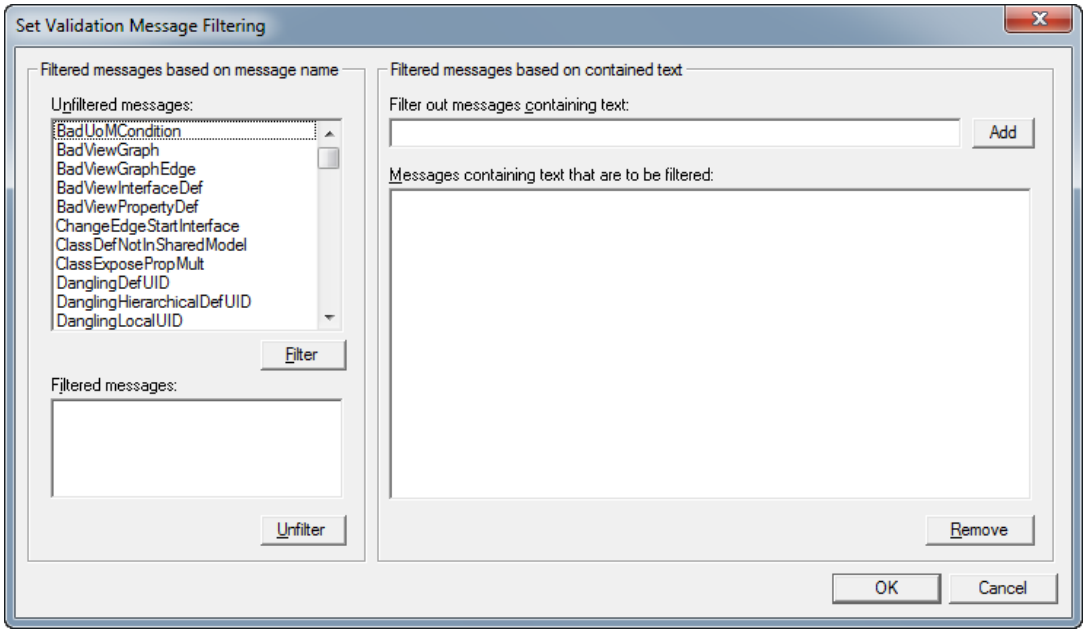
Set Validation Message Filtering

The **Set Validation Message Filtering** allows you to filter all messages of a certain group or message name, for example, any message in the BadViewGraph group. You can also filter message using specific text contained in an error or warning message, such as a UID, a specific graphs/views of a class, or a subset of the text contained in the error or warning message. Message filters

To filter messages, do one of the following:

- To filter by group or message name, select a message name from the **Unfiltered messages** list, the click **Filter**.

- Type the text by which to filter messages, then click **Add**.



SECTION 15

Verification of Tool Map Schema and Data Files

In an integrated environment, determining whether the data published by one application is sufficient for use by another application that retrieves that data is difficult and sometimes unachievable. A publishing application that provides some but not all of the necessary information required by a retrieving application can create complications for the retrieving application.

The process of checking that the information published by a publishing application contains all the necessary data required by a retrieving application is called verification. In Schema Editor, verification is performed on a business object basis. A particular publishing application might provide the necessary information for retrieval of some business objects by another application but fail to provide the necessary information for other business objects. Therefore, verification checking must be performed for each and every business object published by one application and retrieved by another application.

The Schema Editor supports two types of verification:

- Verification of a publishing application's tool map schema files versus one or more retrieving applications' tool map schema files
- Verification of a published data file versus one or more retrieving applications' tool map schema files

In both cases, published business objects are compared with the requirements for retrieval for those business objects (as defined by the retrieving application's map file) to determine the level of compliance of the publishing application.

The following definitions determine the level of compliance between a publishing application and a retrieving application.

- Level 0 compliance – Some of the required retrieved properties for application A are published by application B.
- Level 1- compliance – Some of the required retrieved properties for application A are optionally published by application B.
- Level 1+ compliance – All required retrieved properties for application A are published by application B. If a retrieving application has no required properties, then the publishing application need only publish one of the retrieved properties to qualify as level 1+ compliant.
- Level 2- compliance – Includes Level 1- compliance plus all optional retrieved properties for application A are published by application B (optional or required).
- Level 2+ compliance – Includes Level 1+ compliance plus all optional retrieved properties for application A are published by application B (optional or required)

When verification is run against either a publishing tool map schema file or a published data file, a verification report is produced that contains the results of the verification and includes the following information for each published business object.

- The name of the tool map schema file or the data file used for verification
- Published class definition (business object)
- Map class definition for retrieving application
- Level of compliance (0, 1-, 1+, 2-, 2+) between the publishing application and retrieving application for that business object

See Also

Verify Publish Tool Map Schemas (on page 479)
Verify Publish Data Files (on page 481)

Verify Publish Tool Map Schemas

1. Open a schema.
2. Open the tool map schema for the publishing application.
3. Open one or more tool map schema files for the retrieving application(s).
4. Click **Tools > Another Tool Schema File > Verify Active Map Schema**.

The following is an example of a verification results report.

```
Verifying publish definition for D:\SchemaFiles\MyMapSchemaPID.xml

Publish of InstrumentCtrlSysFunc

Publish of Drawing

Publish of InstrumentInline
      Retrieval by InstrumentInline verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml

Publish of Instrument
      Retrieval by InstrumentInline verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml

Publish of InstrLoop
      Retrieval by InstrLoop verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of ItemNote

Publish of Nozzle
      Retrieval by Nozzle verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of PipingComp
      Retrieval by PipingComp verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of PipeRun
      Retrieval by PipeRun verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of PipingPoint

Publish of EquipComponent
      Retrieval by EquipComponent verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of Representation

Publish of SignalRun
      Retrieval by InstrLoop verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of SignalPoint

Publish of PipingBranchPoint

Publish of SignalBranchPoint

Publish of Pipeline

Publish of EquipmentOther
      Retrieval by Exchanger verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Mechanical verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Vessel verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by EquipmentOther verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of Exchanger
      Retrieval by Vessel verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Exchanger verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Mechanical verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by EquipmentOther verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of Mechanical
      Retrieval by Vessel verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Exchanger verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Mechanical verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by EquipmentOther verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of Vessel
      Retrieval by Vessel verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Exchanger verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by Mechanical verified as level 1+  (D:\SchemaFiles\MyMapSchemaIN.xml)
      Retrieval by EquipmentOther verified as level 2+  (D:\SchemaFiles\MyMapSchemaIN.xml)

Publish of OPC
```

See Also

Verification of Tool Map Schema and Data Files (on page 478)
Verify Publish Data Files (on page 481)

Verify Publish Data Files

1. Open a schema.
2. Open one or more tool map schema files for the retrieving application(s).
3. Open a data file.
4. Click **Tools > Data > Verify Data File**.

   A verification results report appears.

Glossary

adapter

Authoring tool software that facilitates the sharing of data between the authoring tool and other integrated tools. Tool adapters generate XML files for publish operations and consume XML when tools retrieve documents. Adapters are also called SmartPlant adapters.

attribute

An object characteristic.

authoring tools

Applications where documents are created and then shared through integration. Can also refer to applications where 3D model data is created. Integrated authoring tools can be third-party applications or Intergraph applications, and include Aspen Basic Engineering, SmartPlant P&ID, SmartPlant Electrical, SmartPlant Instrumentation, Smart 3D, SmartPlant Materials, and SmartPlant Foundation.

common UI

An ActiveX component (.dll) that provides a standard user interface for integration functionality, such as publish, retrieve, and register. The authoring tools display the common UI when the user clicks particular SmartPlant commands in the authoring tool.

component schema

A subdivision of the complete SmartPlant schema that contains the set of class definitions that are used within a specific domain or application area.

container

An object used by the tool adapters and the SmartPlant software components to pass data back and forth between a tool and SmartPlant Foundation. A container may hold data or metadata related to the data model or actual instance data.

correlation

The relationship between items that represent the same object in multiple authoring tools.

database

Collection of files of comprehensive information that have predefined structure and organization; a specific program can communicate, interpret, or process these files.

design basis

An item in an authoring tool that represents an item from an upstream application (an application used earlier in the lifecycle of the plant). Plant items placed with the authoring tool correspond to a particular design basis item.

Design basis items provide a means of determining if the plant items within the authoring tool are consistent with the items from the upstream application and help users maintain consistency as changes are made in all authoring tools.

design file

A file generated by a design tool, such as SmartPlant P&ID or Aspen Basic Engineering.

domain

A set of data (for example: tool data, SmartPlant Foundation administration data, and schema data) that is segregated and managed independently in SmartPlant Foundation. Data segregation improves performance and maintains data integrity. The SmartPlant Enterprise authoring tools create data in one SmartPlant Foundation domain and publish into another domain.

enumerated entry

A member of an enumerated list that defines one possible value for a property in the SmartPlant schema. Enumerated values are sometimes called enumerated entries.

enumerated list

A list of possible string property values defined for a property definition in the SmartPlant schema. Enumerated sets are sometimes called enumerated lists, picklists, codelists, and lookups.

exposes

The relationship between interface definitions and property definitions in the SmartPlant schema. Interface definitions *expose* the property definitions for class definitions.

graph definition

A connected network of edge definitions with structure. Each graph definition in the SmartPlant schema starts at an interface definition and traverses through one or more relationship definitions to another interface definition at the other end. Graph definitions are sometimes referred to as directed graph definitions.

hierarchy

A classified structure with superiors, or roots, and subordinates, or dependents, used for grouping data.

implies

The relationship between two interface definitions in the SmartPlant schema. If an interface definition *implies* another interface definition, then any class definition that realizes the first interface definition can also realize the implied interface definition.

integration

Technology that standardizes and improves communication among the various SmartPlant Enterprise authoring tools used in design, construction, and operation of a plant. Integration manages data exchange among these authoring tools, which enables sharing and re-use of plant information throughout the plant lifecycle.

interface definition

A named collection of property definitions that represents a role for a class definition in the SmartPlant schema.

meta schema

A set of schema objects that describe the objects in the SmartPlant schema. The meta schema provides the building blocks upon which the SmartPlant schema is built.

picklist

A list of applicable values for a given property in SmartPlant Foundation.

Plant Breakdown Structure (PBS)

The composition of the plant based on the grouping of physical objects by their function in the plant. The plant usually occupies the top level of the hierarchy and is typically followed by areas and units.

property

An object characteristic.

property definition

A basic attribute shared by all members of an object class. Property definitions are grouped using interface definitions in the SmartPlant schema.

publish

To share a document and its data with other authoring tools by exporting an XML file containing the document data and relationships. When a document is published, the software places the XML file in the appropriate SmartPlant Foundation vault and loads the data from the XML file into the SmartPlant Foundation database. After the document is published, users can retrieve the data from the XML file located in the SmartPlant Foundation vault into other authoring tools.

realizes

The relationship between class definitions and interface definitions in the SmartPlant schema. Class definitions *realize* interface definitions. The interface definitions that are realized by a class definition expose the properties for that class definition.

register

To map an authoring tool and all its projects to a SmartPlant Foundation URL, which points to one SmartPlant Foundation database. You must register your authoring tool plant before you can publish and retrieve in an integrated environment.

relationship definition

Associations between interface definitions in the SmartPlant schema. Relationship definitions identify two specific objects that fulfill the roles on each end of the relationship.

resurrected object

A deleted authoring tool object that is reintroduced to the SmartPlant Foundation database when recovered tool data is republished. For example, a tool initially publishes a document containing an object, but later deletes the object and republishes. SmartPlant Foundation then detects that the object is deleted. However, if the tool restores data that was backed up before the object was deleted, through a data restoration or other tool-specific mechanism, and republishes, then SmartPlant Foundation recognizes that the object was previously deleted but has been reintroduced (resurrected).

schema

A model used to describe and validate the structure of XML files.

Schema Component

A suite of ActiveX components that provide functionality surrounding the creation, parsing, validation, and comparison of the SmartPlant schema and data. The tool adapters interact with the Schema Component to read the SmartPlant schema, to create data for publish, and to retrieve data.

scoped by

The relationship between property definitions and property types in the SmartPlant schema. The *scoped by* relationship specifies the property type that defines acceptable values, or scopes, a particular property definition. Every property definition in the SmartPlant schema is scoped by one and only one property type. All properties of that property definition must be of that property type.

shared object definition

A schema object used to group together similar class definitions that define the same object in different domains. Class definitions that can be shared have a Sharing relationship with shared object definitions in the SmartPlant schema.

SI

International System of Units, sometimes referred to as the metric system. When values for units of measure are published, they are converted to SI units and stored, regardless of the units of measure selected when the user defined the value in the authoring tool.

SmartPlant Enterprise

A suite of Intergraph engineering applications that are delivered together.

SmartPlant schema

An XML file that describes the structure of the XML files generated by integrated authoring tools in much the same way as a data dictionary describes the structure of a database. As tools publish documents in XML format, those documents must adhere to the format defined by the schema to ensure that the XML data can be loaded into SmartPlant Foundation and retrieved into the other authoring tools.

TEF

The Engineering Framework; technology behind the integration of SmartPlant Enterprise products.

tombstone

Delete instructions for an object that has been removed in one of the authoring tools. Upon retrieval of a tombstone, delete tasks are created in the authoring tool's **To Do List** to allow the tool to delete the object from its database.

tool

See authoring tool.

tool adapter

See adapter.

tool schema

A set of schema objects that describe the data in the authoring tool databases before it is transformed into the format prescribed by the SmartPlant schema. The tool schema also specifies the mapping between objects in the tool database and the SmartPlant schema.

tool signature

A unique identifier for the relationship between a plant in SmartPlant Foundation and a specific plant in an authoring tool database. The relationship is created when an authoring tool registers with SmartPlant Foundation.

unit of measure list

A collection of different units that measure the same property in SmartPlant Foundation.

UoM

A unit of measurement.

vault

A folder where files are stored on a host computer.

view definition

A named group of properties extracted from the possible properties that a graph definition exposes. View definitions are used in an integrated environment to provide a different view of data from that provided by the underlying schema.

Work Breakdown Structure (WBS)

The composition of the plant based on the construction work to be completed. The plant usually occupies the top level of the hierarchy; it is typically followed by projects, contracts, and documents.

XML

Extensible Markup Language; the format for all documents published or retrieved in an integrated environment. These XML files must conform to the structure defined by the SmartPlant schema.

Index

A

adapter • 482
Add a new schema file to a configuration file • 208
Add an Object to the Tool Metadata • 428
Add Objects to a Tool Map Schema • 429
Add Objects to One of the Tool Map Schemas • 364
Add Objects to the SmartPlant Schema • 334, 365, 429
Add Objects to the Tool Metadata • 333
Align Objects in Editable UML Views • 118
All object types • 160
Allow Multiple Users to Edit the Same CMF File • 88
Associate a vault with the new plant • 226
attribute • 482
authoring tools • 482
Auto-correlate SmartPlant Schema Enumerated Lists in Mapping Table • 402
Auto-Generate UML Diagrams Based on Published Data • 120
Automatic Validation • 476
Automatically Correlate a Tool Enumerated List with a Schema Enumerated List • 359
Automatically Correlate New Tool Map Objects with Schema Objects • 356
Automatically Create and Map New Enumerated Entries Created in the Tool • 348
Automatically Create and Map New Properties Created in the Tool • 352

B

Before You Begin Mapping for Tools with Metadata Adapters • 328

C

Cardinalities for Mapping • 267
Check in the CMF and load schema changes • 218
Check in the updated CMF file • 198
Check out and launch the CMF file • 213
Check Out the CMF and Launch Schema Editor • 333, 364
Check out the CMF and Launch the Schema Editor • 428
Class definitions • 160
Class Definitions • 19
Clear Editable UML Views • 118
common UI • 482
Compare a Data File Set • 100
Compare a Tool Map Schema to the SmartPlant Schema • 385
Compare Files • 464
Compare the Tool Map Schema to the SmartPlant Schema and Resolve the Differences • 365
Comparing Tool Map Schema with the SmartPlant Schema • 304
Comparison • 463
component schema • 482
Component Schemas • 42
Configuration Managed Files (CMFs) • 82
Configure the Reference File List • 410
Connecting to a SmartPlant Foundation Plant • 296
container • 482
Convert an Existing Pseudo File into a Physical Pseudo File • 197
Copy Objects from Reference File to Target File • 412
correlation • 482
Create a Class Definition • 169
Create a Component Schema • 170
Create a File Configuration • 93
Create a Graph Definition • 180
Create a Layout • 69
Create a Layout to Display Graphical Mapping Views • 425
Create a Map Class Definition • 430
Create a Map Edge Definition • 435
Create a Map Enumerated List Definition • 437
Create a Map Property Definition • 433
Create a Map Unit of Measure List Definition • 457
Create a New Document Type that Existing Tools Can Publish and Retrieve • 251
Create a new plant and structure • 225
Create a New Pseudo File in the Schema Editor • 193
Create a Personalized Menu Configuration • 65

Create a Property Definition • 172
Create a Relationship Between Schema Objects • 186
Create a Relationship Definition • 176
Create a schema extension file • 207
Create a Session File • 92
Create a UML Diagram • 111
Create a Unit of Measure List • 175
Create a View Definition • 182
Create an Edge Definition • 177
Create an Enumerated List • 173
Create an Interface Definition • 171
Create and Map New Enumerated List Entries Starting in the SmartPlant Schema • 366
Create and Map New Enumerated List Entries Starting in the Tool • 334
Create and Map New Properties with Enumerated Lists Starting in the SmartPlant Schema • 376
Create and Map New Properties with Enumerated Lists Starting in the Tool • 342
Create and Map New String Properties Starting in the SmartPlant Schema • 371
Create and Map New String Properties Starting in the Tool • 338
Create Mapping for Classes in Graphical Mapping • 417
Create Mapping for Enumerated Lists in Graphical Mapping • 397, 419
Create Mapping for Enumerated Lists with MultiMap Editor • 399
Create Mapping for Enumerating Lists in Map Environment • 395
Create Mapping for Units of Measure in Graphical Mapping • 420
Create new objects and relationships in the schema • 198
Create new objects and relationships in the software • 209
Create New Schema Objects • 168
Create SmartPlant Schema Enumerated Lists from Mapping Table • 401
Create Temporary Smart 3D Enumeration Mapping Relationships • 460
Creating an alternate plant breakdown structure (PBS) • 210
Creation rules • 159

D

database • 482
Define a custom PBS hierarchy • 215
Define a custom PBS hierarchy from a template • 214
Define Mapping • 59
Define Mapping Between the Tool Map Schema and the SmartPlant Schema • 365, 441
Define Mapping for a Tool Enumerated List • 452
Define Mapping for a Tool Unit of Measure List • 454
Define Naming Rules for Objects • 78
Define Publish Mapping for Tool Properties • 442
Define Retrieve Mapping for Tool Properties • 447
Define the Storage Location for Your Personalized Menu Configurations • 67
Defining Authoring Tool Mapping • 289
Delete Relationships for Graphical Mapping • 425
Deletion rule • 159
design basis • 482
design file • 483
Detect New UIDs for Named Objects • 469
Detect Tombstones • 468
Disable or Enable an Existing Property Relationship in the Schema • 361
Display Options for Graphical Mapping • 423
Distribute the Modified Tool Map Schema and SmartPlant Schema • 462
domain • 483

E

Edge Definitions • 33
Edge Definitions, Graph Definitions, and View Definitions • 33
Edit in Excel • 231
Editable Configurations • 84
Enumerated entries • 164
enumerated entry • 483
enumerated list • 483
Enumerated lists • 163
Enumerated Lists • 27
Excel (.xlsm) File • 231
Export a Displayed View • 229
Export Pseudo File Configurations • 228
Export Schema Containers • 228
Export the Schema to Excel • 228
Export Tool Mapping to a Spreadsheet • 325

exposes • 483
Extend the Existing Stand-Alone SmartPlant Foundation Schema • 206
Extending the Schema for SmartPlant Integration • 191
Extending the SmartPlant Schema • 146
Externalize Authoring Tool Metadata • 240
Extract the schema file to load • 198

F

File Configurations • 93
Filter Options for Graphical Mapping • 423
Filter the Mapping Objects in the Map Environment • 394
Filter the Reference File Objects in the Integrated Reference File Environment • 409
Filtered Interface-Based Query • 140
Find Objects by Class • 125
Find Objects by Interface • 135
Find Objects Using Predefined Named Query • 133
Find Objects Using Search Criteria • 124
Find Objects Using Simple Criteria • 128
Finding Objects in the Schema Editor • 124
Forward and Backward Schema Compatibility • 47
Full Comparison
An Example • 465
Full Interface-Based Query • 137

G

Generate authoring schema and load files • 221
Generate Authoring Tool Metadata Files • 257
Generate component schema files • 220
graph definition • 483
Graph Definitions • 37

H

hierarchy • 483

I

implies • 483
Import Data into Schema Editor • 237
Import Enumerated List Entries from a Spreadsheet • 174, 384, 440
Import Existing Customizations in Extensions.xml into the CMF File • 203
Import from Excel • 237
Integrated Reference File Environment Color Legend • 410
integration • 483
interface definition • 483
Interface definitions • 161
Interface Definitions • 22

K

Keyword Search • 129

L

Layouts • 69
Load alternate PBS objects • 224
Load schema changes into the database • 199
Load stand-alone schema changes into SmartPlant Foundation • 209
Load the pseudo file changes into the data warehouse database • 202
Load the Updated Schema into SmartPlant Foundation • 334, 366, 462

M

Map Between the Tool and the SmartPlant Schema • 334
Map Enumerated List and Map Unit of Measure List Mapping • 283
Map Environment Color Legend • 395
Map Publish and Retrieve Properties in a Graphical View • 414
Map Publish and Retrieve Properties in the Map Environment • 391
Mapping Directly Against the SmartPlant Schema • 277
Mapping New Objects Originating in the SmartPlant Schema • 363
Mapping New Objects Originating in the Tool Metadata • 332
Mapping Objects Created in the Authoring Tool • 334
Mapping Objects Created in the SmartPlant Schema • 366
Mapping Relationship Definitions • 266
Mapping with Metadata Adapters • 291
Mark Individual Map Enumerated Lists as Not to be mapped • 405
Mark Multiple Map Enumerated Lists as Not to be mapped • 406
Merge a Previously Generated Schema into a Pseudo File • 195

meta schema • 484
Meta Schema • 41
Model Definitions • 20
Modify a File Configuration • 94
Modify an Existing Personalized Menu Configuration • 68
Modify Existing View and Graph Definitions • 187
Modify Reference Files with the Integrated Reference File Environment • 408
Modifying Objects and Relationships in the Schema • 187
Move Customizations from Project to Project • 205
Multiple Editable Configurations • 84

N

Naming Conventions in the SmartPlant Schema • 146
Navigating Files and Data in the Schema Editor • 101

O

Open a Data File • 99
Open a Data File Set • 99
Open a File Configuration • 94
Open a Personalized Menu Configuration File • 67
Open a Schema • 56, 97
Open a Session File • 92
Open a Startup File • 56
Open a Tool Map Schema • 98
Open a Tool Schema • 56
Open Other Schema Files • 56
Open Saved UML Views • 119
Open the CMF File for Editing • 192
Opening and Using Files in the Schema Editor • 82
Opening Files • 96
Other Workflows Dialog Boxes • 60

P

Patterns and Abstract Roles • 147
Personalized Menus • 63
picklist • 484
Plant Breakdown Structure (PBS) • 484
Preface • 9
Prepare the pseudo file data to be loaded into SmartPlant Foundation • 199
property • 484
property definition • 484
Property definitions • 163
Property Definitions • 26
Pseudo Files • 85
Publish • 484
Publish and Retrieve File Types for Tools • 239

R

realizes • 484
Regenerate the component schema files • 203
register • 484
Register and publish the PBS • 226
relationship definition • 484
Relationship definitions • 167
Relationship Definitions • 31
Remove incomplete schema relationships • 461
Remove Objects from Editable UML Views • 118
Remove the existing PBS structure • 212
Resize the Graphical Mapping Tree View • 424
resurrected object • 484
Revision rule • 159
Rule enforcement • 159

S

schema • 485
Schema Component • 485
Schema Editor - Standard Overview • 55
Schema Editor Naming Rules • 71
Schema Editor Roles • 64
Schema Editor Toolbars • 53
Schema Editor User Interface • 48
Schema modification rules • 158
scoped by • 485
Search for Schema Objects • 58
Search for Tool Schema Objects • 59
Selectively Update • 471
Session Files • 92
Set Dependencies for a Schema File • 95
Set the Active Schema File • 56, 98
Set the Active Tool Schema File • 99
Set Validation Message Filtering • 476
Set View Options • 123
shared object definition • 485
Shared object definitions • 167
Shared Object Definitions • 21
SI • 485
Simple Interface-Based Query • 135

Single Active Configuration with New Objects in Multiple Configurations • 84
SmartPlant Enterprise • 485
SmartPlant Mapping Overview • 258
SmartPlant schema • 485
SmartPlant Schema Documentation • 144
SmartPlant Schema Editor Overview • 17
SmartPlant Schema Modification Rules • 156
SmartPlant Schema Overview • 18
Special Behavior Supported While Editing the Excel Workbook • 234
Special Configuration Commands • 86
Standard Workflows Window • 49
Synchronize Another Tool Map Schema with Tool Metadata • 57
Synchronize Dialog Box • 331
Synchronize the Tool Map Schema with the Tool Map Schema • 364
Synchronize the Tool Map Schema with the Tool Metadata • 333
Synchronize Tool Metadata with the Tool Map Schema • 328
Synchronizing Tool Map Schemas with Tool Metadata • 303

T

TEF • 485
tombstone • 485
Tombstone Comparison Results
An Example • 469
tool • 485
tool adapter • 486
Tool Map Schema Objects • 261
Tool Map Schemas • 259
Tool Objects Required for Metadata Adapters • 297
tool schema • 486
Tool Schema Models • 260
tool signature • 486
Tree View
An Example • 103
Tree/Editable UML View
An Example • 108
Tree/MultiTab View
An Example • 122
Tree/Table View
An Example • 105
Tree/Viewable UML View
An Example • 107
Types of Mapping for Map Class Definitions • 271

U

UIDs File • 236
UML Notation • 45
Understanding Mapping • 428
Unit of measure • 166
unit of measure list • 486
Unit of measure lists • 165
Units of Measurement • 31
Unmap Existing Properties in the Schema • 362
UoM • 486
Update Document Types that Existing Tools Can Publish and Retrieve • 255
Using Excel to Export and Import Schema • 227
Using View Definitions to Define Mapping • 275

V

Validate Files • 473
Validate Schemas • 473
vault • 486
Verification of Tool Map Schema and Data Files • 478
Verify Publish Data Files • 481
Verify Publish Tool Map Schemas • 479
View and Map Properties in the Graphical Mapping Table • 421
view definition • 486
View Definitions • 39
View Mapping Relationships in Graphical Mapping • 421
View Options • 123
View SmartPlant Schema Documentation • 145
View Tool-to-SmartPlant Mapping in a Viewable UML View • 308
View Tool-to-Tool Mapping for Classes and Properties • 316
View Tool-to-Tool Mapping for Enumerated Lists • 319
View Tool-to-Tool Mapping for Units of Measure • 322
View Tool-to-Tool Mapping in a Editable UML View • 313
Viewable Configurations • 84
Viewing Authoring Tool Mapping • 307

W

What's New in SmartPlant® Schema Editor? • 12

Work Breakdown Structure (WBS) • 486
Work Process for Mapping for Tools without Metadata Adapters • 427
Work Processes for Mapping for Tools with Metadata Adapters • 327

X

XML • 486
XML Files • 43