

SmartPlant Modeling and Mapping

Course Guide

May 2007

Version 3.8

Process, Power & Marine



 **INTERGRAPH**

SmartPlant Modeling and Mapping *Course Guide*

May 2007

Version 3.8

DSPF1-TP-100008A

Copyright

Copyright © 2002 - 2007 Intergraph Corporation. All Rights Reserved.

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.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) of the *Contractor Rights in Technical Data* clause at DFARS 252.227-7013, subparagraph (b) of the *Rights in Computer Software or Computer Software Documentation* clause at DFARS 252.227-7014, subparagraphs (b)(1) and (2) of the *License* clause at DFARS 252.227-7015, or subparagraphs (c) (1) and (2) of *Commercial Computer Software--Restricted Rights* at 48 CFR 52.227-19, as applicable.

Unpublished---rights reserved under the copyright laws of the United States.

Intergraph Corporation
Huntsville, Alabama 35894-0001

Warranties and Liabilities

All warranties given by Intergraph Corporation about equipment or software are set forth in your purchase contract, and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of such warranties. 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 discussed in this document is furnished under a license and may be used or copied only in accordance with the terms of this license.

No responsibility is assumed by Intergraph for the use or reliability of software on equipment that is not supplied by Intergraph or its affiliated companies. THE USER OF THE SOFTWARE IS EXPECTED TO MAKE THE FINAL EVALUATION AS TO THE USEFULNESS OF THE SOFTWARE IN HIS OWN ENVIRONMENT.

Trademarks

Intergraph, the Intergraph logo, SmartSketch, FrameWorks, SmartPlant, SmartPlant Foundation, SmartPlant P&ID, SmartPlant Instrumentation and INtools are registered trademarks of Intergraph Corporation. Microsoft and Windows are registered trademarks of Microsoft Corporation. Other brands and product names are trademarks of their respective owners.

This courseware was developed by Bill Crego and Mitch Harbin, PPM-PIM Training, Huntsville, Alabama.

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

Course Outline

Day 1

1. Overview of SmartPlant

- SmartPlant Components
 - The SmartPlant Schema
 - Authoring Tool Schemas
 - Schema Mapping
 - Introduction to the Schema Editor
-  **Activity – Starting the Schema Editor**

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

INTERGRAPH

2. *Using to the Schema Editor*

- Viewing an Open Schema File
- Finding Schema Objects
- Custom Drag-Drop UML View
- Change Object Display/Object Color
- Placing Text in a View
- Align Nodes
- Erasing Displayed Relationships
- Open a Package
- Erasing the View

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Activity – Using the Schema Editor
 - Creating a Project Schema
- Activity – Creating a Project Schema

Day 2

3. Schema Overview Modeling New Classes

- Model Definitions
- Component Schemas
- Interactive Activity – Creating a Component Schema

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Class Definitions
 - █ Interactive Activity – Creating a Class Definition
- Interface Definitions
 - █ Interactive Activity – Creating an Interface Def
- █ Activity 1 – Reviewing Schema Concepts
- █ Activity 2 – Creating Objects and Relationships with Schema Editor

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

INTERGRAPH

4. *Creating Properties, Enumerated Lists and Relationships*

- Relationship Definitions
- Properties and Property Types
 - Interactive Activity – Creating Property Defs
- Enumerated List Type
 - Interactive Activity – Creating an Enumerated List Property
 - Interactive Activity – Creating Realized Interface Defs

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

INTERGRAPH

- Relationship Definitions
 - Interactive Activity – Creating Relationship Definitions
- Shared Object Definitions
 - Activity – Creating Properties, Enumerated Lists and Relationships

Day 3

5. *Creating Edge Defs, Graph Defs and View Defs*

- Edge Definition
 - Interactive Activity – Creating Edge Definitions

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Creating Graph Defs
- Interactive Activity – Creating Graph Defs
- View Definitions and Class View Maps
- Interactive Activity – Creating View Defs and Class View Maps

6. Viewing and Finding Data

- XML Files Overview
- Opening and Viewing a Data File
- Finding Data Objects

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

INTERGRAPH

- Comparing Data Files
- Activity – Viewing and Finding Data

7. Introduction to Schema Mapping

- Extending Authoring Tool Enum Lists
- Finding the SmartPlant P&ID Data Dictionary Code List ID
- SmartPlant Schema Additions
- Tool Schemas Overview
- Modifying the SmartPlant Schema
- Extending an Existing Enumerated List
- Mapping an Existing Enumerated List
- Published and Retrieved Document Types

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- SmartPlant Foundation Mapping Spreadsheets
 - Activity 1 – Extending the Authoring Tool
 - Activity 2 – Mapping with the Schema Editor

8. Mapping with SmartPlant P&ID

- Adding Properties to the Meta schema
- Performing a Schema Analysis
- SmartPlant Schema Additions
- Adding a New SmartPlant Property
- Mapping New Properties
 - Activity 1 – Adding and Mapping a Simple Property

SmartPlant® SmartPlant Modeling and Foundation Mapping (SPMM)

INTERGRAPH

Day 4

- Adding Enum Extensions to the Schemas
- Adding a Complex Property to the Schemas
- Mapping a Complex Property
 - Activity 2 – Adding and Mapping a Complex Property

9. Loading and Testing the Schema Changes

- Schema Load Process
- Testing the Schema Load

SmartPlant® Foundation SmartPlant Modeling and Mapping (SPMM)

INTERGRAPH

- Publishing Documents
- Viewing the Created XML File
- Activity – Loading and Testing the Schema Changes

10. Mapping with SmartPlant Instrumentation

- Modifying the Authoring Tools
- Extending the SPPID Tool Map Schema
- Mapping New Enum Entries for SPPID
- Extending the SPI Tool Map Schema
- Mapping New Enum Entries for SPI

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Modifying View Definitions
 - Loading the Extensions into SmartPlant Foundation
 - Publishing a Change from SmartPlant P&ID
 - Retrieving a Change into SmartPlant Instrumentation
-  Activity 1 – Extending an Existing Enumerated List
- Adding a Custom Property to SmartPlant Instrumentation
 - Mapping a Custom Property in SPI

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Retrieve Mapping for a Custom Property
- Testing Custom Property Retrieval
- Publish Mapping for a Custom Property
- Testing Custom Property Publishing
 - Activity 2 – Adding a Custom Property to SmartPlant Instrumentation
- Adding a Custom Property to SmartPlant Instrumentation
- Mapping a Custom Property in SPI

SmartPlant® SmartPlant Modeling and Foundation Mapping (SPMM)

INTERGRAPH

11. Mapping with SmartPlant Electrical

- Adding Properties to the Meta schema
 - Performing a Schema Analysis
 - Mapping New Properties
-  Activity 1 – Adding and Mapping a Simple Property
- Adding Enum Extensions to the Schemas
 - Adding a Complex Property to the Schemas
 - Mapping a Complex Property
 - Testing Mapped Properties
-  Activity 2 – Adding and Mapping a Complex Property

SmartPlant® **SmartPlant Modeling and** **Foundation** **Mapping (SPMM)**

INTERGRAPH

Day 5

12. Mapping with SmartPlant 3D

- General Information about SmartPlant 3D Schema Mapping
- Adapters and Map Files
- How Mapping is Configured for Retrieve
- How Mapping is Configured for Publish
- Publish and Retrieve Limitations
- Design Basis Mapping
- Limitations on the SmartPlant Schema
- Mapping Configuration for Retrieve in SP3D

SmartPlant® SmartPlant Modeling and Mapping (SPMM)

- Loading objects into SP3D
- Mapping Configuration for Publish in SP3D
- Verify Mapped data from SP3D
 - Activity – Mapping with SmartPlant 3D

Wrap up and Dismissal

Table of Contents

1. Overview of SmartPlant	1-3
1.1 SmartPlant Components	1-6
1.1.1 The SmartPlant Client	1-7
1.1.2 The SmartPlant Server	1-9
1.1.3 The SmartPlant Loader	1-10
1.2 The SmartPlant Schema	1-12
1.2.1 Introduction to Class Definitions	1-18
1.2.2 Introduction to Property Definitions	1-20
1.2.3 Introduction to Interface Definitions	1-22
1.2.4 Relationships	1-25
1.3 Authoring Tool Schemas	1-26
1.4 Schema Mapping	1-29
1.5 Introduction to the Schema Editor	1-31
1.5.1 Starting the Schema Editor	1-34
1.5.2 Exiting the SmartPlant Schema Editor	1-39
1.6 Using the Class VM Session	1-41
1.7 Activity – Starting the SmartPlant Schema Editor	1-47
 2. Using the Schema Editor	 2-3
2.1 Viewing an Open Schema File	2-5
2.1.1 Schema File Tree View	2-7
2.1.2 Schema Tree/Properties View	2-10
2.1.3 Schema Tree/Table View	2-13
2.1.4 Schema Tree/Drag-Drop UML View	2-17
2.1.5 Interface Relationships	2-23
2.1.6 Schema Tree/UML View	2-27
2.1.7 Schema Editor View	2-32
2.2 Finding Schema Objects	2-35
2.2.1 View Schema Find Tab	2-35
2.2.2 Finding Objects from the Workflows Dialog Box	2-41
2.3 Custom Drag-Drop UML View	2-44
2.4 Change Object Display	2-57
2.5 Change Object Color	2-59
2.6 Placing Text in a View	2-61
2.7 Align Nodes	2-65
2.8 Erasing Displayed Relationships	2-66
2.9 Open a Package	2-68
2.10 Erasing the View	2-72
2.11 Exiting the Schema Editor	2-75
2.12 Activity 1 – Using the Schema Editor	2-77

2.13 Creating a Project Schema	2-87
2.14 Activity 2 – Creating a Project Schema	2-101
3. Schema Overview and Modeling New Classes	3-3
3.1 Model Definitions	3-5
3.2 Component Schemas	3-8
3.2.1 Properties of a Component Schema	3-10
3.3 Interactive Activity – Creating a Component Schema	3-11
3.4 Class Definitions	3-18
3.4.1 Properties of a Class Definition	3-19
3.5 Interactive Activity – Creating a Class Definition	3-21
3.6 Interface Definitions	3-27
3.6.1 Role Example	3-28
3.6.2 Interface Definitions and Relationships	3-29
3.6.3 Properties of an Interface Definition	3-32
3.7 Interactive Activity – Creating Interface Definitions	3-34
3.8 Activity 1 – Reviewing Schema Concepts	3-49
3.9 Activity 1 – Answer Key	3-51
3.10 Activity 2 – Creating Objects and Relationships with Schema Editor	3-53
4. Creating Properties, Enumerated Lists and Relationships	4-3
4.1 Property Definitions	4-4
4.1.1 Properties of a Property Definition	4-5
4.2 Property Types	4-6
4.3 Interactive Activity – Creating Property Definitions	4-7
4.4 Enumerated List Type	4-19
4.4.1 Properties of an Enumerated List Type	4-24
4.4.2 Unit of Measure List Types	4-26
4.4.3 Properties of a Unit of Measure List Type	4-28
4.5 Interactive Activity – Creating an Enumerated List Property	4-30
4.6 Interactive Activity – Creating a Realized Interface Definition	4-44
4.7 Relationship Definitions	4-56
4.7.1 Properties of Relationship Definitions	4-59
4.8 Shared Object Definitions	4-70
4.8.1 Properties of a Shared Object Definition	4-72
4.9 Activity – Creating Properties, Enumerated Lists and Relationships	4-73

5. Creating EdgeDefs, Graph Defs, View Defs and Class View Maps	5-3
 5.1 Edge Definitions	5-5
5.1.1 Properties of an Edge Definition	5-7
 5.2 Interactive Activity – Creating Edge Definitions	5-17
 5.3 Graph Definitions	5-45
5.3.1 Properties of a Graph Definition	5-47
 5.4 Interactive Activity – Creating Graph Definitions	5-49
 5.5 View Definitions	5-53
5.5.1 Properties of a View Definition	5-55
 5.6 Interactive Activity – Creating View Definitions	5-57
 5.7 Class View Maps	5-66
5.7.1 Properties of a Class View Map	5-67
 5.8 Interactive Activity – Creating Class View Maps	5-68
6. Viewing and Finding Data	6-3
 6.1 XML Files Overview	6-6
 6.2 Opening and Viewing a Data File	6-8
6.2.1 Data File Tree/UML View	6-10
6.2.2 Data Tree/Table View	6-16
6.2.3 Data Tree/Properties View	6-19
 6.3 Finding Data Objects	6-21
6.3.1 View Data Find Tab	6-21
 6.4 Comparing Data Files	6-27
6.4.1 Full Comparison	6-29
6.4.2 Detection of Tombstones	6-44
 6.5 Activity – Viewing and Finding Data	6-49
7. Introduction to Schema Mapping	7-3
 7.1 Extending Authoring Tool Enum Lists	7-11
 7.2 Finding the SmartPlant P&ID Data Dictionary Code List ID	7-16
 7.3 SmartPlant Schema Additions	7-23
7.3.1 Modifying the Config File	7-30
 7.4 Tool Schemas Overview	7-33
 7.5 Modifying the SmartPlant Schema	7-34
7.5.1 Connecting to SmartPlant	7-35
7.5.2 Verifying the SmartPlant Schema	7-43
7.5.3 Extending an Existing Enumerated List	7-47
7.5.4 Mapping an Existing Enumerated List	7-56

7.5.5 Saving Schema Changes _____	7-60
7.6 Published and Retrieved Document Types _____	7-64
7.7 SmartPlant Foundation Mapping Spreadsheets_____	7-73
7.8 Activity 1 – Extending the Authoring Tool _____	7-75
7.9 Activity 2 – Mapping with the Schema Editor _____	7-79
 8. Mapping with SmartPlant P&ID _____	8-3
8.1 Adding Properties to the Meta schema _____	8-6
8.2 Performing a Schema Analysis _____	8-10
8.3 SmartPlant Schema Additions _____	8-15
8.3.1 Adding a New SmartPlant Property _____	8-20
8.3.2 Mapping New Properties _____	8-25
8.3.3 Saving Mapping Changes _____	8-33
8.4 Activity 1 – Adding and Mapping a Simple Property _____	8-37
8.5 Adding a New Select List/Enum List _____	8-43
8.5.1 Adding Enum Extensions to the Schemas _____	8-50
8.5.2 Adding a Complex Property to the Schemas _____	8-61
8.5.3 Mapping a Complex Property _____	8-65
8.5.4 Saving Enum/Property Additions _____	8-71
8.6 Activity 2 – Adding and Mapping a Complex Property _____	8-73
 9. Loading and Testing the Schema Changes _____	9-3
9.1 Schema Load Process _____	9-3
9.2 Testing the Schema Load _____	9-10
9.2.1 Publishing Documents _____	9-15
9.3 Viewing the Created XML File _____	9-19
9.4 Activity – Loading and Testing the Schema Changes _____	9-25
 10. Mapping with SmartPlant Instrumentation _____	10-3
10.1 Modifying the Authoring Tools _____	10-4
10.2 Extending an Enumerated List in SmartPlant P&ID _____	10-5
10.2.1 Extending the SPPID Tool Map Schema _____	10-9
10.2.2 Mapping New Enum Entries for SPPID _____	10-17
10.2.3 Saving SPPID Schema Changes _____	10-19
10.3 Extending an Enumerated List in SmartPlant Instrumentation _____	10-22
10.3.1 Extending the SPI Tool Map Schema _____	10-32
10.3.2 Mapping New Enum Entries for SPI _____	10-34
10.3.3 Saving SPI Schema Changes _____	10-38
10.4 Modifying View Definitions _____	10-41

10.5 Loading the Extensions into SmartPlant Foundation	10-52
10.6 Publishing a Change from SmartPlant P&ID	10-56
10.7 Retrieving a Change into SmartPlant Instrumentation	10-61
10.8 Activity 1 – Extending an Existing Enumerated List	10-70
10.9 Adding a Custom Property to SmartPlant Instrumentation	10-75
10.9.1 Mapping a Custom Property in SPI	10-85
10.9.2 Retrieve Mapping for a Custom Property	10-87
10.9.3 Testing Custom Property Retrieval	10-102
10.9.4 Publish Mapping for a Custom Property	10-107
10.9.5 Testing Custom Property Publishing	10-114
10.10 Activity 2 – Adding a Custom Property to SmartPlant Instrumentation	10-119

11. *Mapping with SmartPlant Electrical* **11-3**

11.1 Adding a New Simple Property	11-6
11.1.1 Performing a Schema Analysis	11-13
11.1.2 Updating the Tool Schema	11-17
11.1.3 Mapping the New Property	11-22
11.1.4 Saving Mapping Changes	11-27
11.2 Activity 1 – Adding and Mapping a Simple Property	11-31
11.3 Adding a New Select List/Enum List	11-36
11.3.1 Adding Enum Extensions to the Tool Schema	11-43
11.3.2 Mapping Enumerated List Entries	11-48
11.3.3 Mapping a Complex Property	11-53
11.3.4 Saving Enum/Property Additions	11-56
11.4 Test Mapped Properties	11-59
11.5 Activity 2 – Adding and Mapping a Complex Property	11-69

12. *Mapping with SmartPlant 3D* **12-3**

12.1 SmartPlant 3D Schema Mapping: An Overview	12-4
12.2 General Information about SmartPlant 3D Schema Mapping	12-5
12.2.1 Adapters	12-5
12.2.2 Map Files	12-6
12.3 How Mapping is Configured for Retrieve	12-7
12.4 How Mapping is Configured for Publish	12-8
12.5 Other Information about Mapping	12-9
12.6 Publish Limitations	12-10
12.7 Retrieve Limitations	12-11
12.8 Design Basis Mapping	12-12
12.9 Limitations on the SmartPlant Schema	12-13
12.10 Mapping Configuration for Retrieve in SP3D	12-14

12.11 Loading objects into SP3D	12-26
12.12 Publishing 3D Map Data: An Overview	12-37
12.13 Mapping Configuration for Publish in SP3D	12-39
12.14 Verify Mapped data from SP3D	12-56
12.15 Activity – Mapping with SmartPlant 3D	12-71

<i>Appendix A: SmartPlant Schema Overview</i>	A-3
<i>Appendix B: SmartPlant Schema Evolution</i>	B-3
<i>Appendix C: Modifying the Pipeline Name Display</i>	C-3
<i>Appendix D: Extending Equipment Process Data</i>	D-3
<i>Appendix E: Upgrading a Tool Map Schema</i>	E-3

C H A P T E R

1

Overview/Review of SmartPlant Integrated Engineering Architecture

1. Overview of SmartPlant

SmartPlant supports the integration of engineering authoring tools, such as SmartPlant® P&ID, SmartPlant Electrical, SmartPlant 3D, SmartPlant Instrumentation, PDS, and Aspen Zygad. This integration addresses the flow of data as it moves from one engineering application to another through its lifecycle.



Introduction of SmartPlant Modeling

SmartPlant is a software-based platform that allows for integration of a wide-variety of engineering functions that occur during the process plant life cycle, from initial concept to decommissioning.

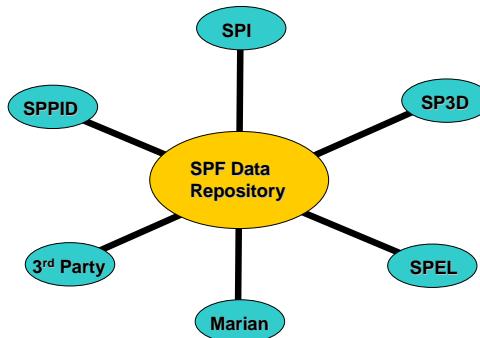
Tantamount to SPF is the facilitation of data flow as data moves from one engineering software application (such as piping & instrumentation diagrams, electrical schematics, 3-D drawings, etc.), to another.

At the center of SmartPlant is **SmartPlant Foundation**, which provides the repository for data published by the authoring tools. SmartPlant components make the exchange of data from the authoring tools to SmartPlant Foundation and back possible.



Introduction of SmartPlant Modeling

SmartPlant Foundation (SPF) is based on a "hub-and-spoke" model. An SPF data repository is at the hub and each integrated software application is a spoke.



© 2005, Intergraph Corp.
All Rights Reserved.

SPF is a "loose integration" solution that allows each software application to maintain its own data model, independent of the SmartPlant Foundation data model. The benefit of "loose integration" is the flexibility to add or remove applications for any desired configuration of SPF.



Overview of SmartPlant

The design of today's SmartPlant provides the following features:

- Transfer of engineering data from one tool to another, eliminating the manual reentry of data
- Management of change resulting from ongoing engineering in upstream applications
- Accessibility of engineering information to other collaborators without requiring the original engineering tools
- Recording of change in data as it moves through the plant lifecycle
- Correlation of shared objects from multiple authoring tools. For example, the full definition of a pump may come from multiple disciplines (electrical, mechanical, and so on), and the data comes from different authoring tools.
- Support for engineering workflows, especially versioning, approval/release, and configuration control

© 2005, Intergraph Corp.
All Rights Reserved.



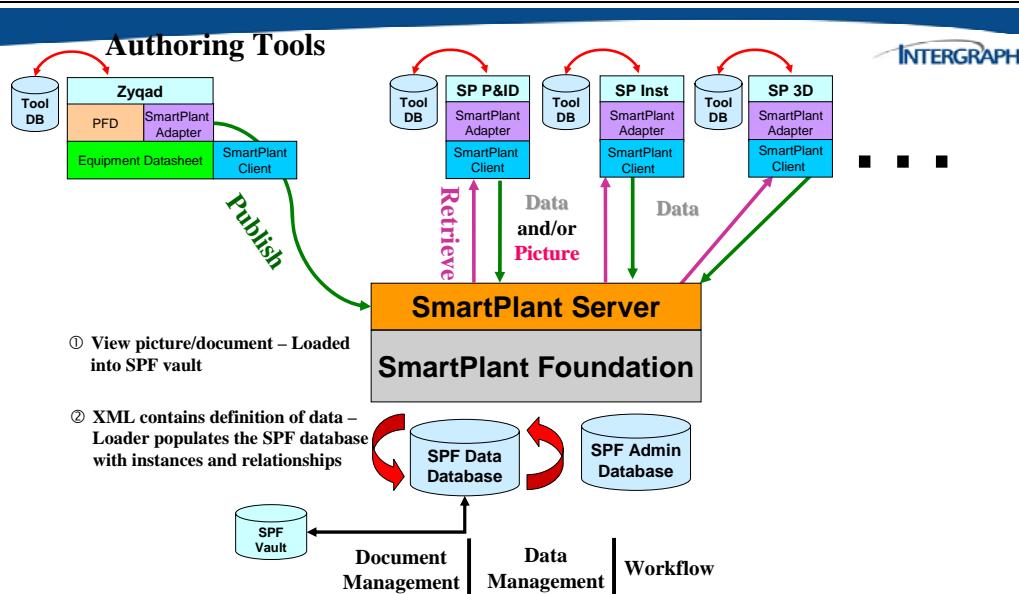
SmartPlant Architecture

SmartPlant Enterprise is functionality built on top of the **SmartPlant Foundation** database. Each tool that integrates with SmartPlant supports:

- The SmartPlant schema: A standard data definition designed to facilitate the flow of engineering information through its lifecycle. Each tool works with XML files that are representations of this schema.
- Commands to publish, retrieve, subscribe, unsubscribe and compare between the engineering tool and SmartPlant Foundation database.
- An adapter that allows communication with SmartPlant. Engineering tools Zyqad, SmartPlant P&ID, SmartPlant Electrical, SmartPlant Instrumentation, and SmartPlant 3D, and PDS integrate with SmartPlant by providing an adapter.

© 2005, Intergraph Corp.
All Rights Reserved.

The following graphic shows the architecture for SmartPlant. At the heart of the system is SmartPlant Foundation (SPF) and the SmartPlant Server. Each tool has an adapter (SmartPlant Adapter) that allows for communication between the tool and the underlying SmartPlant Foundation database and vault.



The SmartPlant Architecture

© 2005, Intergraph Corp.
All Rights Reserved.

1.1 SmartPlant Components

SmartPlant is comprised of the following components:

- SmartPlant Client** - An ActiveX .dll that allows the authoring tools to register with SmartPlant, connect to SmartPlant, and publish and retrieve data. After you install the SmartPlant Client on the client computer with an authoring tool and register the authoring tool project with SmartPlant, the SmartPlant Client is transparent to the user.
- SmartPlant Server** - Communicates with the SmartPlant Client, the SmartPlant Foundation ActiveX component, and the SmartPlant Loader to make publishing, retrieving, and comparing possible. The SmartPlant Server component is installed on the SmartPlant Foundation server.
- SmartPlant Loader** - Loads data published by the authoring tools into SmartPlant Foundation.
- Schema Component** – A suite of ActiveX components that provide functionality surrounding the creation, parsing, validation, and comparison of the SmartPlant schema and data.
- PDS Framework** - uses the PDS Framework Manager to load documents exported from PDS into the SmartPlant Foundation database.

1.1.1 The SmartPlant Client

The SmartPlant Client facilitates communication between the tool adapter and the SmartPlant Server. The SmartPlant Client is a set of components that provide the client-side services of SmartPlant. The SmartPlant common user interface is one part of the SmartPlant Client along with the data services that manage the communications between an application and the SmartPlant Server.



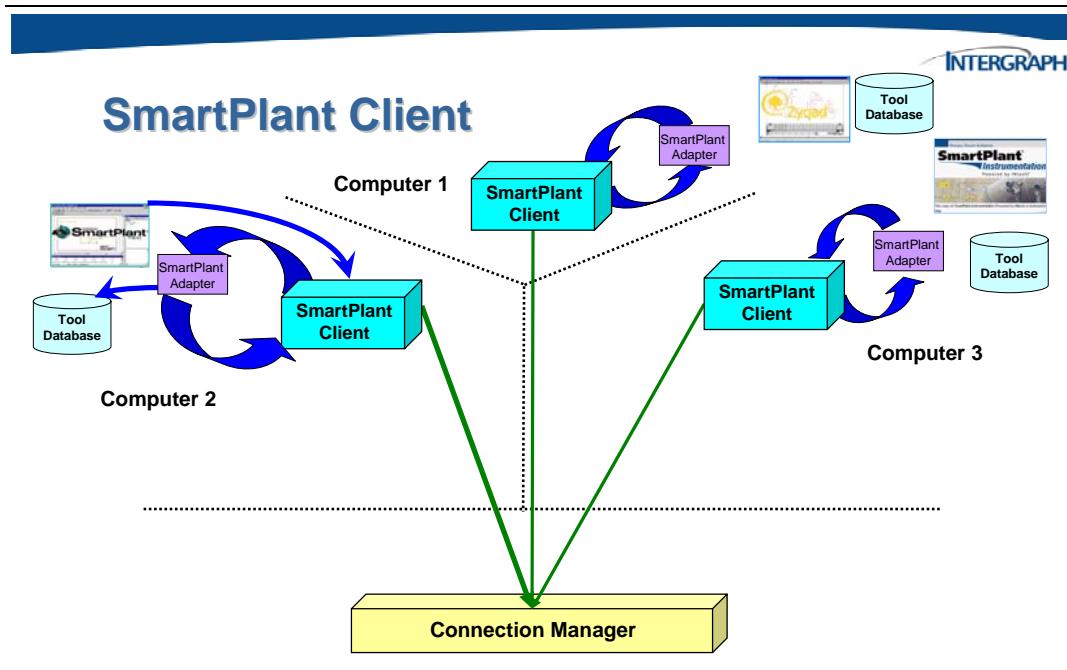
SmartPlant Client

The SmartPlant Client provides the methods for communication between the tool **adapter and the **SmartPlant Server**.**

The SmartPlant Client serves three roles:

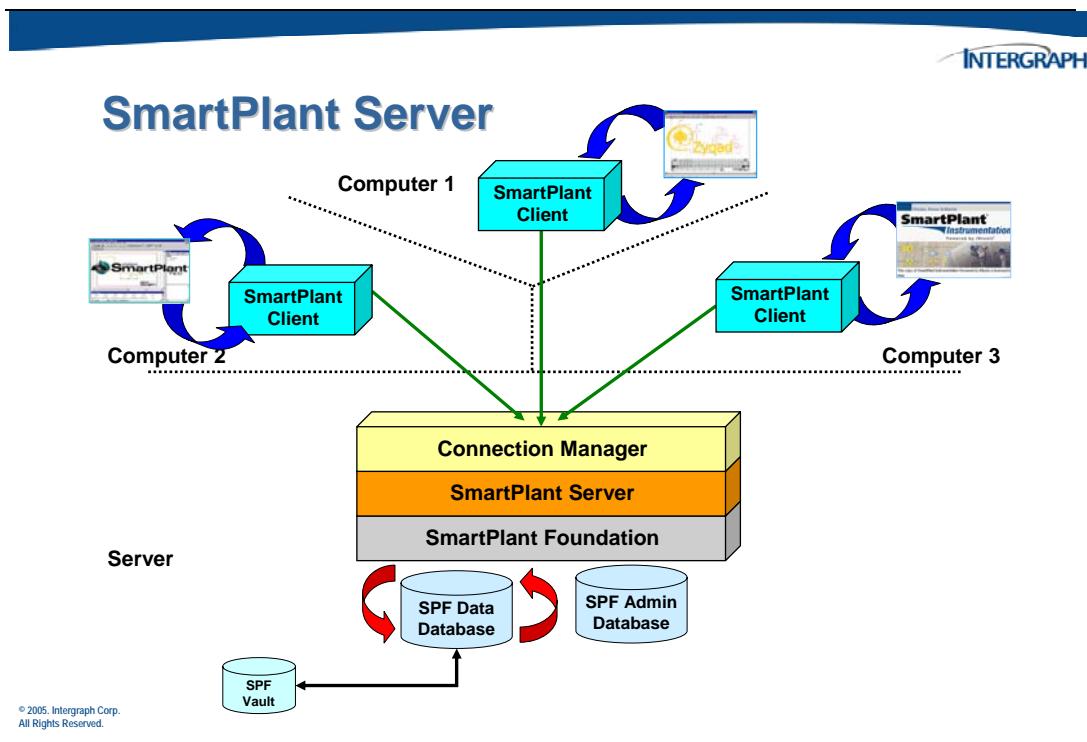
- Communicates with the SmartPlant Server and returns the results of that communication back to the tool adapter**
- Calls interface methods on the tool adapter to perform functions associated with the integration process**
- Provides common user interface components that tools may use to present a consistent UI across all SmartPlant-enabled applications**

Common tools such as Zygad, SmartPlant P&ID, SmartPlant Instrumentation, PDS, and SmartPlant 3D as well as SmartPlant Electrical and MARIAN™ use the **SmartPlant Client**, the **SmartPlant Adapter** and default configuration included in order to communicate with SPF.



1.1.2 The SmartPlant Server

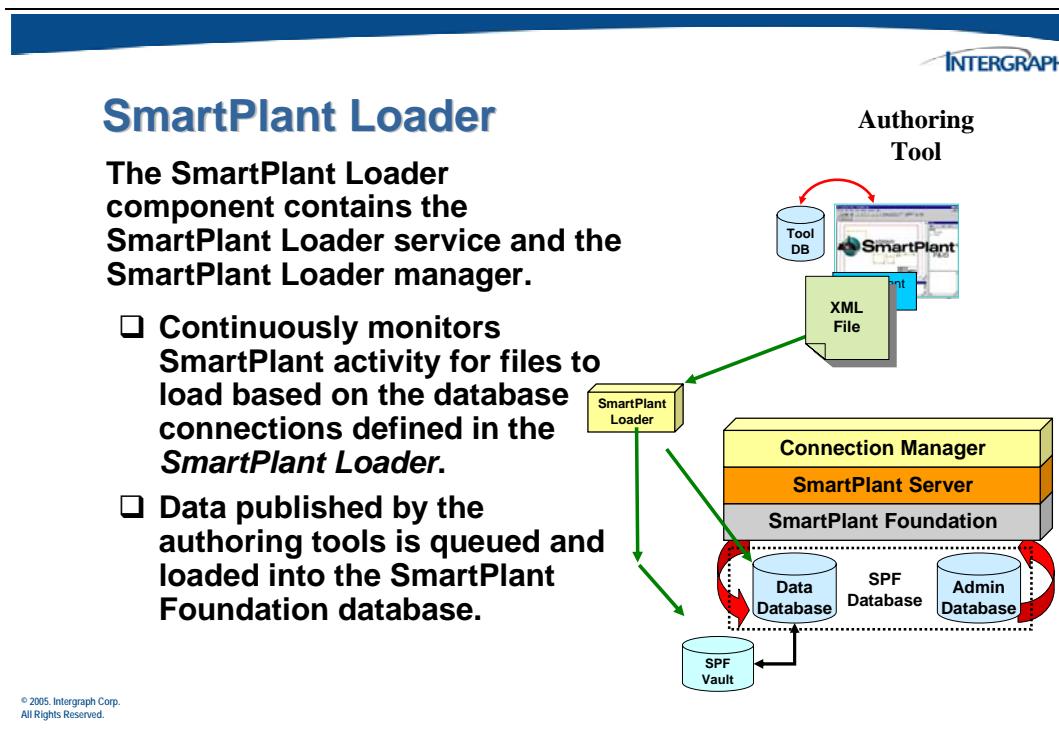
The SmartPlant Server component is a software layer on the SmartPlant Foundation server. The SmartPlant Server takes requests from the SmartPlant Client component and communicates with SPF.



The SmartPlant Server resides on a server computer, processing requests from various SmartPlant Clients and returning the results of those requests to the SmartPlant Clients. The SmartPlant Server issues database requests to retrieve and update the information within the SmartPlant Foundation database.

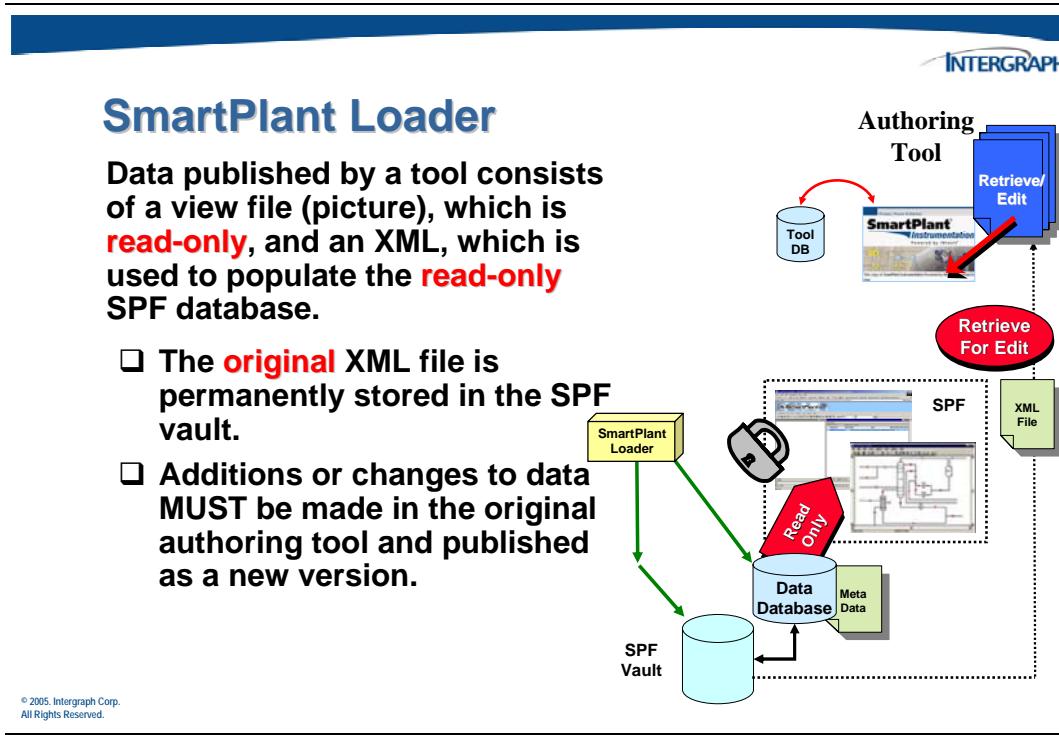
1.1.3 The SmartPlant Loader

The SmartPlant Loader loads data published by the authoring tools into the SmartPlant Foundation database. The SmartPlant Loader continuously monitors SmartPlant activity for files to load and should always be running in the background. You are not required to install the SmartPlant Loader on the SmartPlant Foundation server. However, if you install the SmartPlant Loader on another server, you must install SmartPlant Server or SmartPlant Client and .NET Framework, Oracle Client, MSXML, and MDAC before you install the SmartPlant Loader.



Data created in an authoring tool is **published** in the form of an XML file via the adapter and the SmartPlant Server to the SmartPlant Foundation database.

Data published by a tool consists of a view file (picture), which is **read-only**, and XML, which is used to populate the SPF database. In addition, the XML file is permanently stored in the SPF vault.



A different tool can then retrieve that same data for the next phase of design in the engineering workflow.

1.2 The SmartPlant Schema

One of the key elements to full integration is to have a "schema", and enforce some kind of rules between a "publisher" and a "retriever" of data. It should be simple for two different systems to talk about something that they have in common, e.g., a motor, a pump, a valve, etc. But it is really not that simple because each system has its own idea of what things are, and how they are used. A "schema" is simply another way of saying "a diagrammatic representation, an outline or a model."

The SmartPlant schema describes the structure of data passed through SmartPlant along with its rules.



SmartPlant Schema

The SmartPlant schema, effectively SPF data model, provides the structure and semantics of the data that can be published to, and retrieved from, the SPF data repository.

**SmartPlant Foundation
Data Model**





SmartPlant Schema

Models are used to convey the essence of what you're talking about.



A model of a car isn't something that you really get into and drive to the office, but it is "real-enough" that you make the mental leap that it represents a car strongly enough that you can actually "pretend" to drive the car to work.

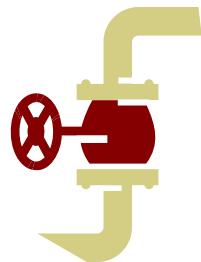
© 2005, Intergraph Corp.
All Rights Reserved.

If you think about the problem of integrating data from a lot of different software applications, you can understand how a "model" of the data that is being passed around might be a good way to simplify the discussion of "how do I get that valve from a P&ID (piping diagram) into my 3-D program?"



SmartPlant Schema

A "model of a valve" is not a valve, but it can expose many aspects of a valve, e.g., its flow-rate, its weight, its inlet size, etc.



Some other program may not care about any of that, and just wants to talk about the height of the valve, or about who carries the valve "in-stock", so that it can be ordered quickly.

© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Schema

Each SmartPlant Enterprise software program or "tool" has some conceptual model of a valve. When we say "tool", we're talking about:

- SmartPlant Foundation** for total information management for the life of your plant.
- SmartPlant 3D** our next generation, data-centric, rule-driven engineering design solution.
- SmartPlant Electrical** an application designed by electrical engineers for electrical engineers.
- SmartPlant Instrumentation** the industry-leading instrumentation solution.

© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Schema

SmartPlant Enterprise tools (con't)

- SmartPlant P&ID** for generating "intelligent" P&IDs and management of related engineering data for use throughout a plant's life.
- SmartPlant Markup** a precision viewing and markup tool.
- SmartPlant Review** for dynamically viewing 3D computer models for in-depth review and analysis.

No one tool cares about "all" the possible attributes of a valve.

© 2005, Intergraph Corp.
All Rights Reserved.

So how does the data about "this particular valve" get "shared" between tools? Make a "model of a valve", and share the model between the tools!



SmartPlant Schema

A "data model" is an abstract representation of the objects that software tools can share, use, and think about.

It makes it easy for each tool, because they can still think about "a valve" in their own way.

It makes it easy for all tools because they can share the concept of "this particular valve" by agreeing on what the data model for a valve is.

© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Schema

The SmartPlant schema defines the structure of the SmartPlant Foundation (SPF) database, and is used as the neutral format through which authoring tools publish and retrieve data, independent of their specific data model.

Other functions of the SmartPlant schema include:

- Enumerated lists:** engineering is full of lists of valid data that can be applied to data values (and shared between applications), i.e. types of valves, fluid codes, etc.
- Connectivity -** engineering is as much about the relationships between equipment as the equipment itself, i.e. a vessel has a nozzles that connects to different pipelines.

© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Schema

SmartPlant schema functions (con't)

- User-defined properties - Software applications provide the means for users to specify user-defined variables. Procedures must be established to assure that this data is implemented in the SmartPlant schema.**
- Support for Units-of-Measure, to allow clear flow of data regardless of the units specified.**
- Rules for evolving the schema, so changes to the structure and semantics of the data can be made due to new or changing requirements.**
- Specific support for engineering concepts such addressing process data, relationships between equipment and nozzles, and so on.**

© 2005, Intergraph Corp.
All Rights Reserved.

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 (data) 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.

The SmartPlant schema can be hard to understand; to make it easier to interpret, the **Schema Component** exists. The Schema Component is a set of .DLLs that assists the tools with the generation and subsequent parsing of the XML data. The tool adapter interfaces with the Schema Component (the main interface point) to read the SmartPlant schema. The SmartPlant schema is covered in more detail in Chapter 3.



SmartPlant Schema

Because SmartPlant is intended to facilitate data exchange from multiple engineering tools, the following rules apply:

- All schema data (data definitions) is defined as part of the SmartPlant schema
- The SmartPlant schema describes everything that goes on in SmartPlant
- A copy of the SmartPlant schema (an XML file) resides on the server(s)
- Component schemas are selective extracts from the SmartPlant schema, and reside on every client
- All changes are made to the SmartPlant schema and then propagated to the component schemas

© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Schema

The SmartPlant schema is defined by an XML file called **EFSchema.xml** (default).

EFSchema.xml

- Contains definitions of:
- Classes
 - Interfaces
 - Properties (attributes)
 - Relationships
 - Views
 - Picklists
 - Units of Measurement
 - Navigation Paths
 - Shared Objects (classes)

© 2005, Intergraph Corp.
All Rights Reserved.

The building blocks of the schema are classes, interfaces, and relationships. Classes are typically real world objects like instruments, tanks (vessels) or pipe runs.

1.2.1 Introduction to Class Definitions

Humans seem to be able to naturally classify things. If you put a big handful of coins down in front of most people, their first instinct is to sort them into groups - pennies, nickels, dimes, etc., going into separate groups. It's just easier to think about things that are somehow grouped properly.

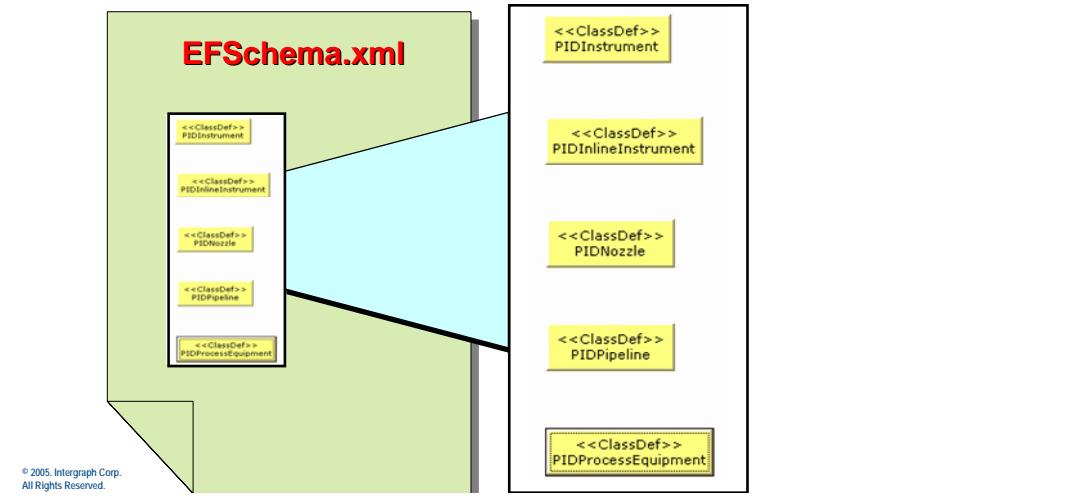
The building blocks of the SmartPlant schema are **classes**, **interfaces**, and **relationships**:

- Classes* typically represent real-world objects like instruments, equipment or pipe runs and are defined by Class Definitions or *ClassDef*'s.
- Interfaces* are used to tightly bind “roles” and are defined by Interface Definitions or *InterfaceDef*'s.
- Relationships* represent the association between two objects and are defined by Relationship Definitions or *RelDef*'s.



Class Definitions

Class definitions define physical or logical types of object data.

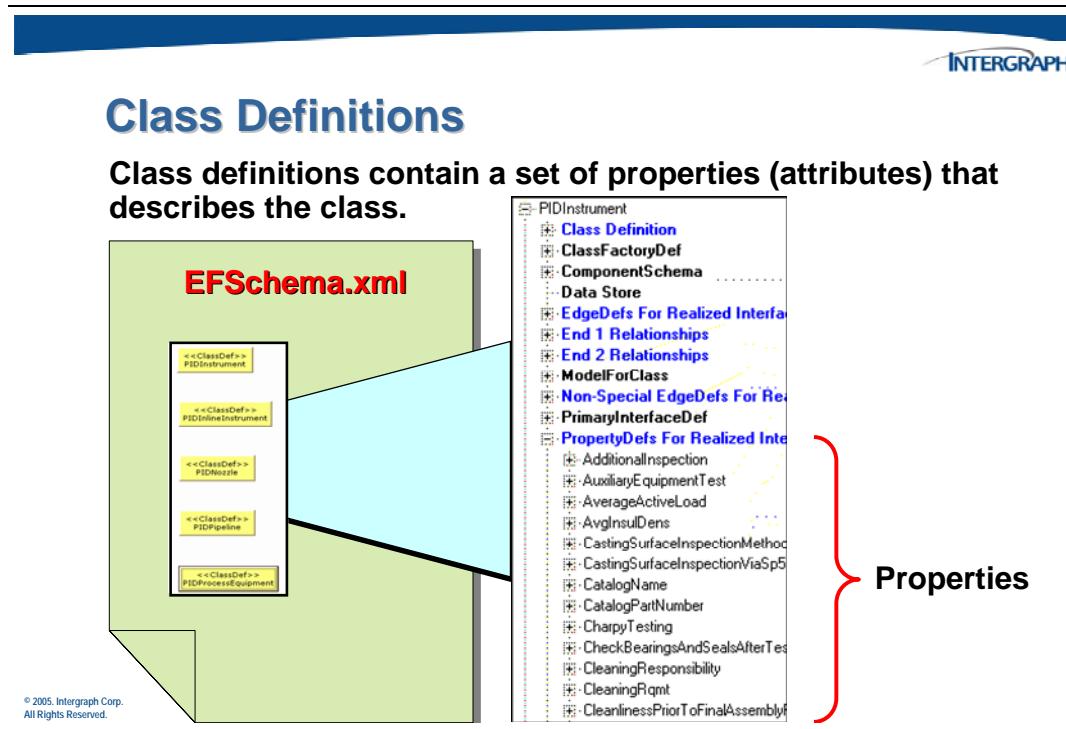


A ClassDef is the SmartPlant schema element that will eventually become something important in the data that is “published” and “retrieved” from your software tool.

To be more precise, a ClassDef can represent physical things, such as pumps, or conceptual things, such as projects. Instances of classes become “objects” within a container (XML) that gets published by a software program, during the “publish” process. For example, an instance of a class that was defined as a ClassDef named

ELEWire can be published, and is expected to contain data that is the electrical system's idea about a wire.

The ClassDef also acts as a container to group a set of properties or characteristics that is used to describe the object class.



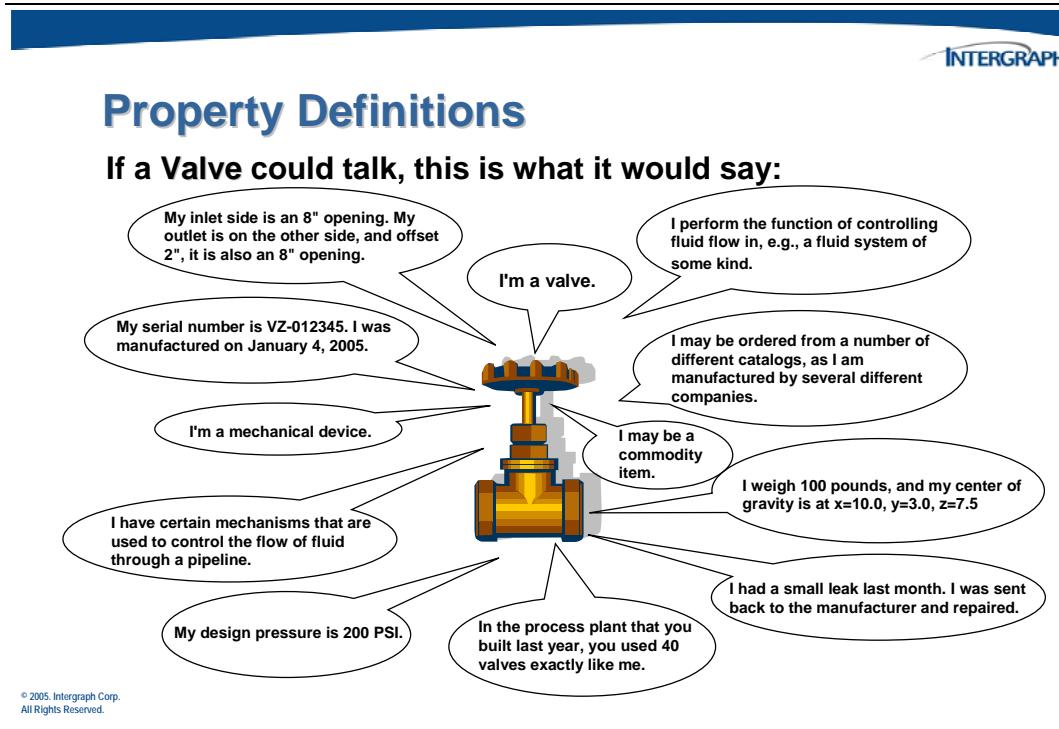
In the schema, class definitions are a named description of a set of objects that support or realize the same interface definitions and share the same property definitions and relationships.

Class definitions realize interface definitions. For example, the **EQDProcessVessel** class definition realizes the **IProcessVessel** interface definition. The **IProcessVessel** interface definition exposes property definitions, such as *ProcessVesselNormalLiquidVolume*.

1.2.2 Introduction to Property Definitions

When different people talk about a "valve", what should come to mind? The problem is: everyone who thinks about a "valve" thinks about in his/her own terms. In fact, there is no one "correct" way to think about a valve, and that's OK.

How can we all think different things about a "valve", and come to some conclusions and agreements before we have fluid flowing through it? Let's try a different approach: instead of us guessing at the nomenclature of a valve, how about if we let the valve "tell us what it is", and we'll just listen carefully.

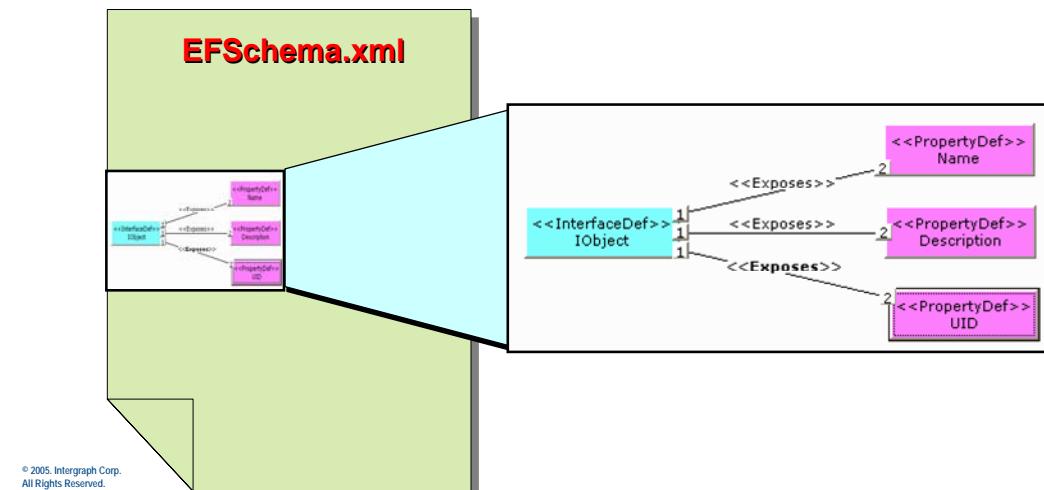


Property Definitions (*PropertyDef's*) are used to provide identification and naming of all object classes. *PropertyDef's* allow you to describe the class with respect to its position in the hierarchy and provide inheritance of information to other classes. All property definitions for a class are exposed through **interface definitions** and never directly by the class.



Property Definitions

A property is a characteristic or attribute used to describe a class.



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*.

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

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

Property information can be used to determine the property type or, in other words, the possible values for that property definition. Standard schema property types include *Boolean*, *integer*, *double*, and *string*.

Property definitions of the enumerated property type (picklist) have a list of possible string property values defined for them. A property definition of this type must match an entry in the list of enumerated property values defined for the property type.

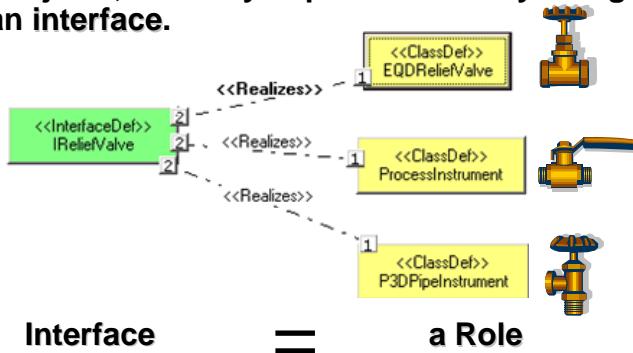
1.2.3 Introduction to Interface Definitions

The "valve" exposes certain "roles" to the universe, and the roles that a valve object "realizes" are there for us to use, if we want to listen and understand. This is called a "role-based" approach to classifying things - let them tell what they are (meaning what roles the object thinks it fulfills).



Interface Definitions

Microsoft has been a leading proponent of the "role-based" view of objects, and they implement this by using an artifact called an interface.



The use of interfaces is common in Microsoft .NET technology.

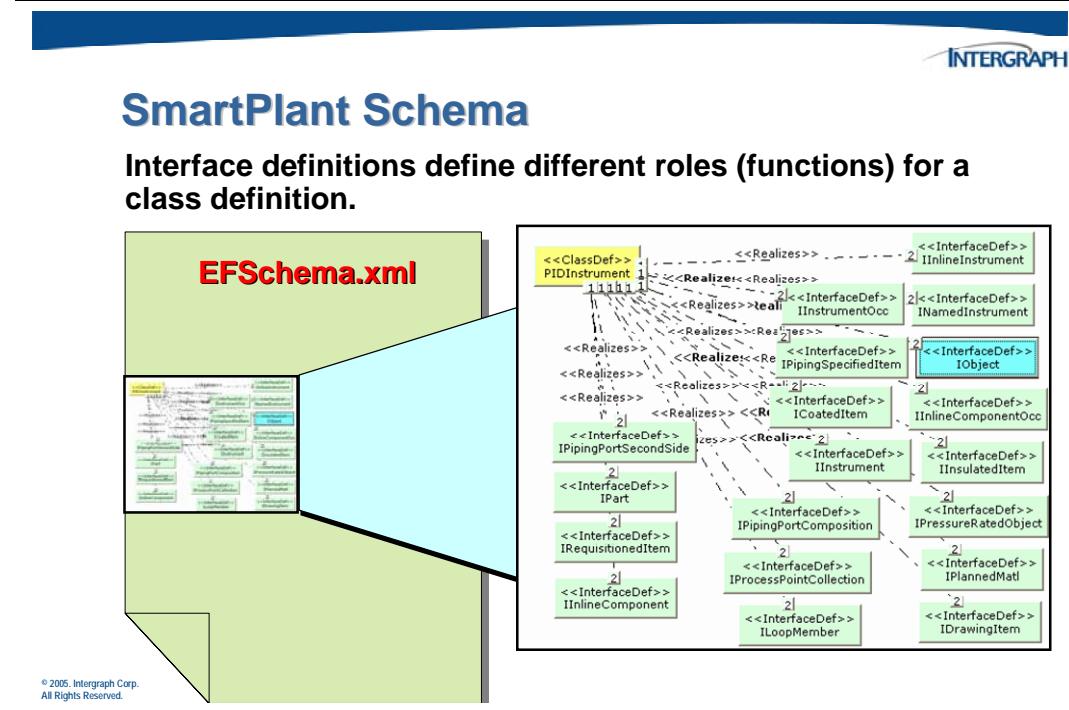
© 2005, Intergraph Corp.
All Rights Reserved.

Interfaces are just convenient "places" to group the tightly-bound properties of a role:

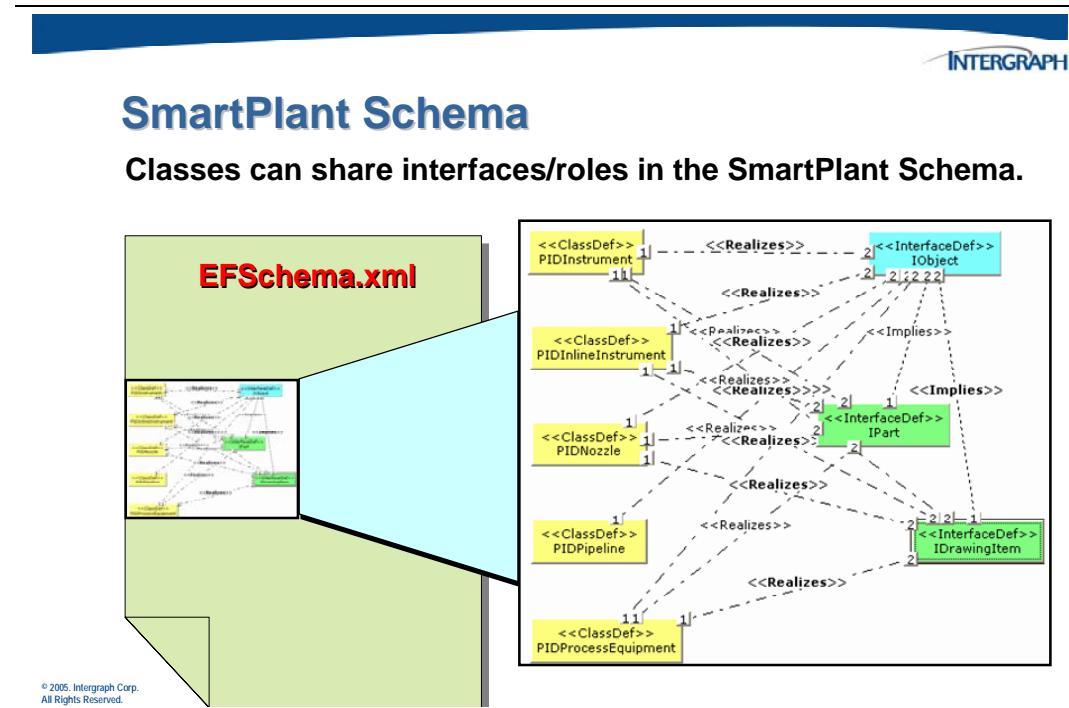
- A role is something that an object yields up to the universe, i.e., "I am a driver, here is my 'driver-ness'."
- A role consists of closely-related properties, such as driver's license number, expiration date, restrictions, picture.
- Properties that you purposefully keep together, such as on a driver's license, in your wallet, in your purse, in your car.
- Because a role makes data retrieval easier, and less ambiguous.

If you want to get more technical - an interface could be thought of as a named table of property definitions that are role-based.

Thus an interface definition is a named collection of property definitions. Interface definitions expose the property definitions for class definitions. Each interface definition represents a "role" for a class definition.



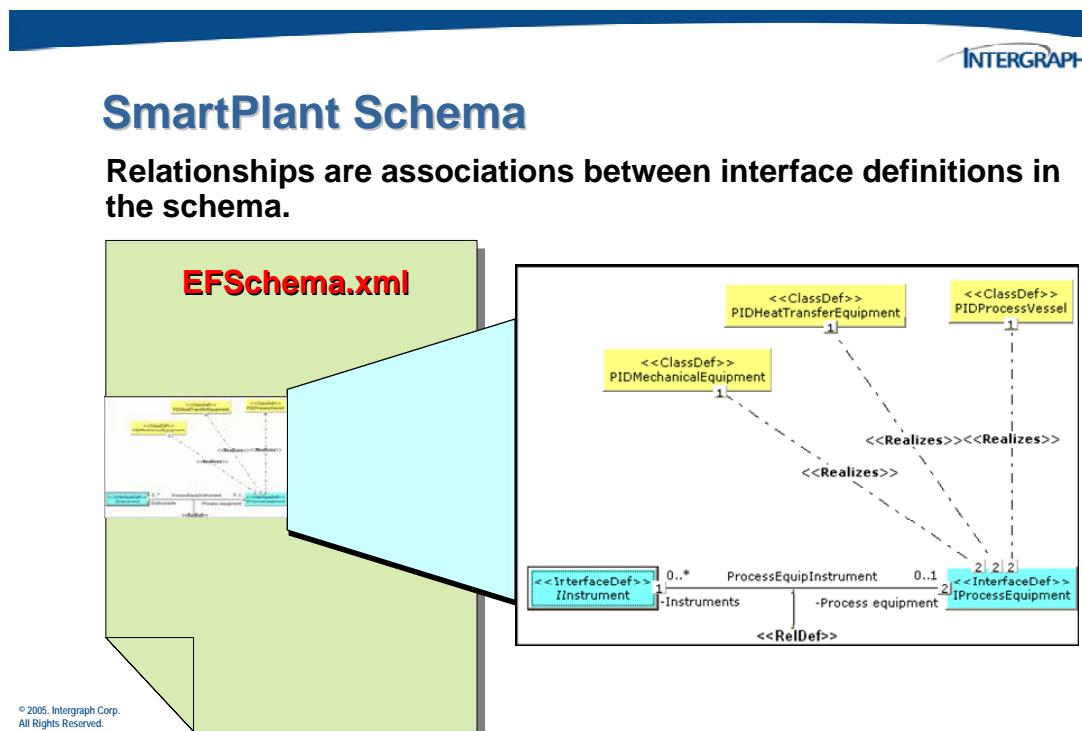
By sharing specific interface definitions, class definitions can also share property definitions, but not the data associated with the properties.



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.

1.2.4 Relationships

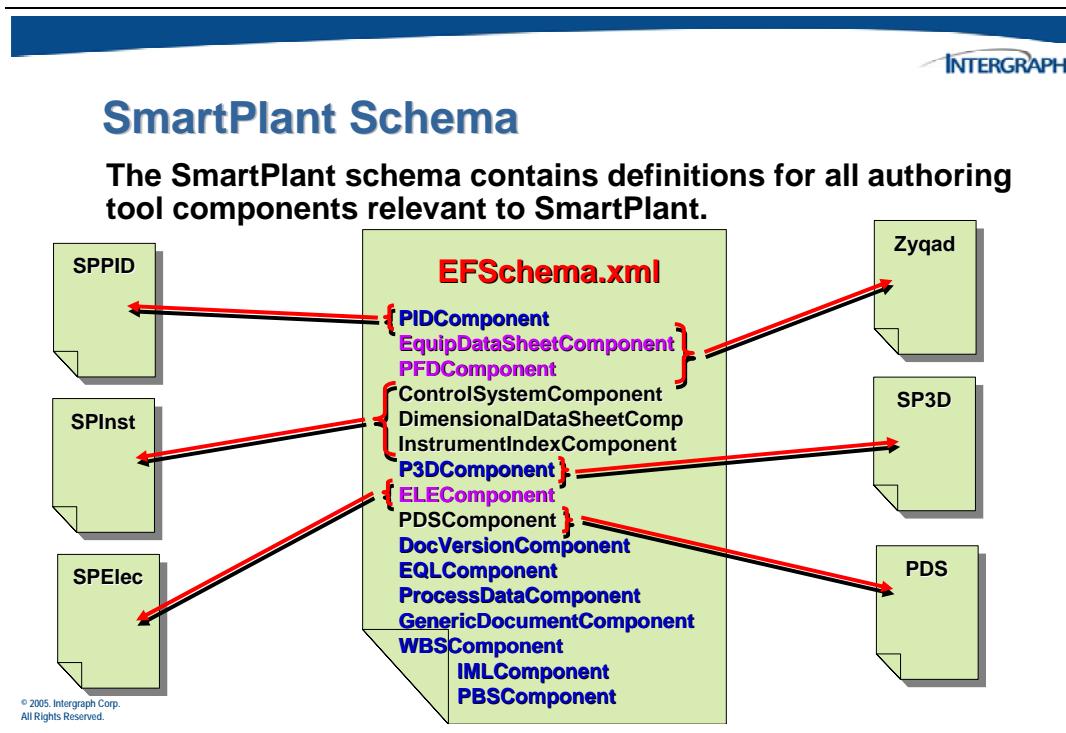
Relationships are associations between interface definitions in the schema. They identify two specific objects that are related by a specific relationship type. The relationship type identifies the interface definitions on the two objects that fulfill the roles associated with the relationship type.



The relationship type also identifies the roles defined at both ends of the relationship. In the schema, relationship definitions are defined between interface definitions, not between class definitions. A relationship can only exist between objects that support (realize) the interface definitions at each end of the relationship definition.

1.3 Authoring Tool Schemas

The engineering application tool user, i.e. the SmartPlant Instrumentation (SPI) customer, does not need to understand and comprehend the SmartPlant schema to use SPF. In fact, they are able to conduct their workflows without concern for database, data model, and underlying architecture. The SmartPlant schema assures that data will be stored and retrieved in a proper fashion in accordance with the defined structure and rules.



The SmartPlant schema will then encompass the items and relationships for all the authoring tools in order to facilitate the transfer of information from one tool to another. This helps reduce the need to recreate data from one tool to the next.

The following defines the component schemas that encompass SPF:

- PBS** stands for Plant Breakdown Structure. Plant breakdown is created in SPF and retrieved into the tools to create an identical structure.
- WBS** stands for Work Breakdown Structure, and like PBS, the objects are created and published in SPF and retrieved by the tools.
- The **GenericDocumentComponent** contains the Document class and represents a very generic document.
- The **DocVersionComponent** schema is all that is required to define the data that appears in the metadata container for publish and retrieve.
- EQLComponent** is the component schema required to support an Equipment List application.

- ProcessDataComponent** is a component schema defining the Instrument Process Datasheet integration between SPF and SPI.
- IMLComponent** is the component schema to support the Instrument Master List application which functions as an instrument index.

A tool schema file, also known as a map file or a tool map file, describes the structure of data as it is defined in the authoring tool database and how the authoring tool classes and properties map into and out of the SmartPlant schema.



Authoring Tool Schemas

Tool map schemas or "Map files", describe the mapping between an authoring tool's internal data model and the SmartPlant schema.

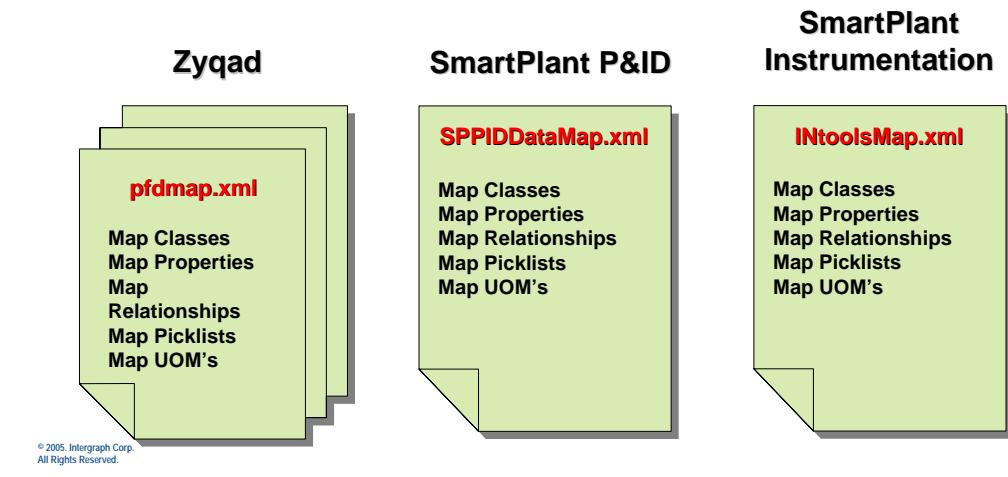
When a document* is published, the authoring tool adapter, with the help of the SmartPlant schema component API uses the tool map schema to convert the data from the tool's internal data model to the SmartPlant schema.

Note: * The word document is "overloaded" with meaning by everyone. For our purposes, we're talking about an XML file that contains data, under control of a schema.



Authoring Tool Schemas

Each authoring tool is responsible for delivering a tool schema which contains the definitions specific to that tool.



Some authoring tools such as SmartPlant P&ID will have one tool schema XML file while other applications such as Zyqad has several tool schemas; eqdmap.xml, pbsmap.xml, pfdmap.xml, pidmap.xml, and wbsmap.xml.



Authoring Tool Adapters

Each authoring tool that is part of SmartPlant Enterprise has an authoring tool "Adapter", which facilitates the sharing of data between the authoring tool and SmartPlant Enterprise.

Tool adapters generate XML files for "publish" operations, and consume XML files for "retrieve" operations.

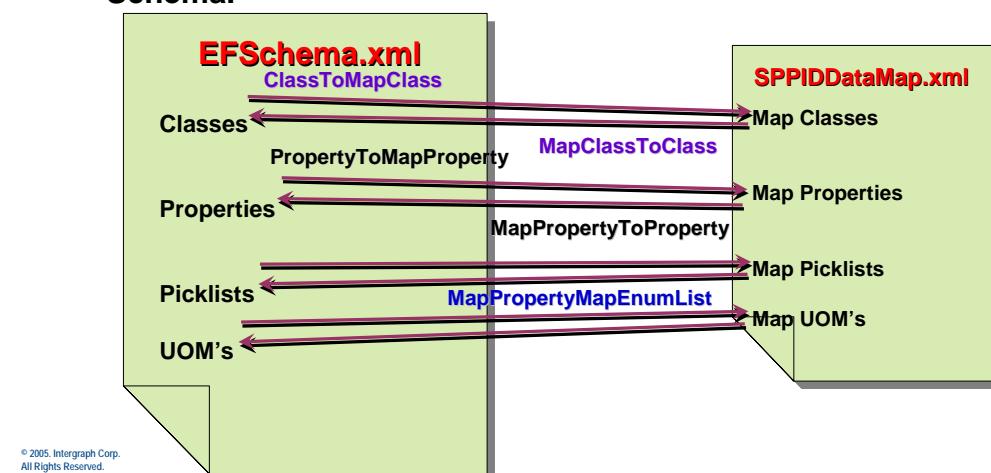
1.4 Schema Mapping

Each authoring tool that uses configurable mapping has its own schema called a tool schema. In order to make data publishing and retrieval easier, mapping is done between the SmartPlant schema and the tool schema.



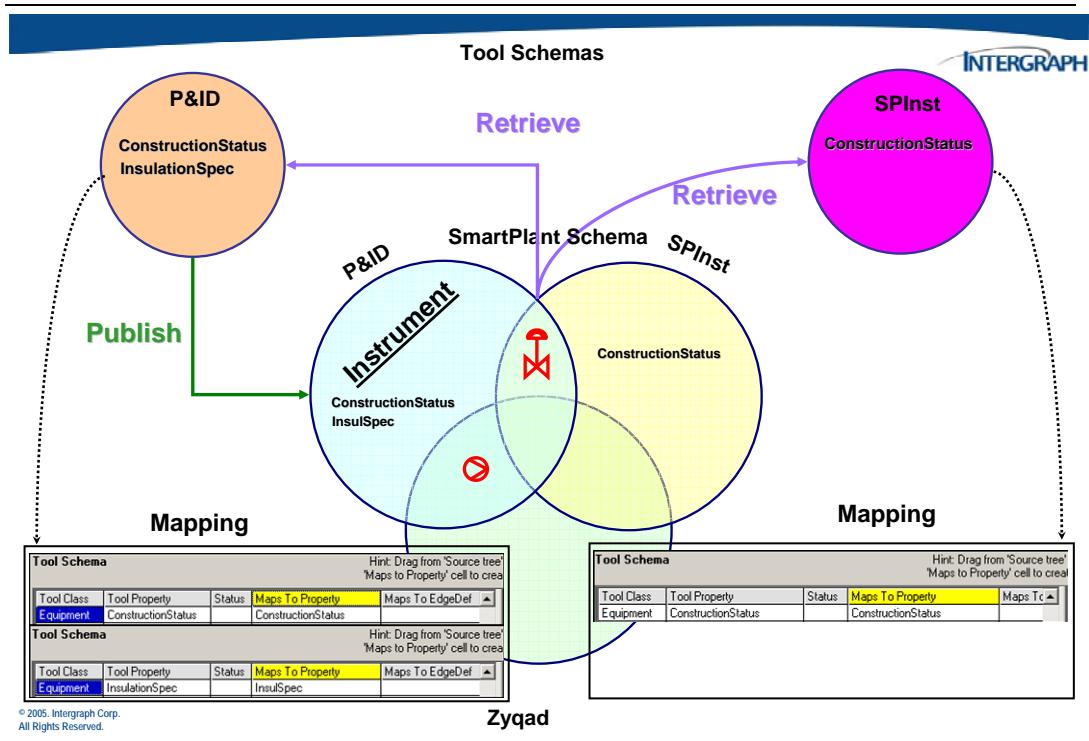
Schema Mapping

Every class in the tool schema that is to be published to SmartPlant maps to a **class, **property**, or **picklist** in the EF Schema.**



Integration with SmartPlant is done in a modular fashion with minimal impact to the application itself. Each authoring tool delivers an adapter that supports the key functionality to publish and retrieve data to SmartPlant. For tools that support configurable mapping, a **map** must be defined between the applications internal data structures and the SmartPlant schema. The Schema Component can be used to help generate the necessary XML files to exchange data and assist with many of the integration operations.

Mapping is not required because there is some hard coding done in the tool adapter so that the adapter can publish default data. However, without mapping, data retrieval can be a big problem. If mapping is done correctly, a tool will be able to retrieve data from other tools.



If a new property is added, it needs to be mapped. Each document/container that is published has an associated component schema that describes the contents that are being published. Therefore, each document type corresponds to a component schema.

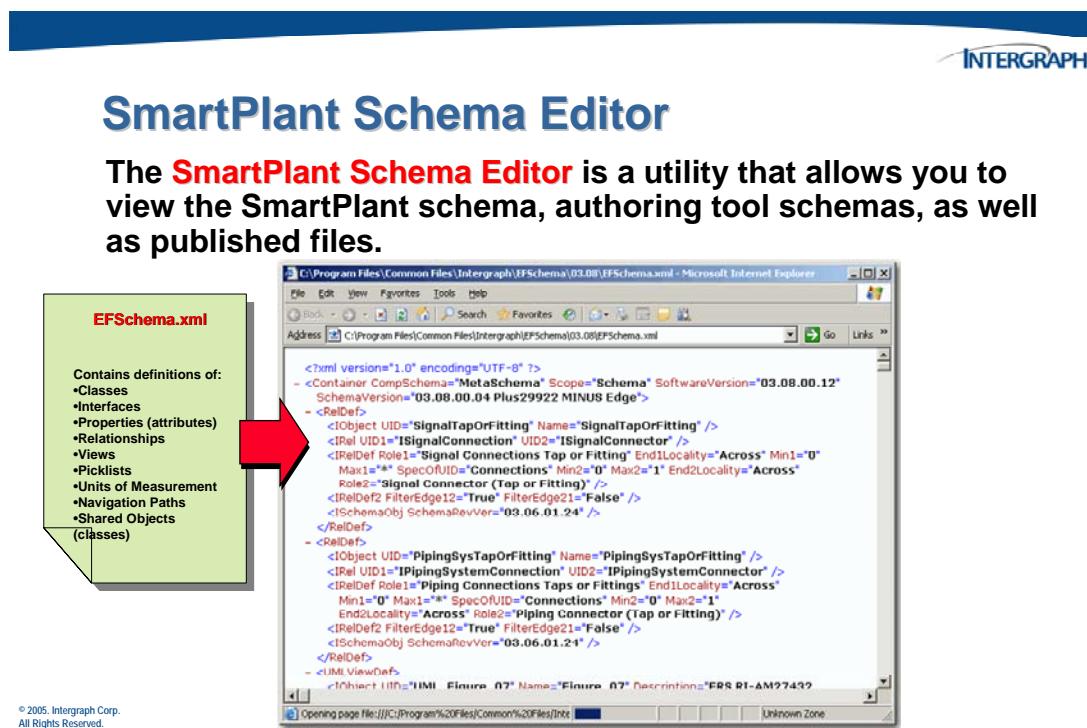
A particular tool may publish documents of one type or of multiple types. Therefore, a particular tool may use a single component or multiple components.

SmartPlant only cares about components and not about tools. No correlation exists as to what tool publishes which document types.

While tools will typically span the different components associated with a particular discipline (for example an instrumentation tool will span the various instrument and wiring document types), no rules exist that require this. As long as a tool can publish documents of the document type for at least one component, it can be a contributor to SmartPlant.

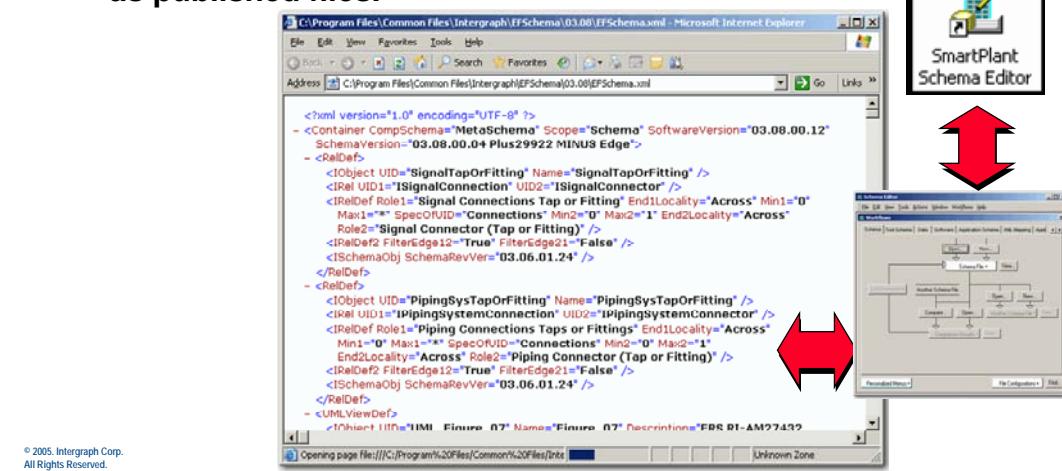
1.5 Introduction to the Schema Editor

The **SmartPlant Schema Editor** is a utility that allows you to view the SmartPlant schema, the meta schema, authoring tool schemas, and data files. This utility is especially useful for familiarizing yourself with the schema and its class definitions, interface definitions, relationship definitions, and properties.



SmartPlant Schema Editor

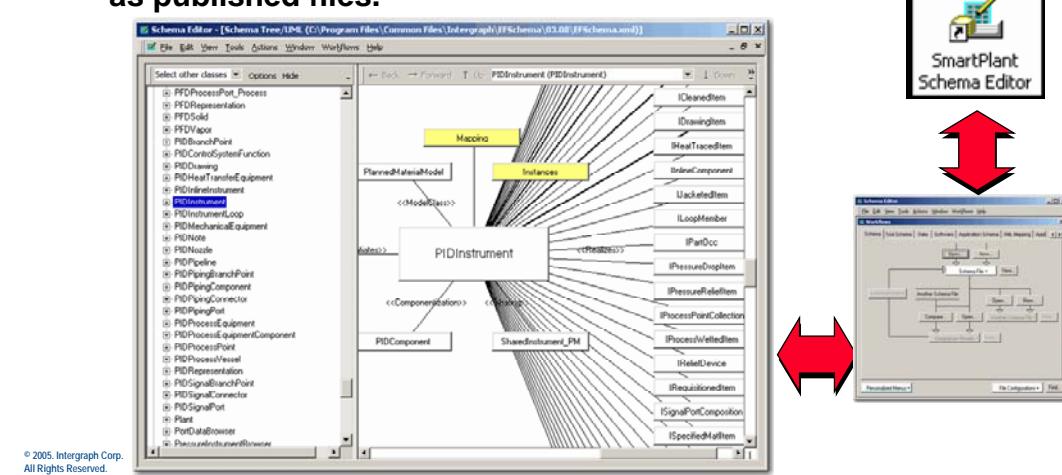
The **SmartPlant Schema Editor** is a utility that allows you to view the SmartPlant schema, authoring tool schemas, as well as published files.



© 2005. Intergraph Corp.
All Rights Reserved.

SmartPlant Schema Editor

The **SmartPlant Schema Editor** is a utility that allows you to view the SmartPlant schema, authoring tool schemas, as well as published files.



© 2005. Intergraph Corp.
All Rights Reserved.

Software developers can also use the Schema Editor to modify schemas, tool schemas, and data files. However, you cannot edit objects and relationships in the meta schema.



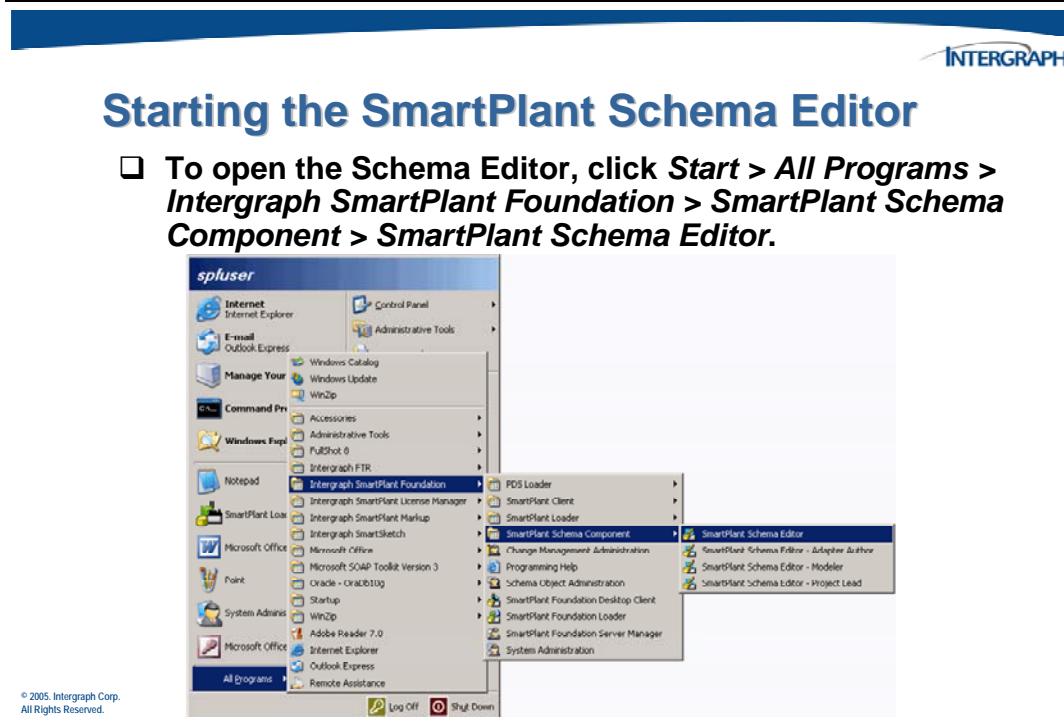
SmartPlant Schema Editor

The SmartPlant Schema Editor provides a mechanism for advanced users, usually software engineers, to perform the following types of tasks, among others:

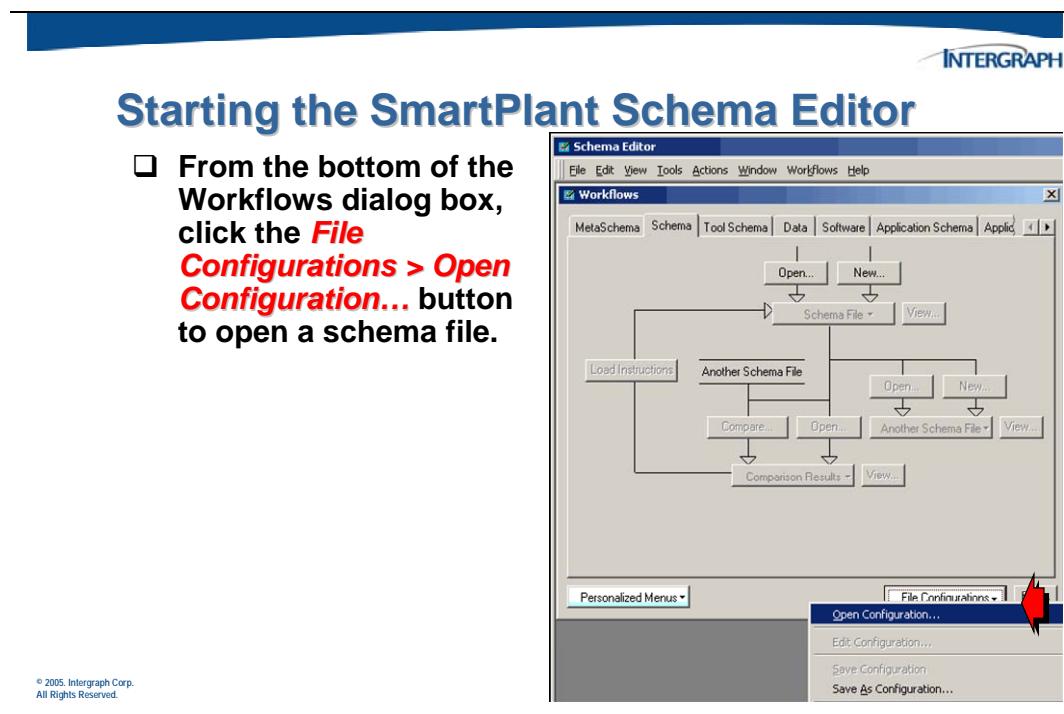
- View the schema, meta schema, data files, and authoring tool schemas in a variety of ways**
- View mapping between the tool schemas and the SmartPlant schema**
- Compare schemas, data files, and tool schemas and view comparison instructions**
- Add new objects and relationships to the schema, tool schemas, and data files**
- Edit existing objects and relationships in the schema, tool schemas, and data files**
- Create new data files for testing**

1.5.1 Starting the Schema Editor

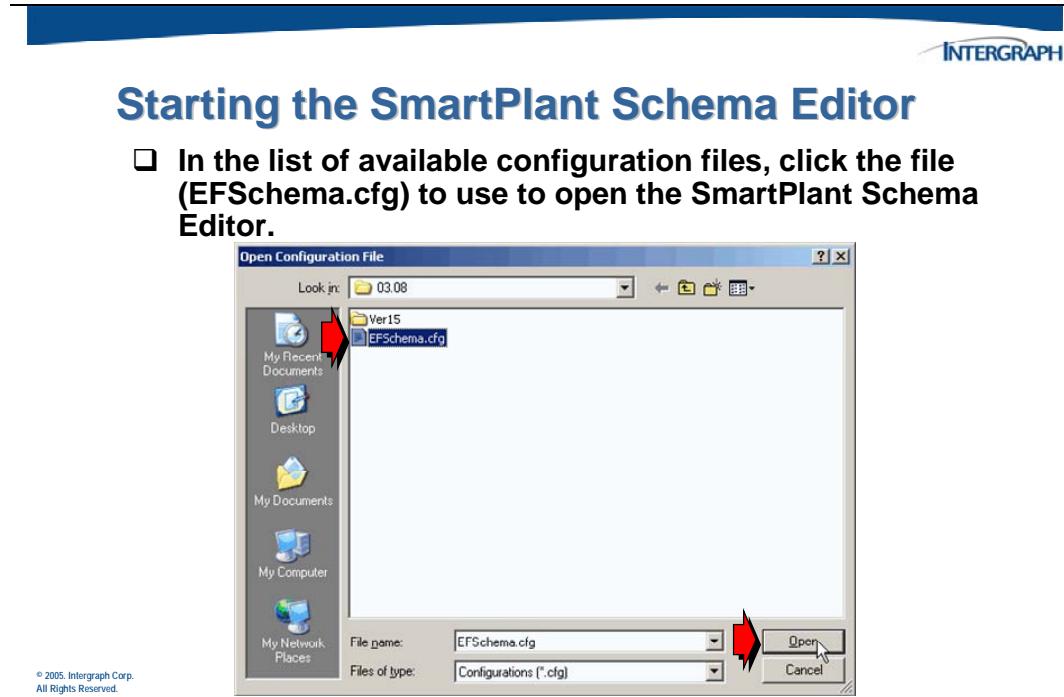
The *Schema Editor* is installed as part of the Schema Component installation. There are a variety of ways to do operations using the Schema Editor.



The first view that displays is the **Workflow** view.

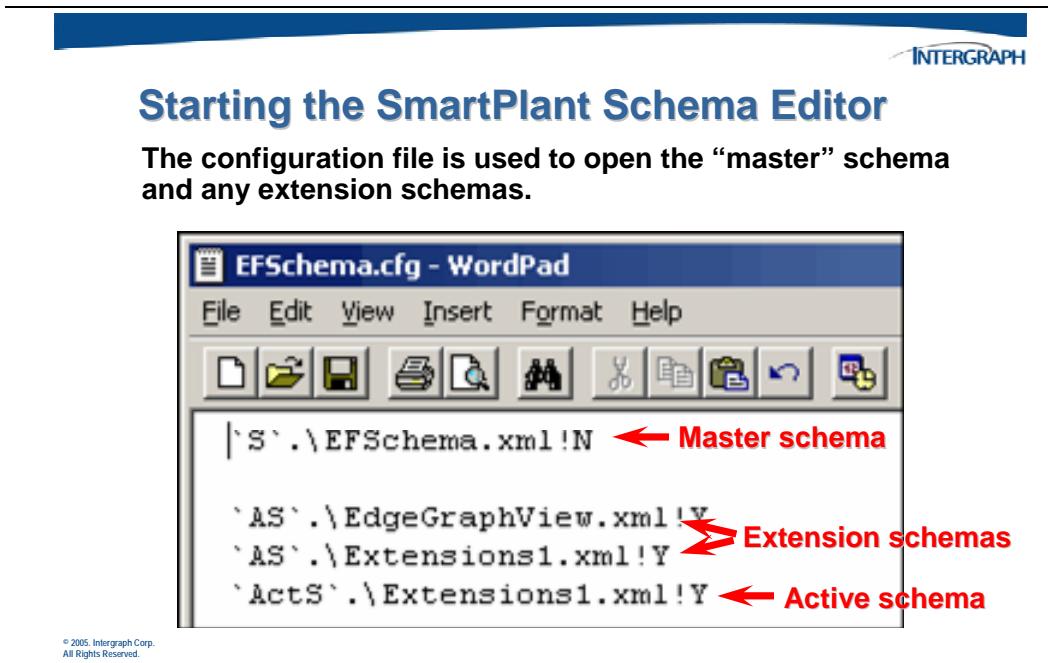


The **Workflow** view presents the Schema Editor functionality in a sequence that you must follow when performing operations using the Schema Editor.

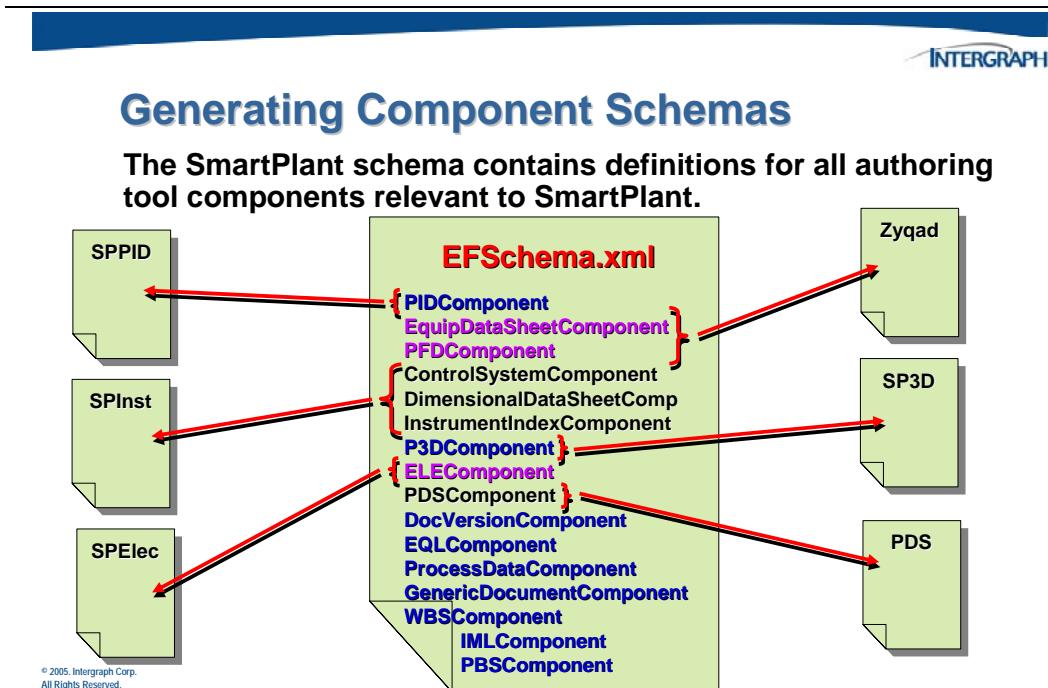


When you start the Schema Editor and open a configuration/schema file, the meta schema is already open.

The **EFSchema.xml** file is the delivered schema containing all of the SmartPlant definitions. Extension schemas are used to add additional definitions to SmartPlant.



Component Schemas are subsets of data definitions from the master schema (EFSchema.xml). The component schemas are typically generated during the installation of the Schema Component.

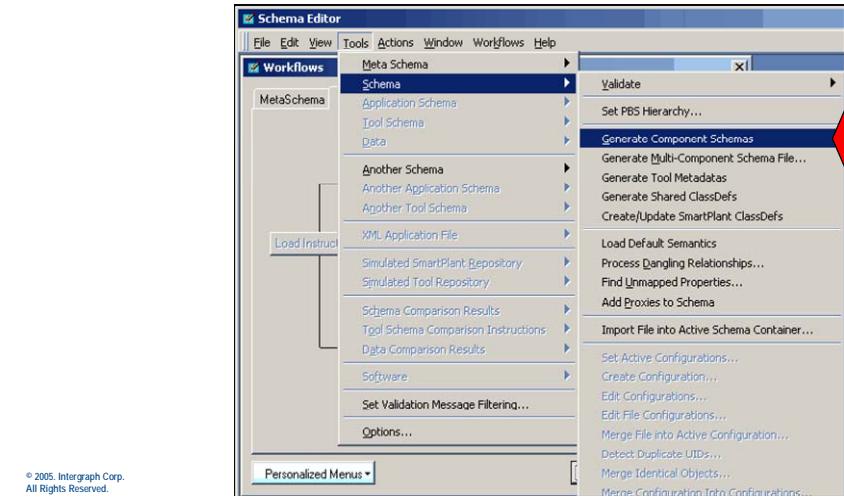


If you change/extend the SmartPlant schema with custom classes/properties, you will need to generate new component schemas in order to have the customizations available in the component schema.



Generating Component Schemas

- ❑ Click **Tools > Schema > Generate Component Schemas** on the Schema Editor menu.

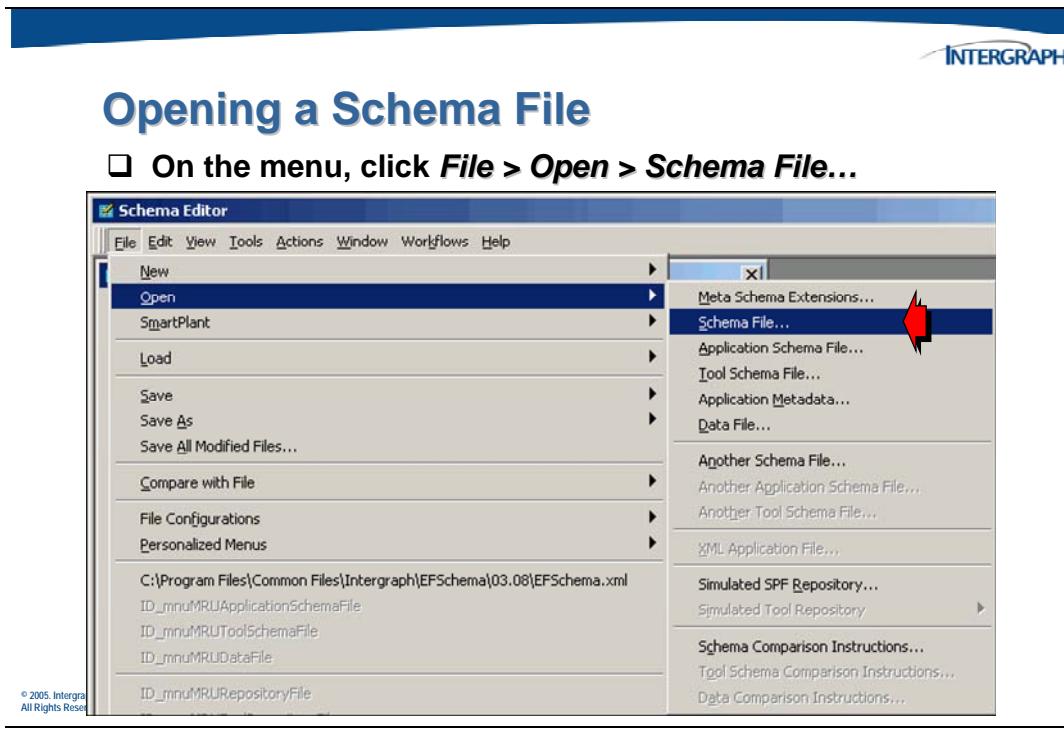


Generating Component Schemas

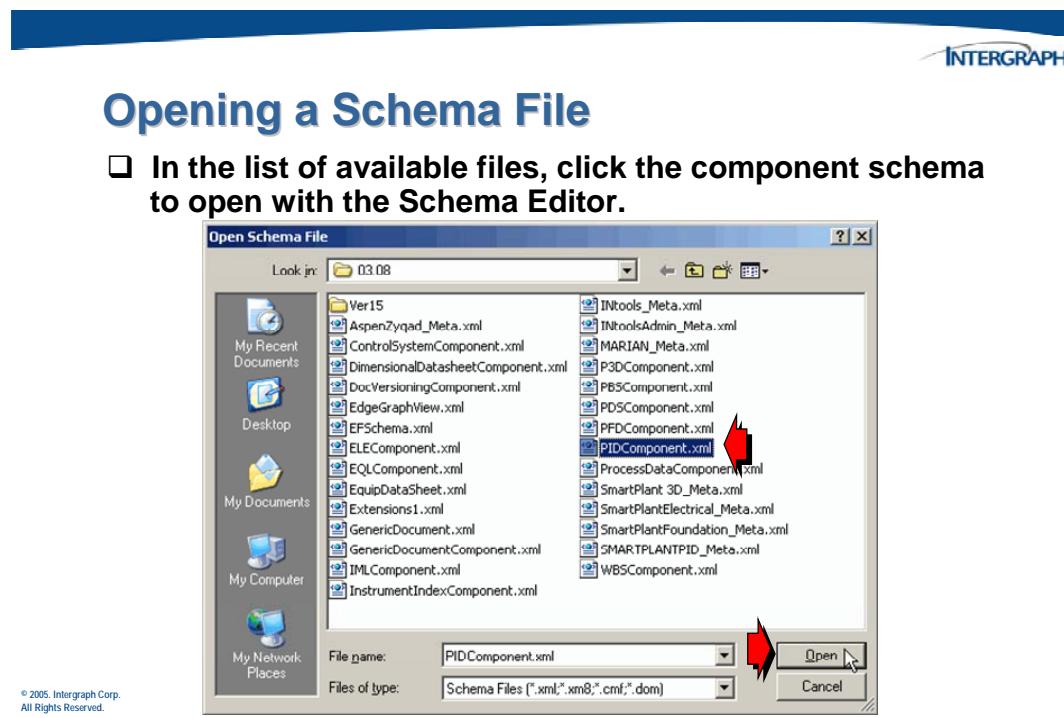
A progress dialog will dynamically show the various component schemas as they are generated.



Another way to open a schema file is to use the ***Open*** command on the ***File*** menu.

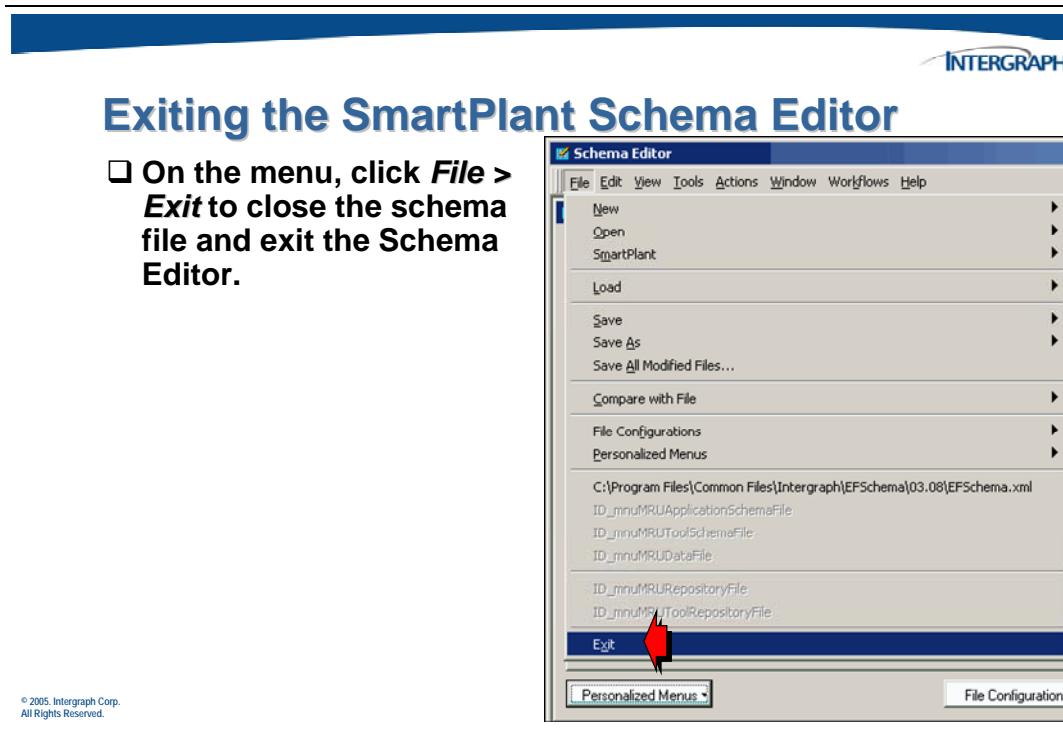


The ***Open Schema File*** dialog box appears.



1.5.2 Exiting the SmartPlant Schema Editor

Once the view windows have been closed, use the *File* command on the main menu to *Exit* from the Schema Editor.

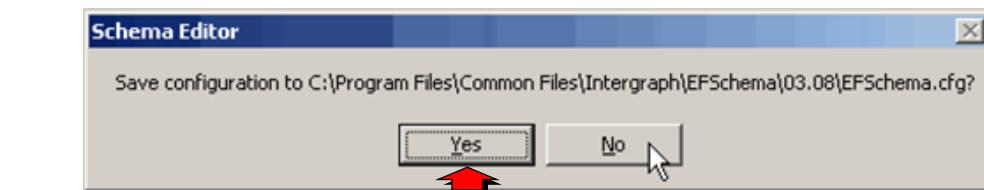


A *Schema Editor* dialog window will display to allow you to save the session configuration.



Exiting the SmartPlant Schema Editor

- Click Yes or No to save the current Schema Editor configuration.**



© 2005, Intergraph Corp.
All Rights Reserved.

Saving a configuration file will be discussed in detail in a later chapter.

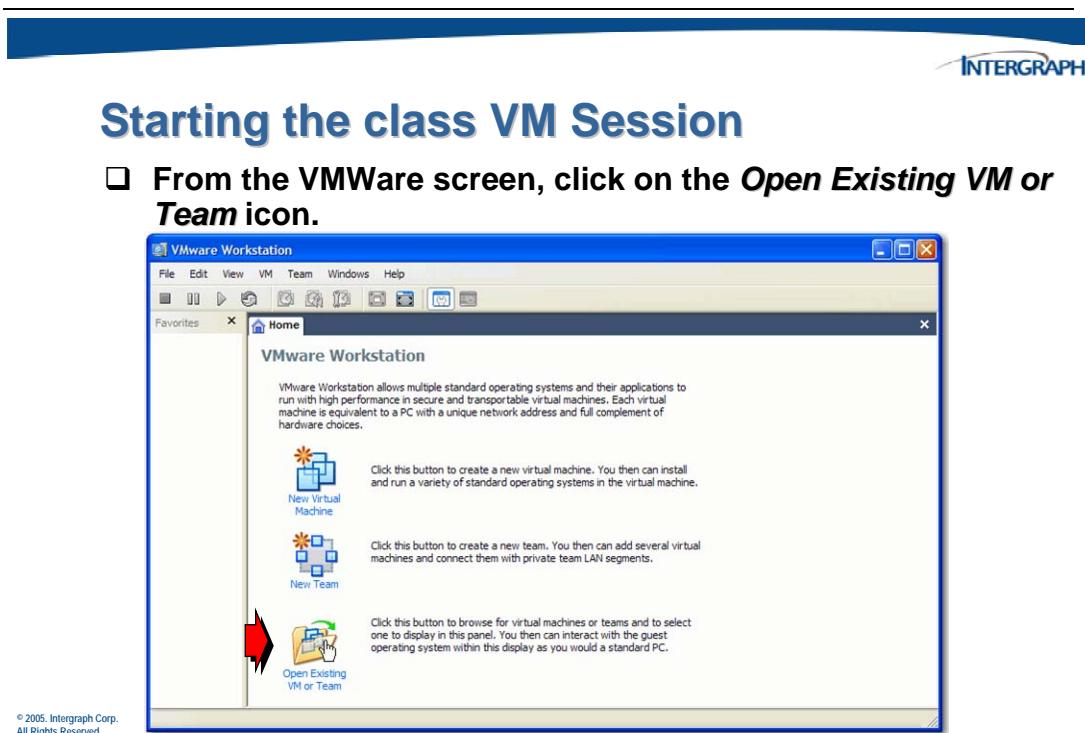
1.6 Using the Class VM Session

Your class will be using an application called VMWare to enable you to login and run the SmartPlant Foundation application and the class hands on activities. This software is a virtual installation of an entire PC machine complete with the Windows 2003 Server operating system and all other necessary applications. You will find an icon on the desktop of your native class machine called *VMWare Workstation*.



Double click on this icon to start the VMWare application.

The *VMWare Workstation* window will display.

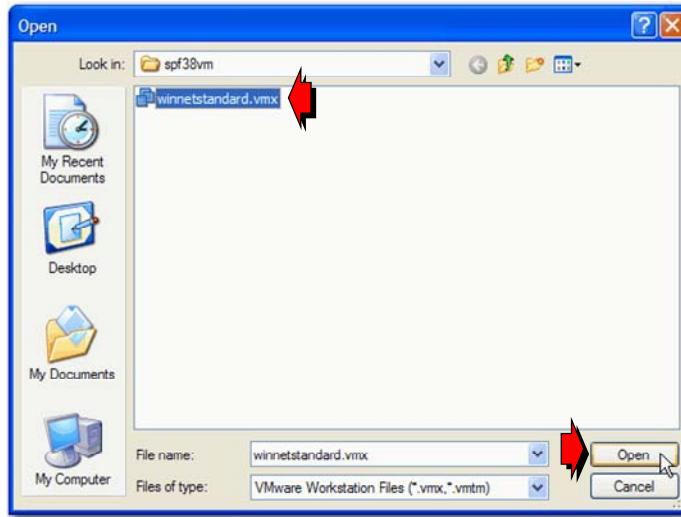


Set your open browser to the folder path specified by your instructor. Write down the path here _____.



Starting the class VM Session

- Choose the class VMWare configuration file as shown.

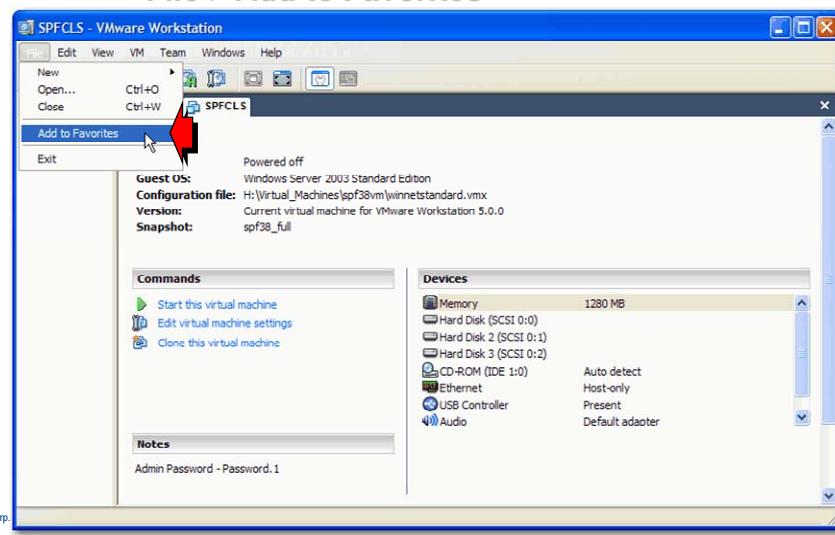


For future convenience, add the vm machine to your favorites.

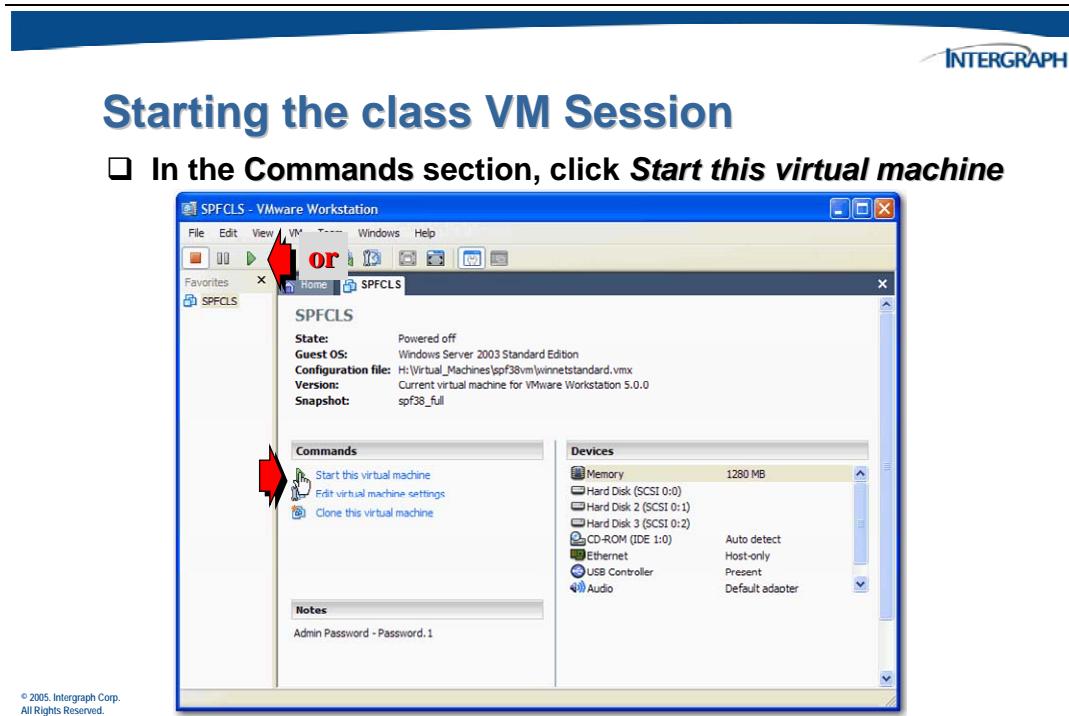


Starting the class VM Session

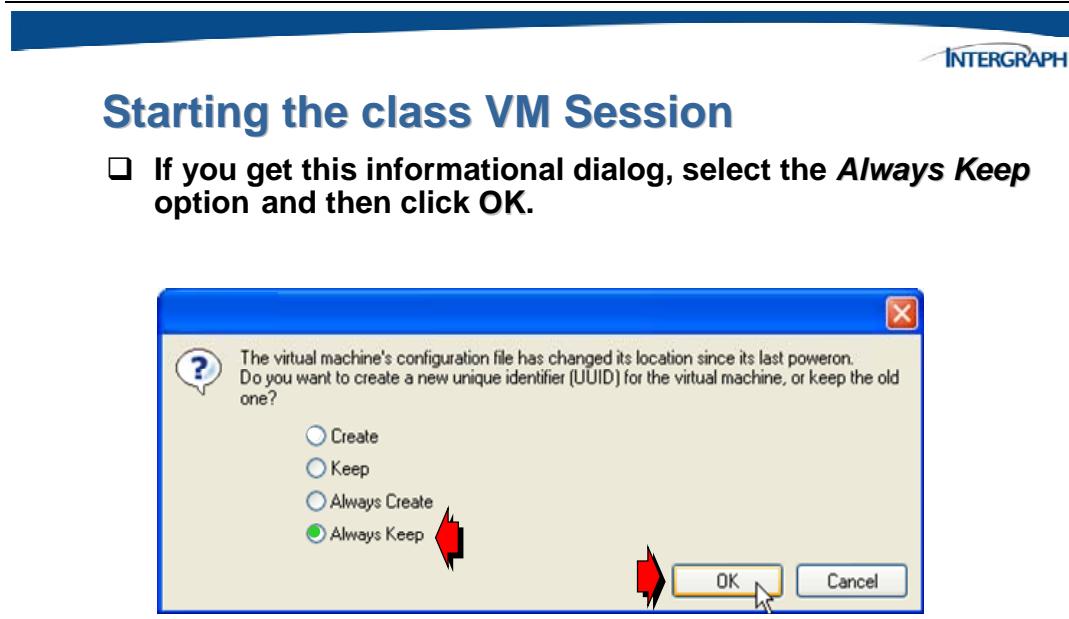
- Select **File > Add to Favorites** from the VMWare menu.



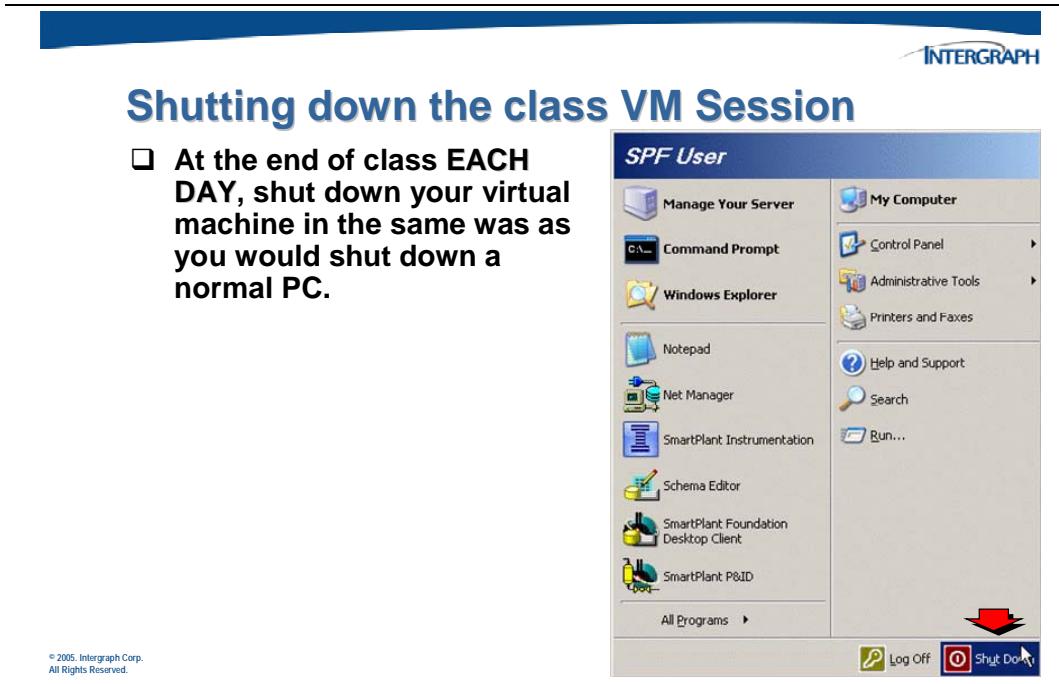
Boot up the virtual machine by using the *Start* command. You can use either the green start button at the top of the window or the one in the *Commands* section.



You may or may not get the next dialog displayed.



When you have finished with your hands on exercises at the end of each day, please shut down your VM session. This will free up memory in your native machine in preparation for the next day.

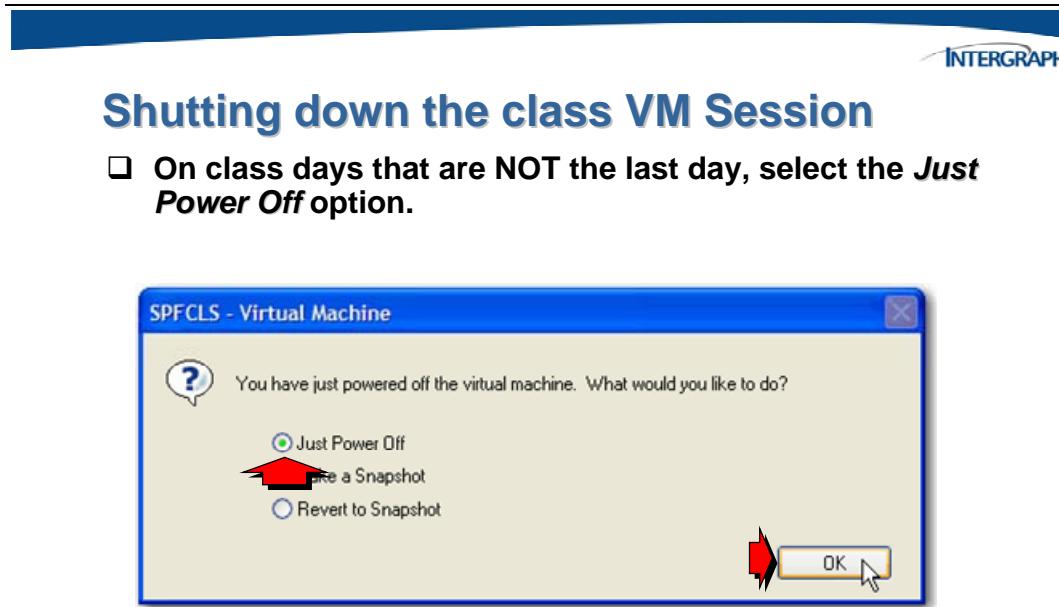


The image shows the Windows Start Menu for a user named 'SPF User'. The menu includes standard items like My Computer, Control Panel, and Administrative Tools. On the right side, there is a 'Shut Down' button highlighted with a red arrow. At the bottom left, it says '© 2005, Intergraph Corp. All Rights Reserved.'

Shutting down the class VM Session

- At the end of class EACH DAY, shut down your virtual machine in the same was as you would shut down a normal PC.**

You will be prompted for an option when powering off the virtual machine.



The image shows a 'SPFCLS - Virtual Machine' dialog box. It asks the user what they would like to do after powering off the virtual machine. The 'Just Power Off' option is selected and highlighted with a red arrow. There are also 'Create a Snapshot' and 'Revert to Snapshot' options. An 'OK' button is at the bottom right, also highlighted with a red arrow. At the bottom left of the dialog box, it says '© 2005, Intergraph Corp. All Rights Reserved.'

Shutting down the class VM Session

- On class days that are NOT the last day, select the Just Power Off option.**

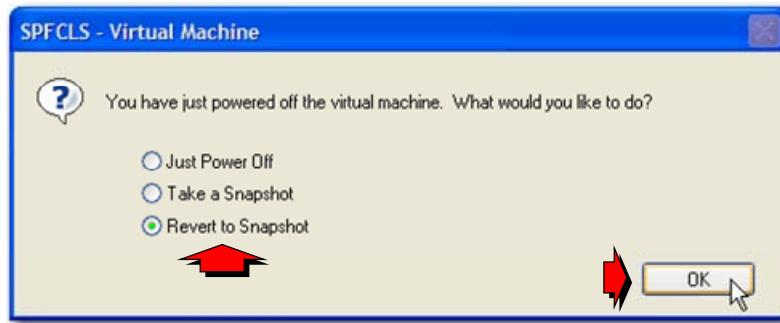
This will keep all of your work just as you left it from that day.

You will only revert to the snapshot on the last day of class when you shut down for the last time. Using the revert option will cause you to lose all of your work from the week.



Shutting down the class VM Session

- ❑ On the last class day ONLY, select the **Revert to Snapshot** option.



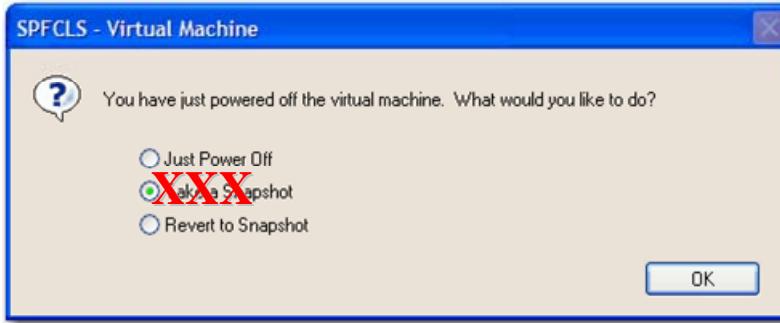
© 2005, Intergraph Corp.
All Rights Reserved.

Don't use the *Take Snapshot* option. This will prevent us from using this VM session for future classes.



Shutting down the class VM Session

- ❑ DO NOT use the **Take Snapshot** option at all during the SmartPlant Foundation training classes.



© 2005, Intergraph Corp.
All Rights Reserved.

1.7 Activity – Starting the SmartPlant Schema Editor

The goal of this activity is to familiarize you with starting the Schema Editor. You will start the Schema Editor, open the SmartPlant schema file, and then familiarize yourself with generating component schemas.

1. Log on to your operating system as *spfuser* with no password.
2. Click **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
3. Once you have started the *Schema Editor* the **Workflows** dialog box appears. Open the SmartPlant schema.
 - Click the **Open** button under the **Schema tab** in the Workflows dialog box.
 - Browse to **C:\Program Files\Common Files\Intergraph\EFSchema\03.08** in the *Open Schema File* dialog box.
 - Select **EFSchema.xml** file and click **Open**.
4. Even though you already have the necessary component schemas delivered as a result of the installation, familiarize yourself with generating component schemas.
 - On the *Schema Editor* menu, click **Tools > Schema > Generate Component Schemas**.
5. Open the PID component schema (PIDComponent.xml) file.
 - Click **File > Open > Schema File...** on the *Schema Editor* menu.
 - Browse to **C:\Program Files\Common Files\Intergraph\EFSchema\03.08** in the *Open Schema File* dialog box.
 - Select **PIDComponent.xml** file and click **Open**
6. Click **File > Exit** command to close the *Schema Editor* utility.
 - A *Schema Editor* dialog window displays to allow you to save the session configuration.
 - Click **No**

Summary:

In this activity, you accessed the Schema Editor and familiarized yourself with opening a schema file and generating component schemas.

C H A P T E R

2

Using the Schema Editor

2. Using the Schema Editor

The SmartPlant schema is represented by a Unified Modeling Language (UML) model entitled SmartPlant schema. A solid understanding of UML is needed to completely comprehend and digest the complexities of the SmartPlant schema.



Unified Modeling Language

Data modelers have agreed to use symbols that are part of a set of modeling practices called Unified Modeling Language, or UML .

UML is used as a symbolic framework upon which the ideas that are being modeled are expressed.

Even an experienced data modeler needs to know what symbols represent what ideas.

The precise nature of the symbols (and symbolic logic, in general) virtually guarantees that ambiguity is minimized, and accuracy is maximized.

© 2005, Intergraph Corp.
All Rights Reserved.

There were many attempts during the last 20 years to arrive at a "common" understanding of how to represent ideas on a diagram. UML won the war because it is so easy to use, and is extensible (for new ideas that might come along at any time).

In the *Schema Editor*, you can view any schema in .XML format. The many different schema views in the Schema Editor provide different types of information that help you understand how the schema works. 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 of the schema. These views allow you to see both static and dynamic UML diagrams containing the classes, relationships, interfaces, and properties defined in the schema.



Viewing a Schema File

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

Some of the most useful ways to view the schema in the Schema Editor are the following:

- Using the **Tree**, **Tree/Table**, and **Tree/Properties** views
- Using the **Tree/UML** graphical view
- Using the **Editor** view
- Using the **Drag-and-Drop UML** views
- Using the **Tree/MultiTab** view

© 2005, Intergraph Corp.
All Rights Reserved.

There are many other ways to view the schema in the Schema Editor depending on what aspect of the schema that you want to understand.

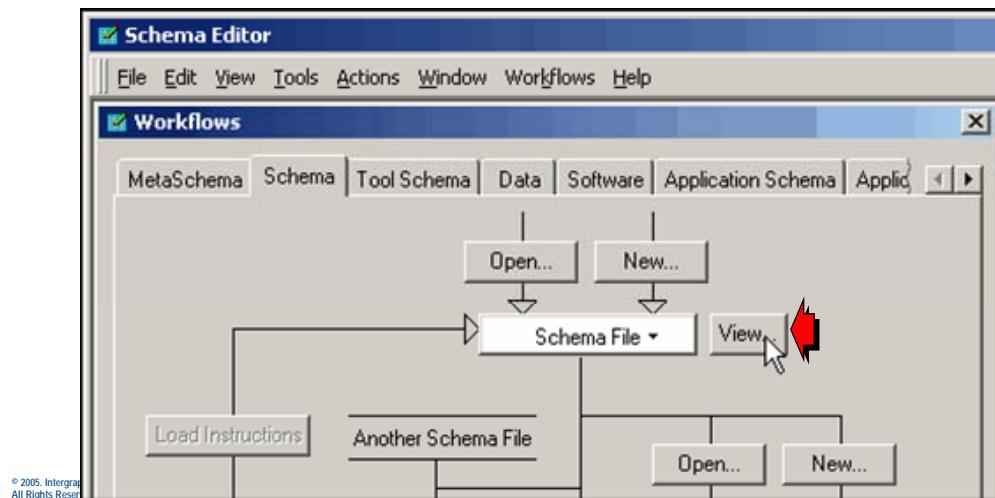
2.1 Viewing an Open Schema File

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



Viewing a Schema File

- In the **Workflows** dialog box, click the **View** button to view the open schema file.



Shown below are some short explanations for the tabs visible in the Workflows dialog:

MetaSchema - Provides access to commands for opening and viewing the metaschema.

Schema - Provides access to commonly-used schema commands, including commands for opening multiple schema files, viewing schema files, and comparing schema files.

Tool Schema - Provides access to commonly-used tool map schema commands, including commands for opening multiple tool map schemas, viewing tool map schemas, comparing tool map schemas, showing mapping, setting the active tool map schema, and so on.

Data - Provides access to commonly-used data commands, including commands for opening multiple data files, viewing data files, comparing data files, and validating data files.

Software - Provides commands to allow the Schema Editor to be used to browse the contents of software libraries (DLLs, OCXs, TLBs, and so on).

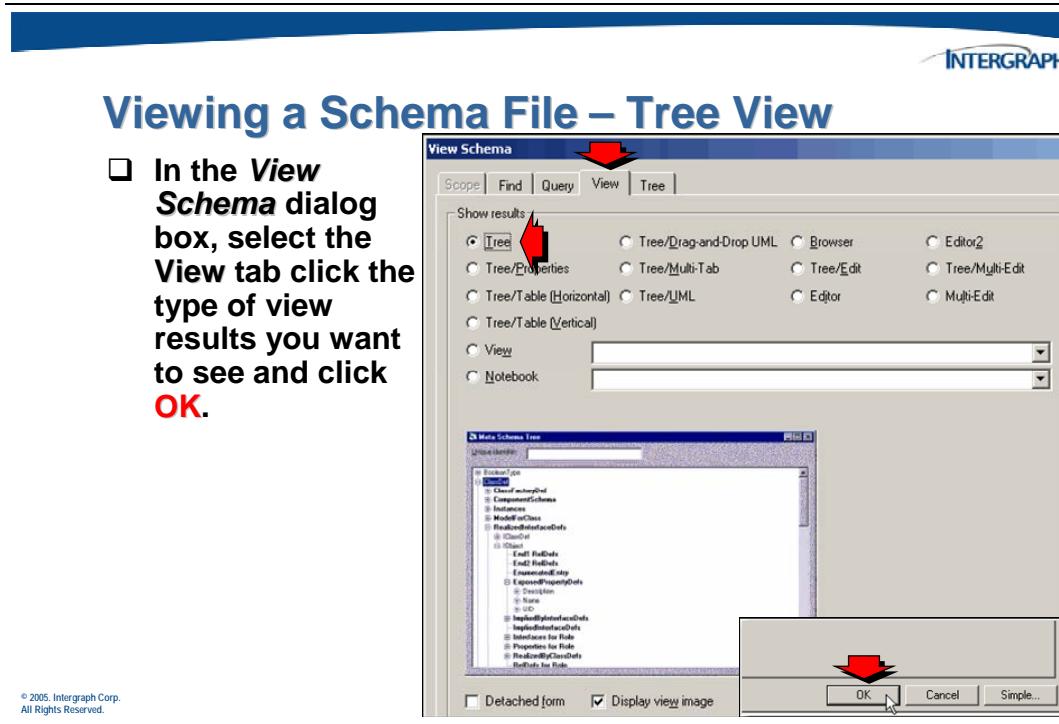
Application Schema - for future use.

Application Metadata - Allows you to connect to SmartPlant and to authoring tool metadata adapters to modify tool metadata in the authoring tool database, define mapping

between the tool metadata and the SmartPlant schema, and synchronize the tool metadata with the tool map schema and the SmartPlant schema and vice versa.

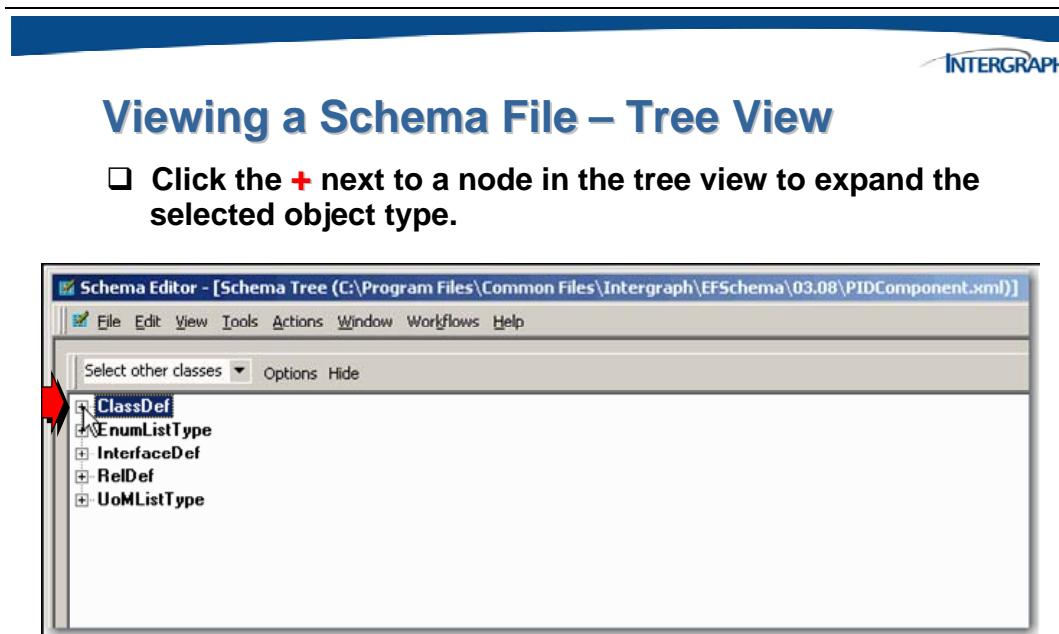
2.1.1 Schema File Tree View

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

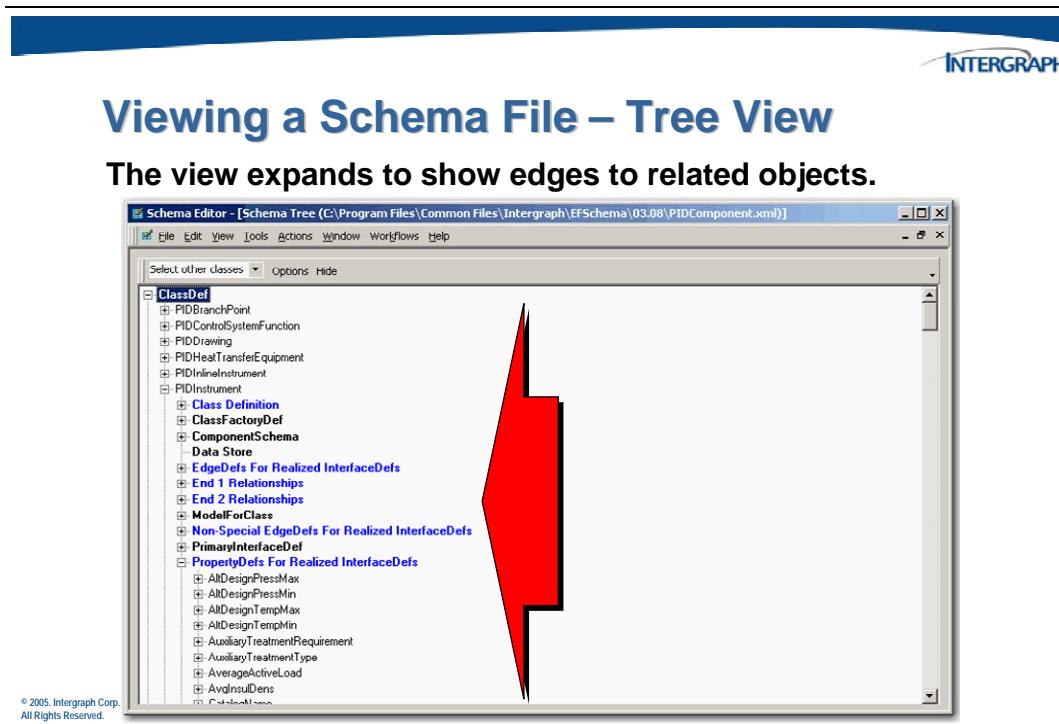


The tree view is useful for general navigation through relationships, but not for looking at property values. You can navigate all the relationships defined in the schema using the Schema Tree view.

The *Schema Tree* window will be displayed.



The *ClassDef* object relationships are shown.

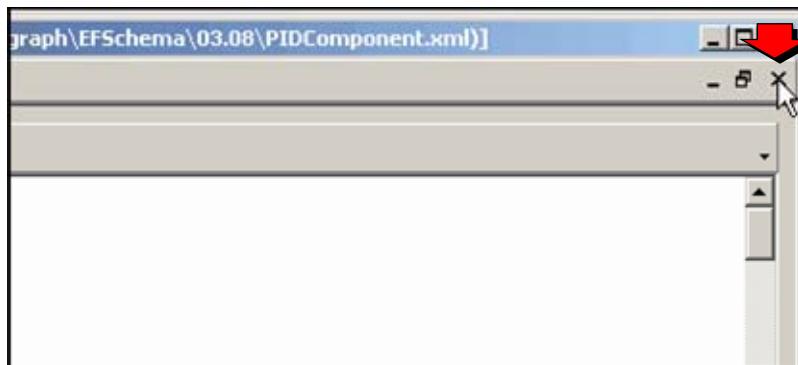


In the tree, objects that are colored **black** are **standard objects** while those in **blue** are **special** (multiple relationships) and are limited.



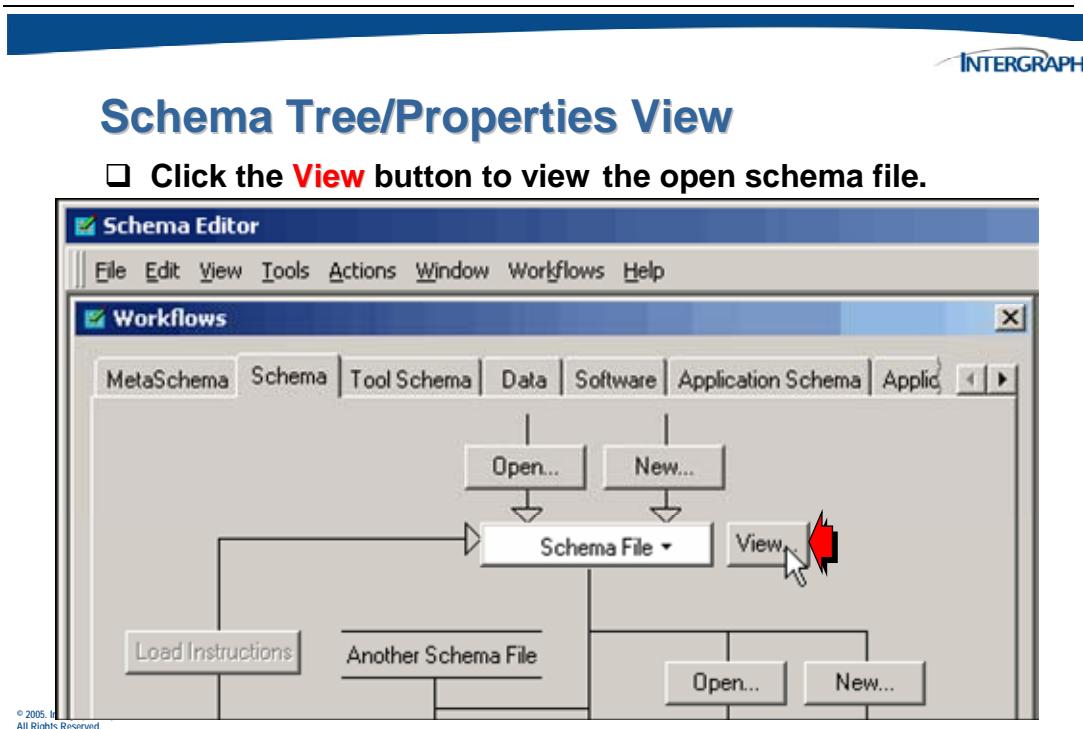
Viewing a Schema File – Tree View

- Click on the X to close the Tree View window.

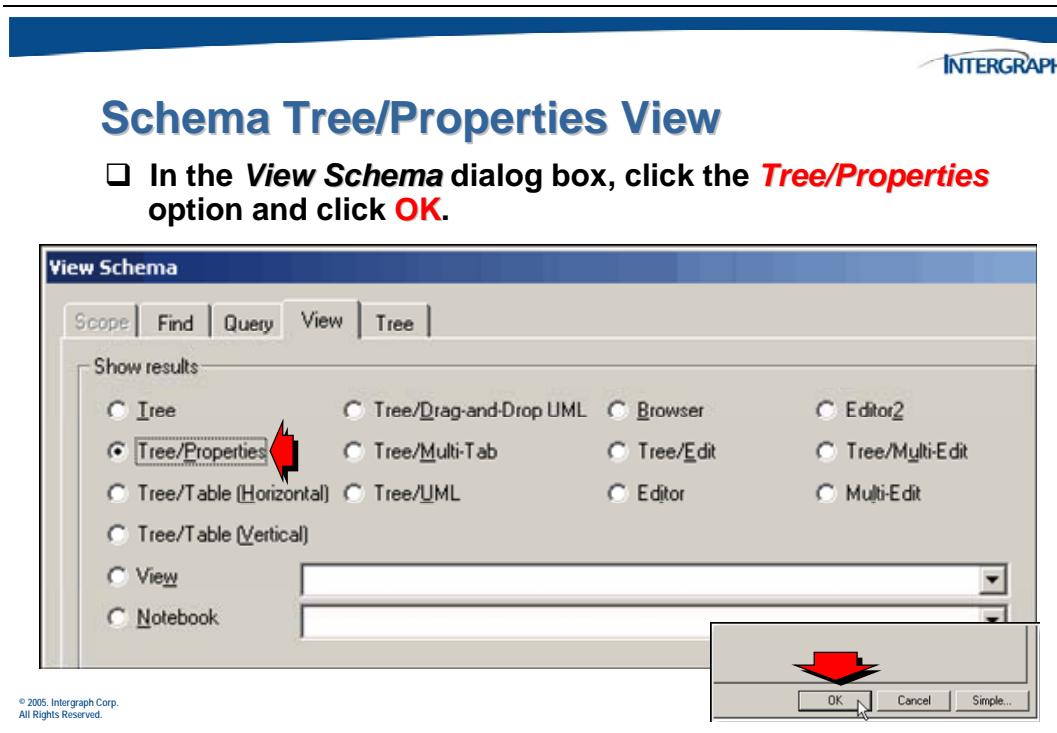


2.1.2 Schema Tree/Properties View

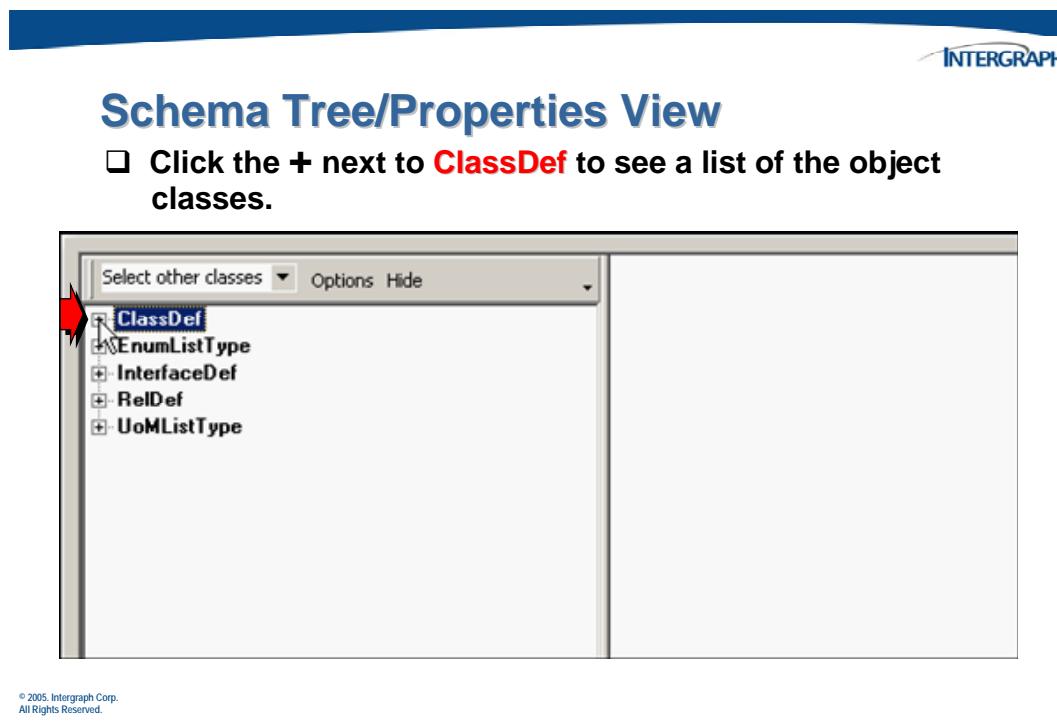
The **Tree/Properties** view combines the same tree view used elsewhere in the Schema Editor with a properties view that emulates display of properties in the right pane in SmartPlant Foundation. This properties view identifies each interface for the selected object with text on a shaded background. Following each interface are the properties exposed by that interface and their values for the selected object.



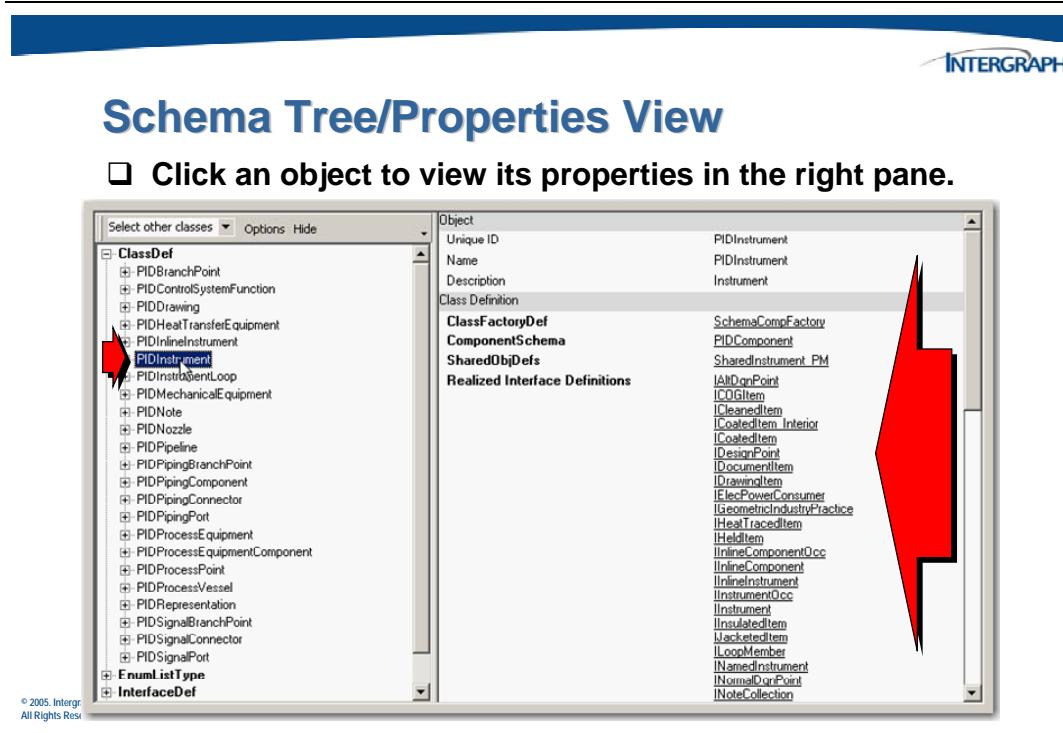
Click the **View** tab in the *View Schema* dialog box.



The *Schema Tree/Properties* window will be displayed.

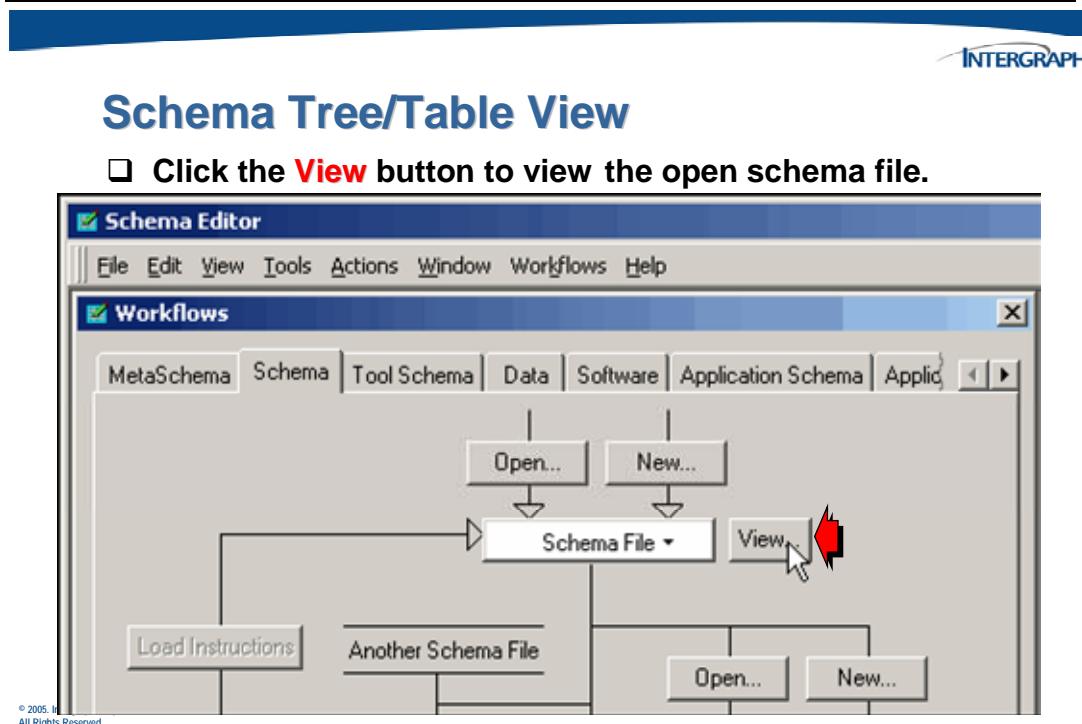


The view expands to show properties related to the class objects.

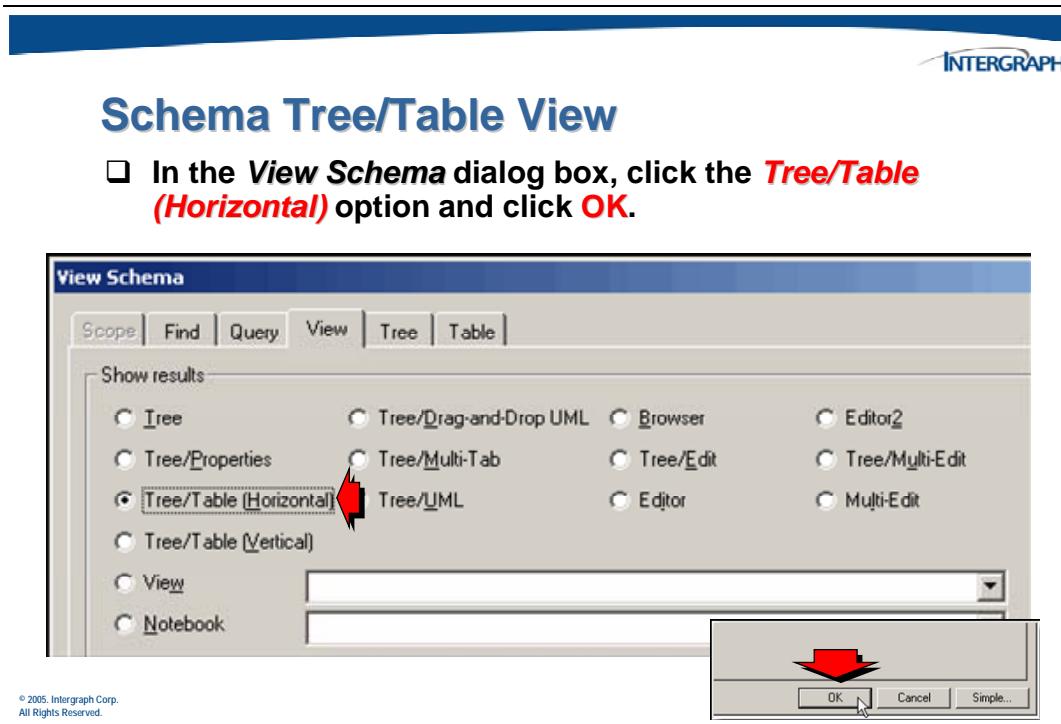


2.1.3 Schema Tree/Table View

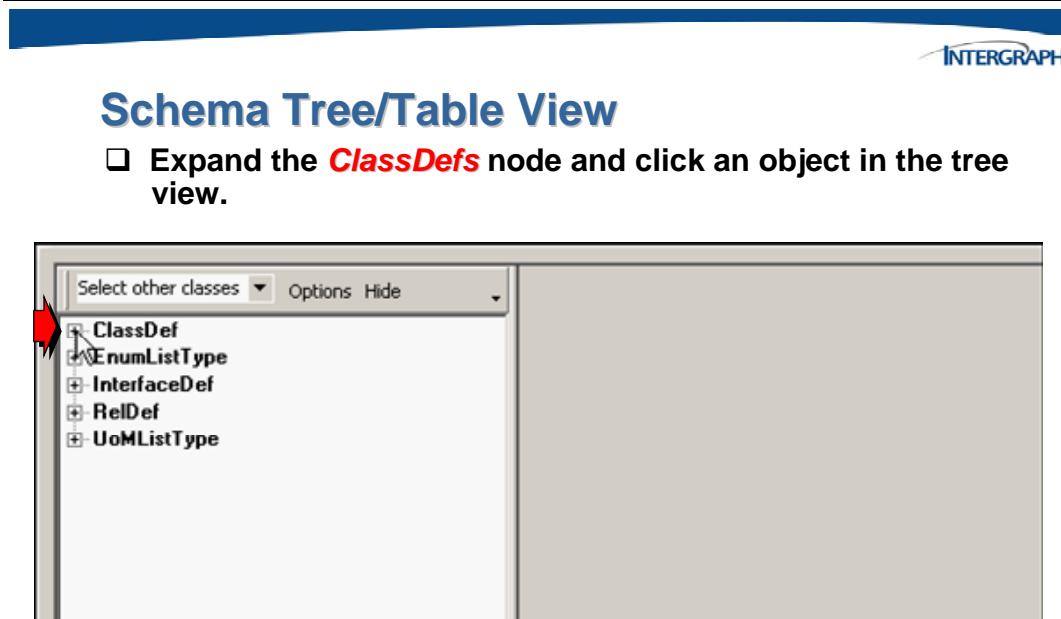
The Schema **Tree/Table** view is a combination of the standard tree view with a table that provides a different view of the information based on what you select in the tree view. The table views are predefined by class to show the most useful information for each type of object.



Click the **View** tab in the *View Schema* dialog box.



The *Schema Tree/Table* window will be displayed.



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. In the following example, a horizontal table is display is used.

The screenshot shows the Schema Tree/Table View interface. On the left is a tree view of classes and interfaces. The main area is a table view with three columns: Class, Interface, and Property. A large red arrow points from the text "The object properties are displayed in the right pane in a horizontal table." to the table area. The table data is as follows:

Class	Interface	Property
PIDInstrument	IDocumentItem	
	IObject	UID Name Description
	IWeightItem	Dry_Stripped_Weight Dry_Installed_Weight Wet_Operating_Weight DesignWeight TestWeight
	ICoatedItem	CoatingColor Exterior_CoatingArea CoatingRequirement CoatingType Exterior_SurfaceTreatmentType AuxiliaryTreatmentRequirement AuxiliaryTreatmentType Exterior_SurfaceTreatmentRequirement
	IGeometricIndustryPractice	
	IInlineComponentDoc	
	IIlineInstrument	
	Instrument	ProcFunc Instrument_LineSize Instrument_FluidType InstrumentType InstrumentType3 Instrument_FluidSize

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.

Properties that appear in bold text in the table view are required for the associated interface.

Bold property types in the table view indicate that the value for the property cannot be null.

The screenshot shows the Schema Tree/Table View interface. On the left is a tree view with nodes like ClassDef, PIDBranchPoint, PIDControlSystemFunction, PIDDrawing, PIDHeatTransferEquipment, PIDInlineInstrument, PIDInstrument, PIDInstrumentLoop, PIDMechanicalEquipment, PIDNote, PIDNozzle, PIDPipeline, PIDPipingBranchPoint, PIDPipingComponent, PIDPipingConnector, PIDPipingPort, PIDProcessEquipment, PIDProcessEquipmentComponent, PIDProcessPoint, PIDProcessVessel, PIDRepresentation, PIDSignalBranchPoint, PIDSignalConnector, PIDSignalPort, EnumListType, and InterfaceDef. On the right is a table view with columns 'Property' and 'Type'. The table contains numerous properties and their types, such as UID (string128), Name (string), Description (string), Dry_Installed_Weight (MassUoM), Wet_Operating_Weight (MassUoM), DesignWeight (double), TestWeight (double), CoatingColor (CoatingColors), Exterior_CoatingArea (AreaUoM), CoatingRequirement (CoatingReq1), CoatingType (CoatingReq2), Exterior_SurfaceTreatmentType (ExteriorSurfaceTreatment2), AuxiliaryTreatmentRequirement (AuxiliaryTreatmentRequirement), AuxiliaryTreatmentType (AuxiliaryTreatmentType2), and Exterior_SurfaceTreatmentRequirement (ExteriorSurfaceTreatmentRequirement). A red arrow points to the scroll bar on the right side of the table view. Another red arrow points to the 'Instrument' column header in the table view.

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.

2.1.4 Schema Tree/Drag-Drop UML View

In the Schema Editor, you can view the schema using a variety of Unified Modeling Language (UML) views. The most customizable of all the UML views are the two drag-and-drop UML views. These views allow you to drag objects from a tree view into 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.



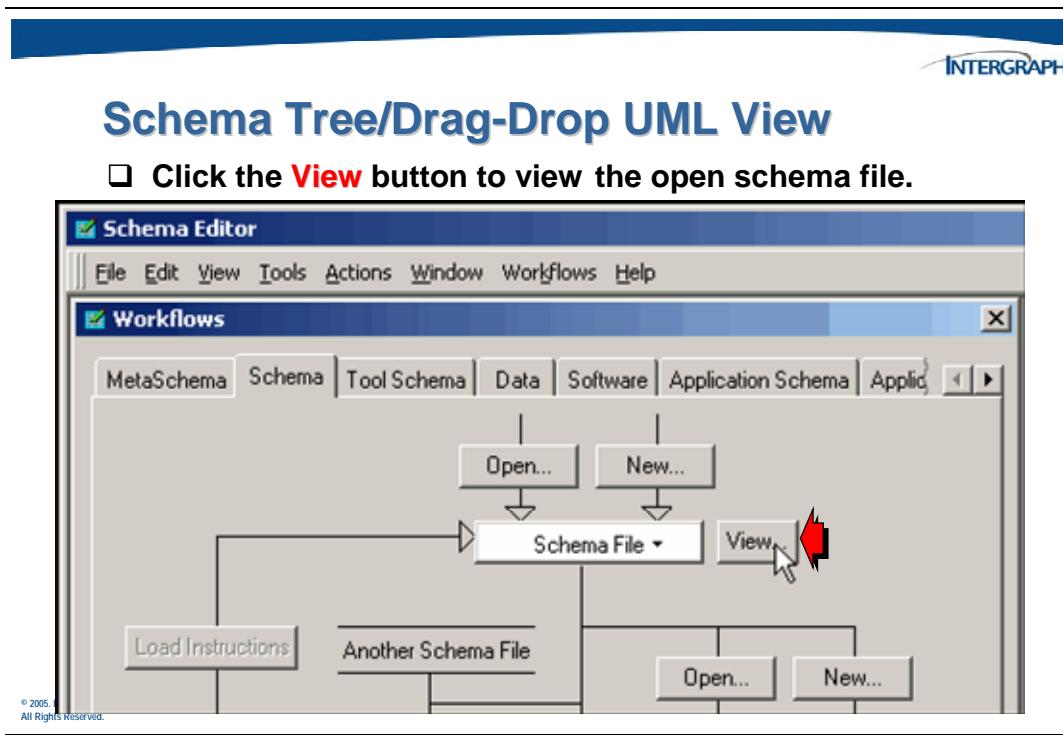
Schema Tree/Drag-Drop UML View

You can create your own UML view by selecting the objects that you want to display in the UML view. You can also select a UML view from the *Packages* tree.

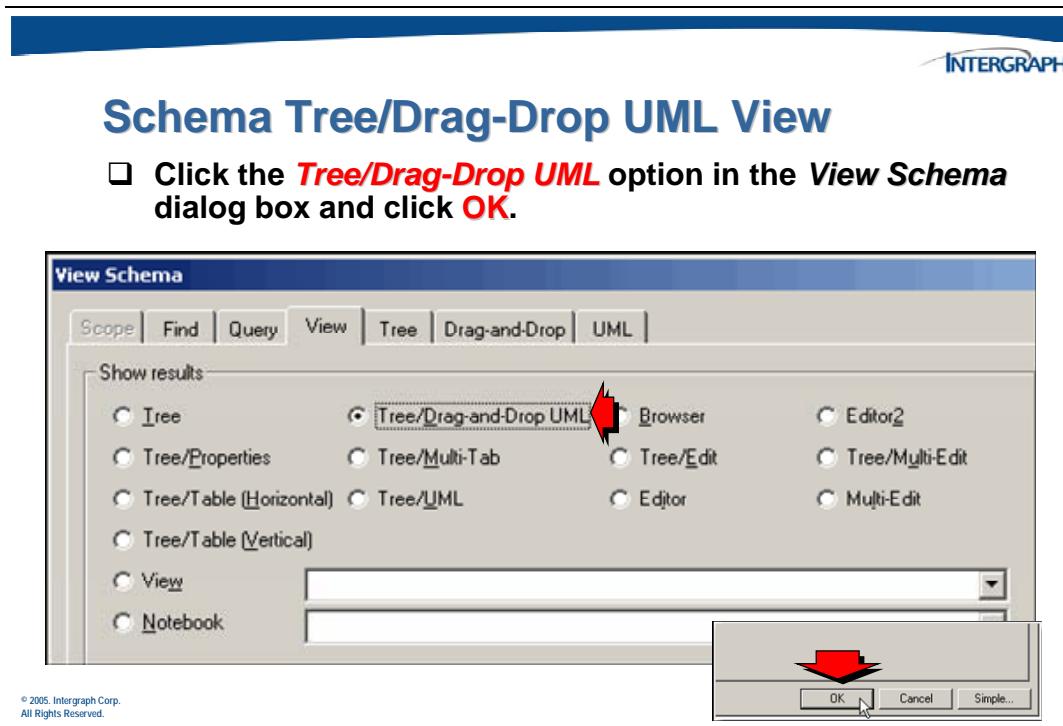
The UML view can be customized even further by:

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

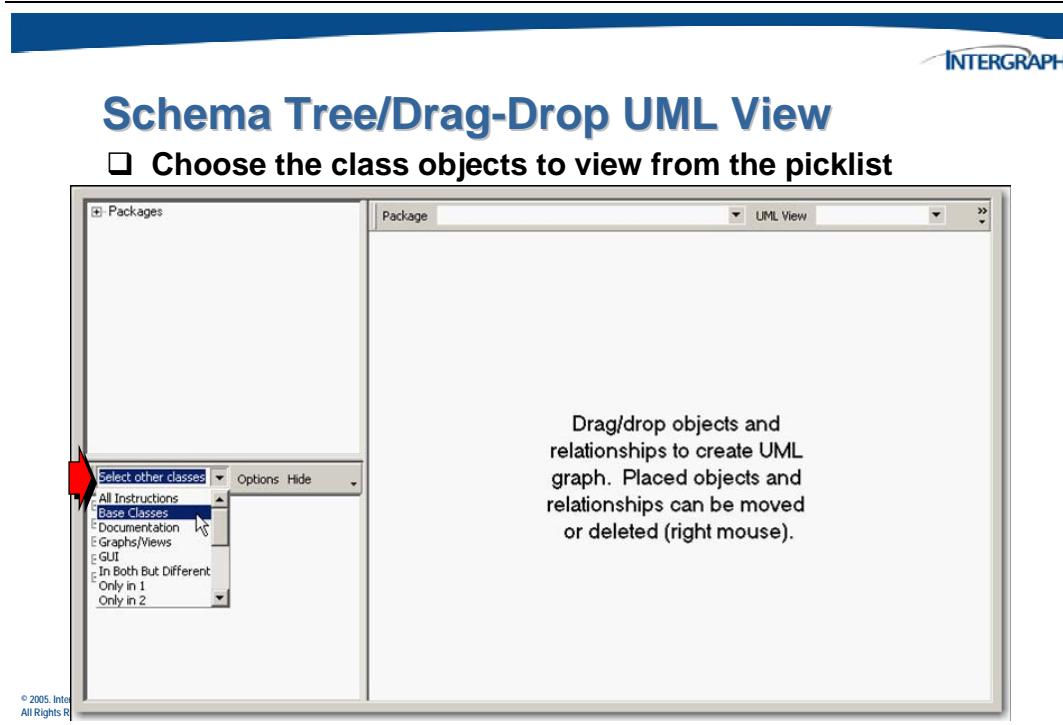
Open the schema that you want to view.



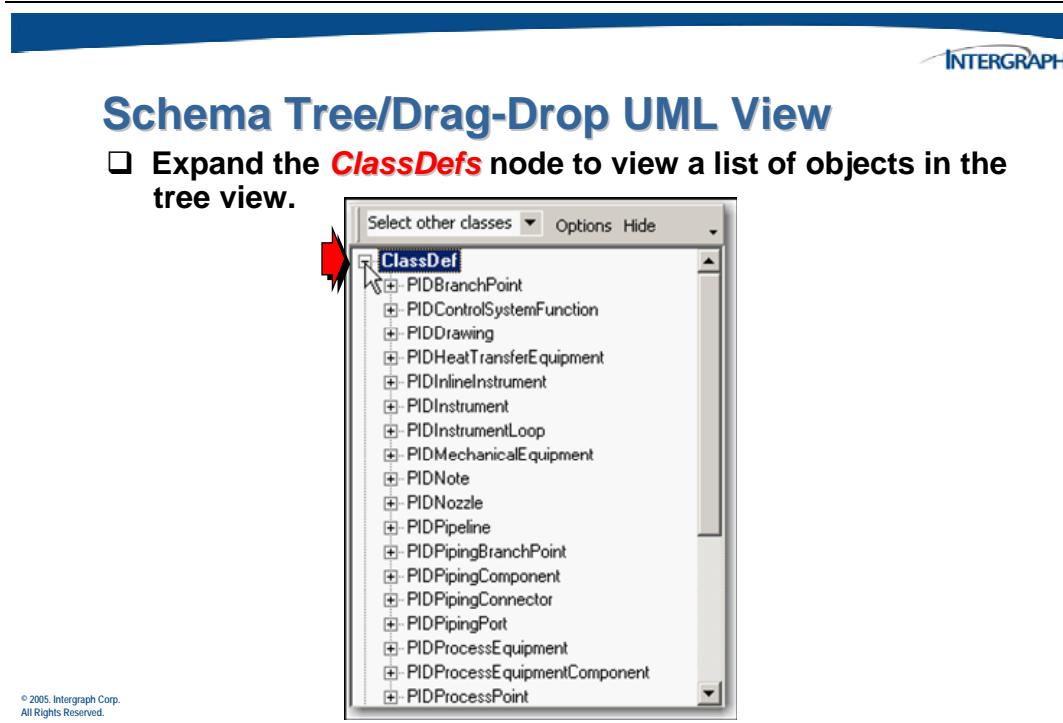
Click the **View** tab in the *View Schema* dialog box.



The **Tree/Drag-Drop UML View** displays a tree view on the left of the drag-and-drop UML pane.



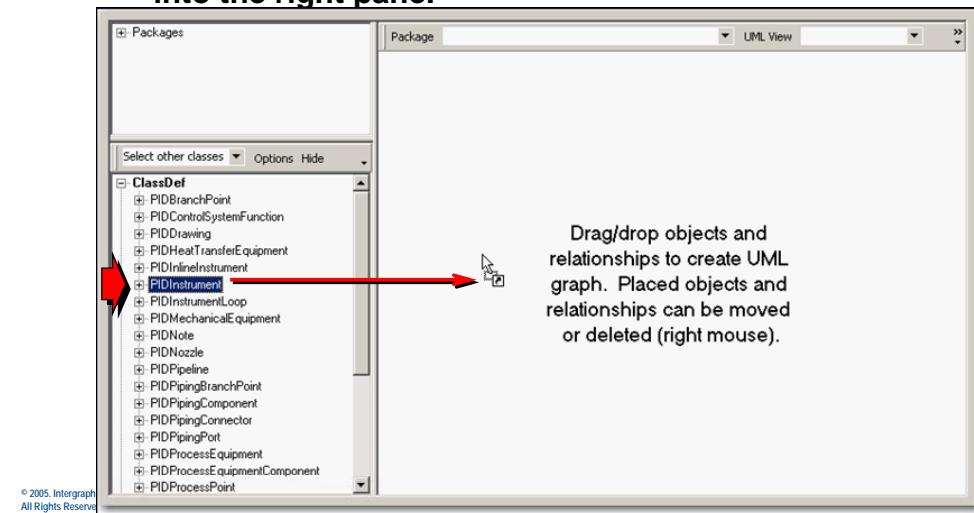
You can select items from the tree view to display in the UML view by dragging the objects from the tree view to the UML view.





Schema Tree/Drag-Drop UML View

- Drag and drop the selected class object from the tree view into the right pane.

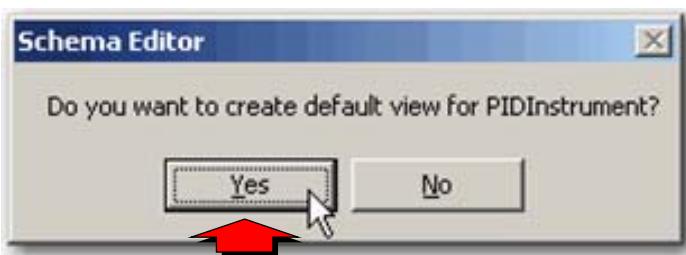


When you drag and drop an object to the UML view, you will be prompted to create either a *custom* UML view or a default view. An example of creating a custom UML view will be demonstrated later.

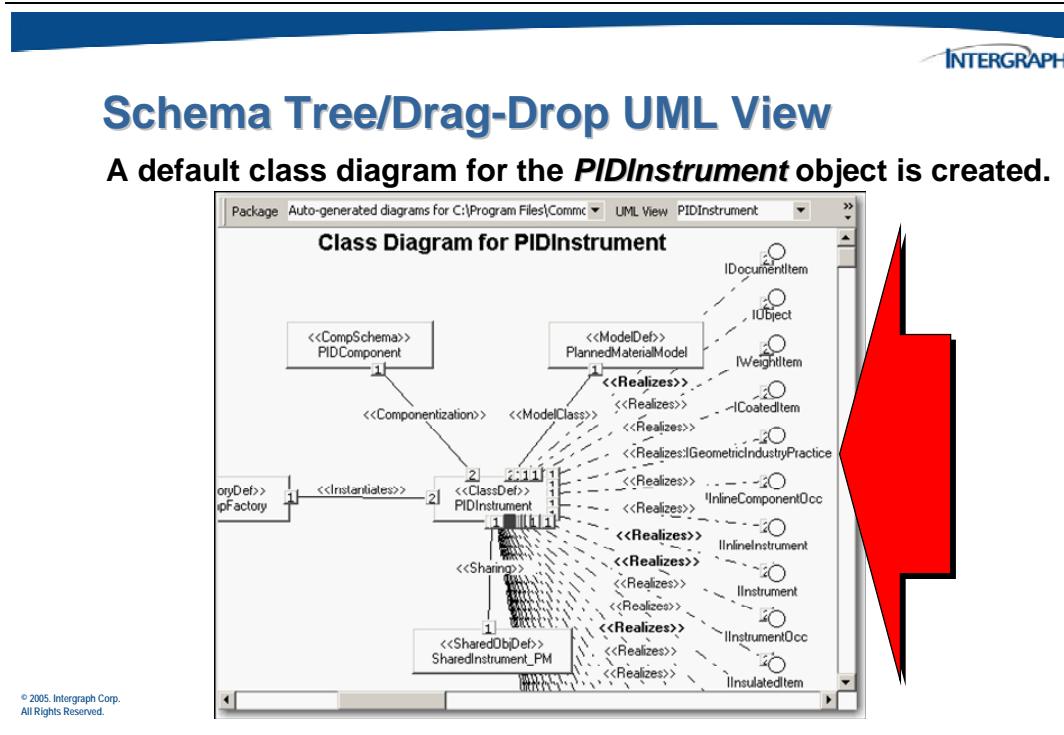


Schema Tree/Drag-Drop UML View

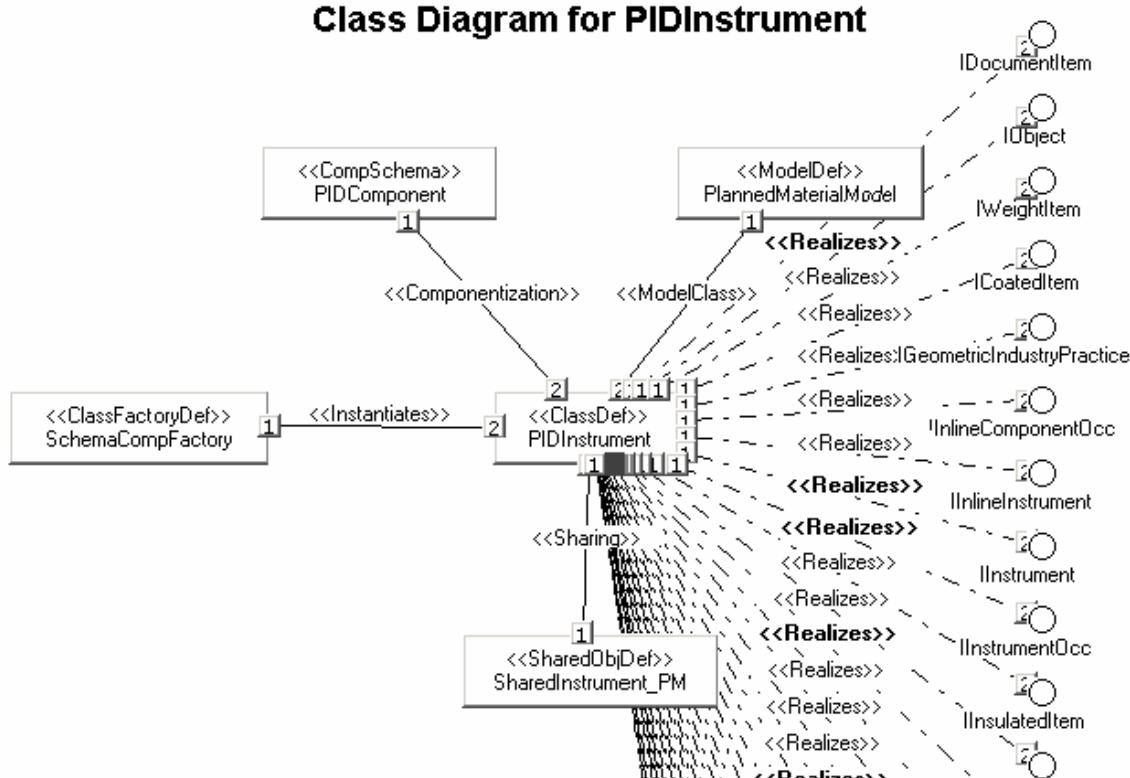
- Click **Yes** to create a default UML view.



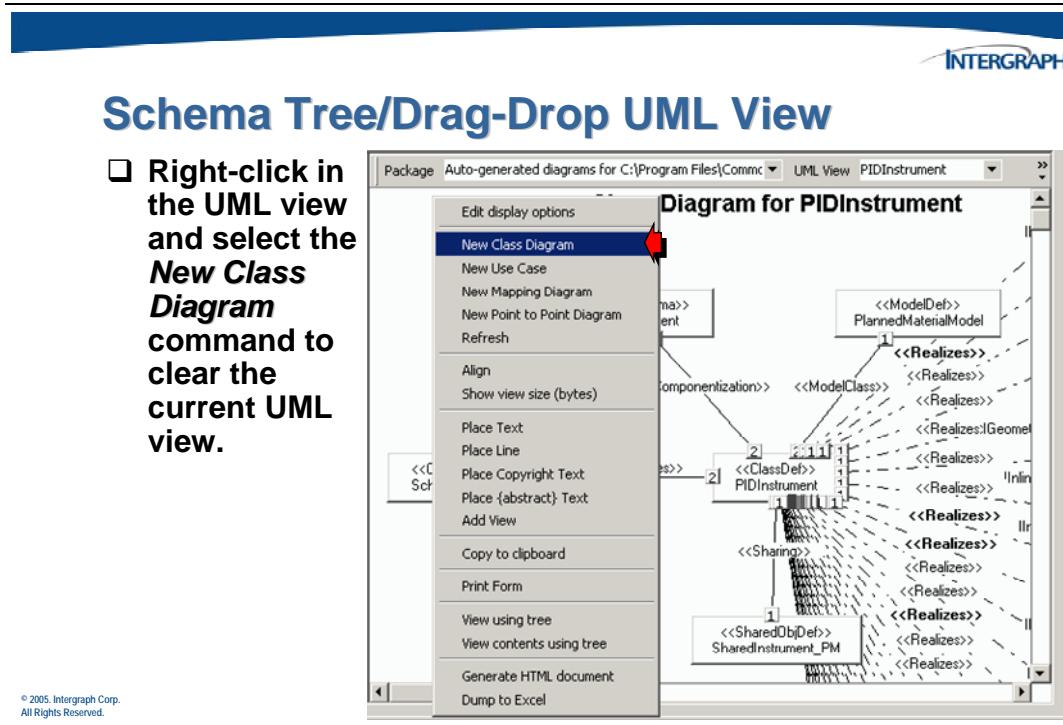
In the resulting UML view, you can view the realizes relationships between the selected objects and the relationship ends for each one.



Below is a magnified example of the Schema Tree/Drag-Drop UML view.



Use the right mouse button menu to clear the UML view back to a blank view.



2.1.5 Interface Relationships

In the previous chapter, relationships between interfaces were introduced. In the examples used in this chapter, you will be exposed to different kinds of relationships that are used to associate interfaces.

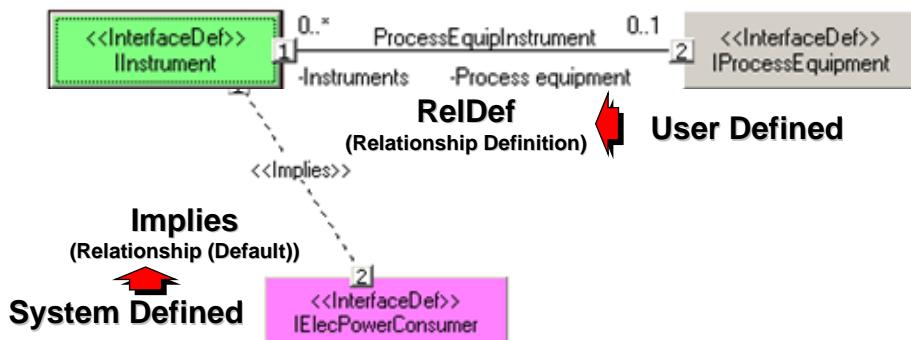
The two types of relationships used from interface to interface are:

- Relationship Definitions (RelDef) – these are user defined relationships defined by the data modeler and are used to associate interfaces that are realized by different class definitions (ClassDef's).
- Implied (Relationship) – this type of relationship is thought of as a system default relationship. It is used primarily for inheritance from one interface to another interface where both are realized by the **same** ClassDef.

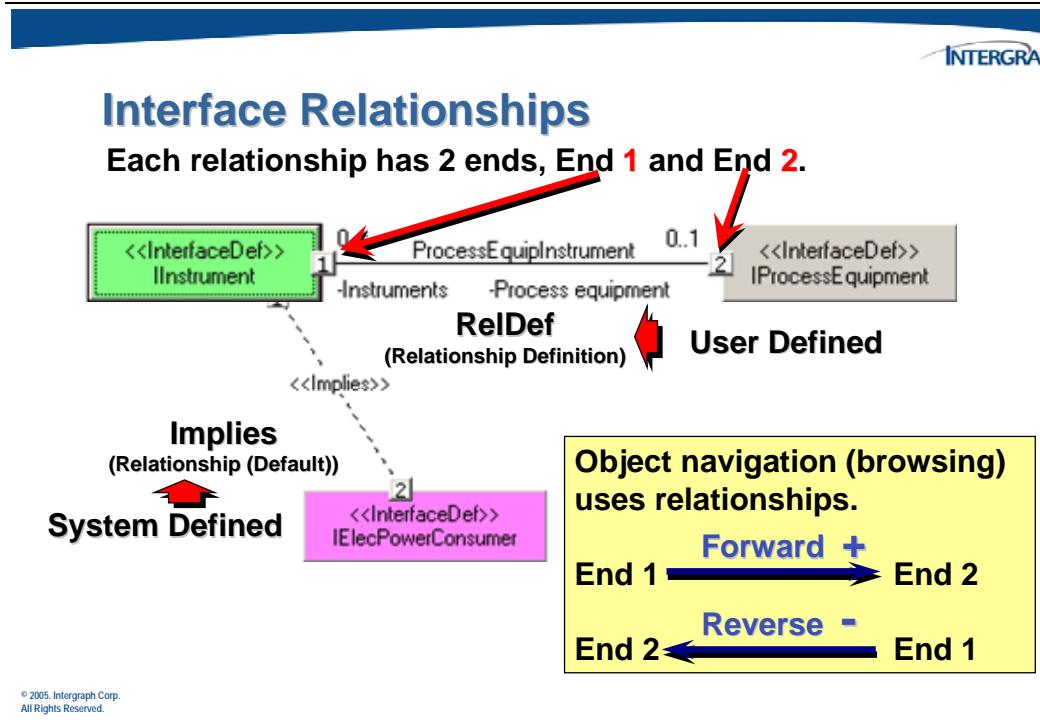


Interface Relationships

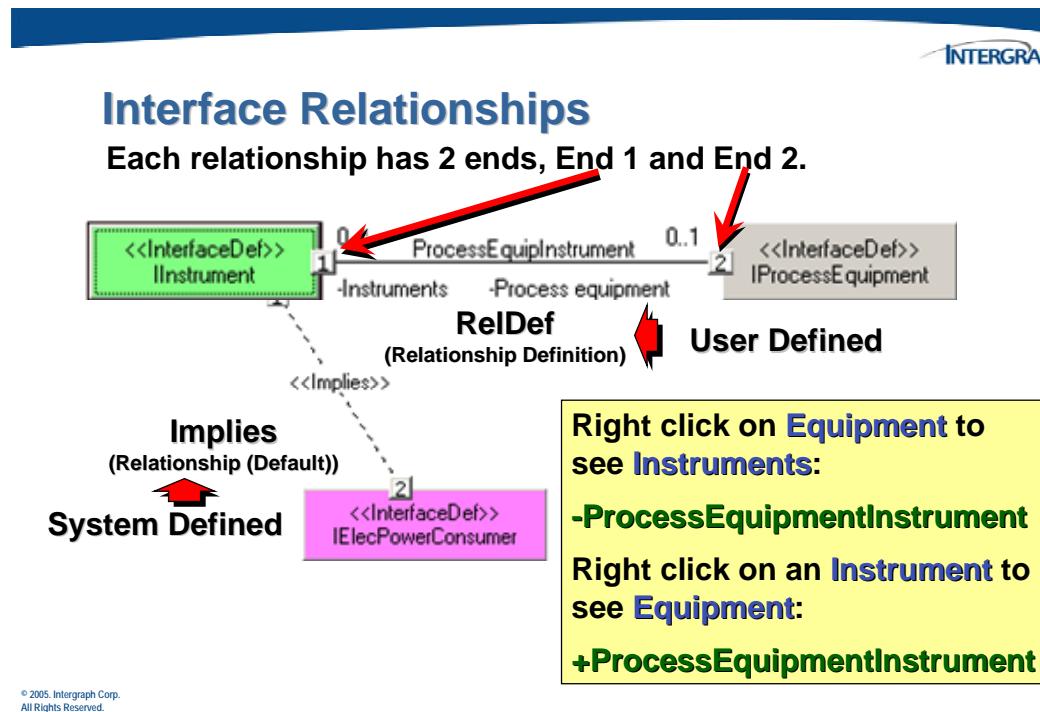
There are two kinds of relationships that can exist between InterfaceDef's.



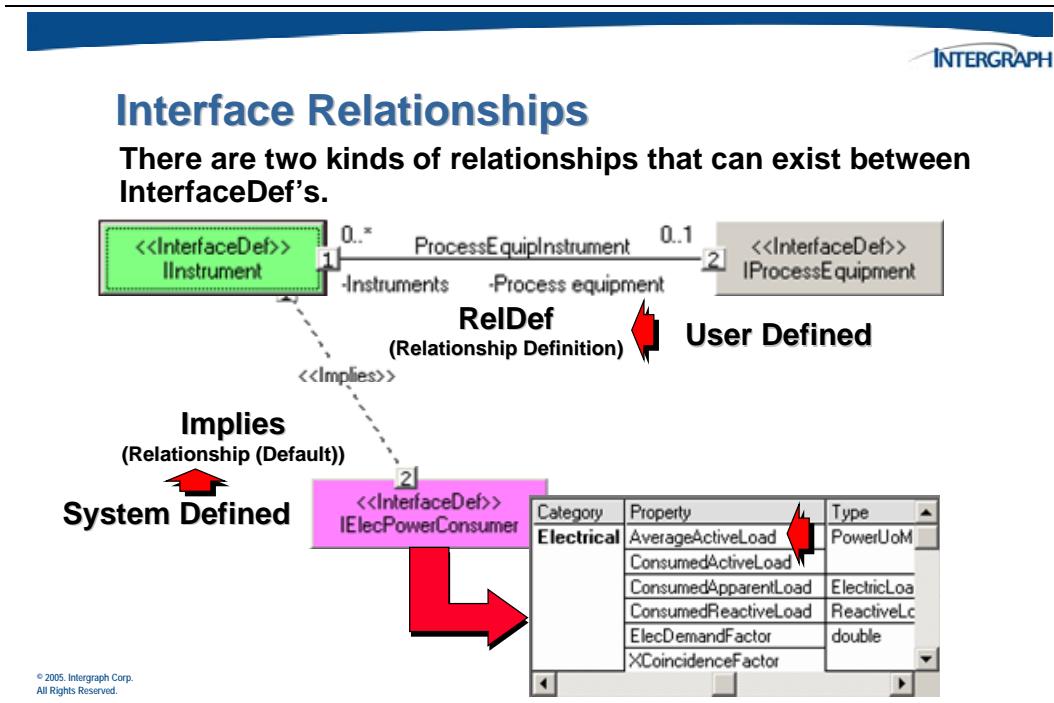
Both *RelDef's* and *Implied* relationships can be used for object navigation or browsing, but RelDef's provide for more browsing control. This will be discussed in more detail in chapter 5.



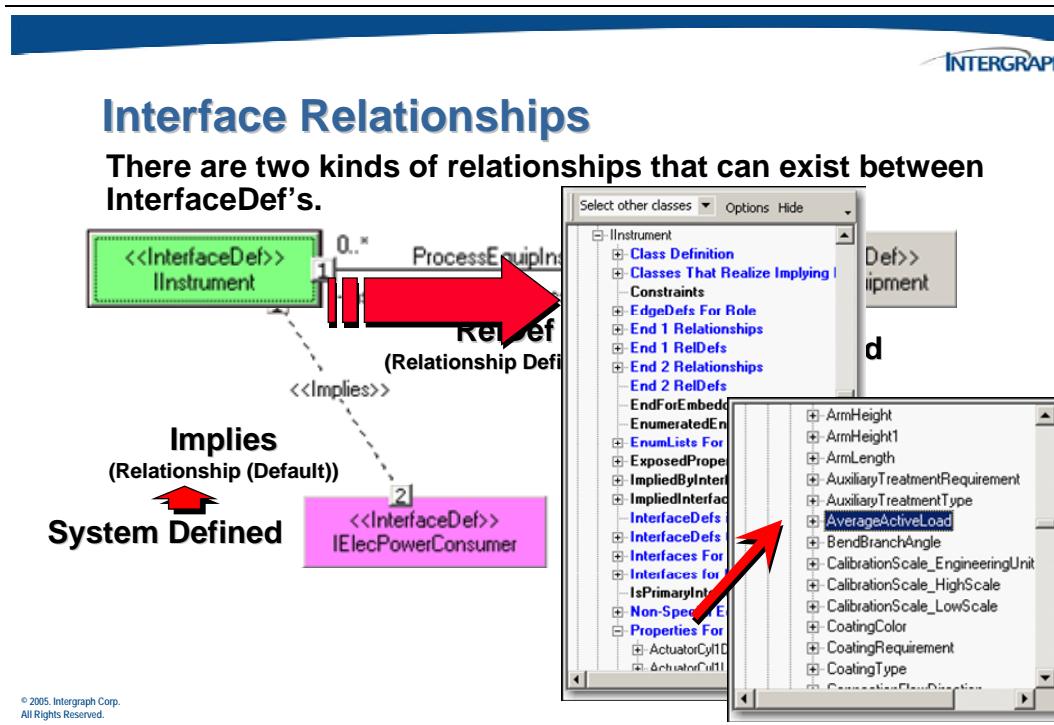
Either forward or reverse browsing is possible in the Desktop Client.



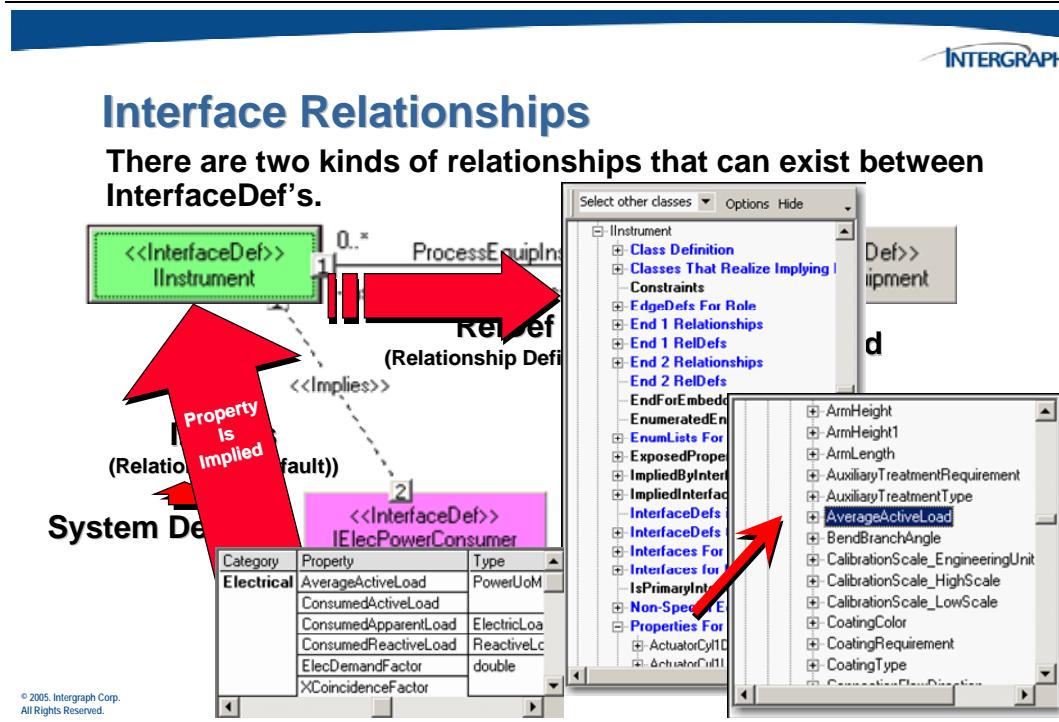
One of the primary functions of the **Implies** relationship is to allow an interface to inherit properties and relationships from an associated interface. For example, the interface *IElecPowerConsumer* exposes a property called **AverageActiveLoad**.



The property *AverageActiveLoad* will display as one of the **Properties For Role** for the interface **IInstrument**.



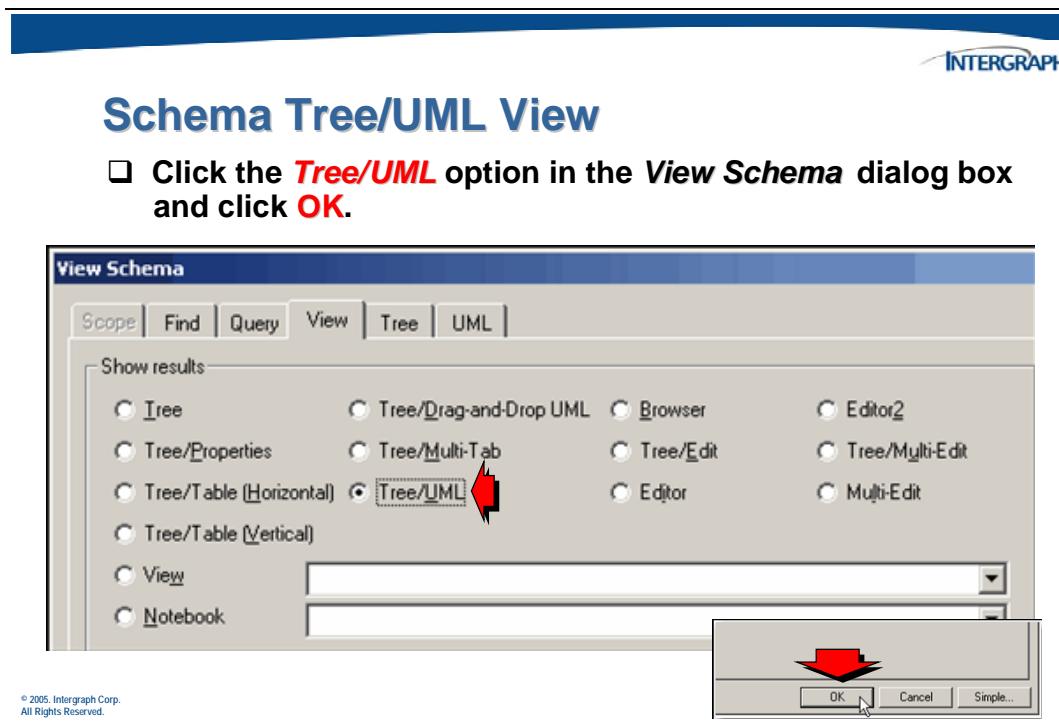
Because IIInstrument **Implies** IElecPowerConsumer, it also inherits AverageActiveLoad as though this property was exposed by IIInstrument directly.



2.1.6 Schema Tree/UML View

The **Tree/UML** view is a combination of the standard tree view with a 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.

Click the **View** tab in the *View Schema* dialog box.

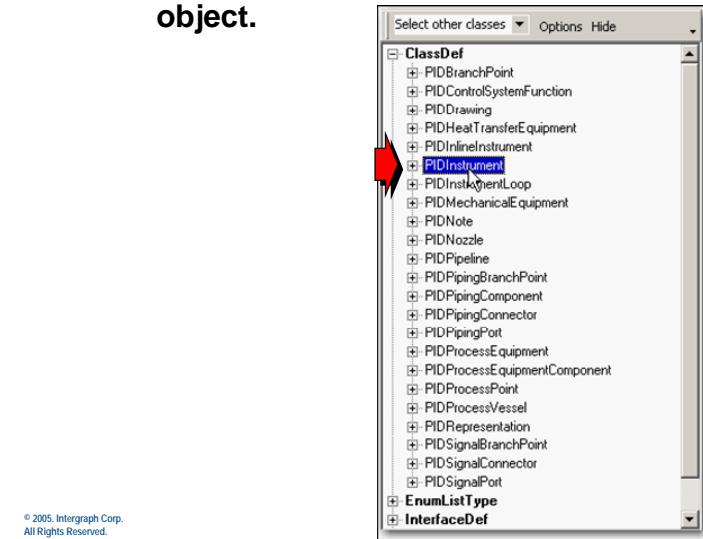


The UML view is useful for viewing relationships in the schema graphically. You can follow relationships by clicking objects, which is a lot like clicking links in your Web browser to explore Web sites. This view is also useful for visualizing relationship definitions that are not displayed elsewhere.



Schema Tree/UML View

- Expand the **ClassDef** node in the tree view and select an object.

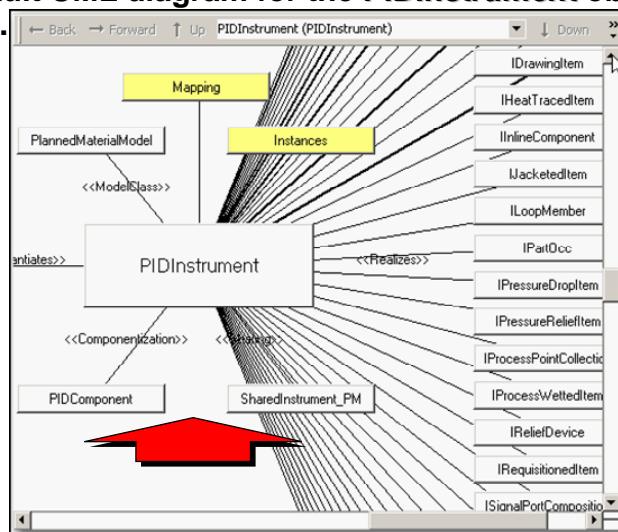


To view the UML for an object, click the object in the tree view.

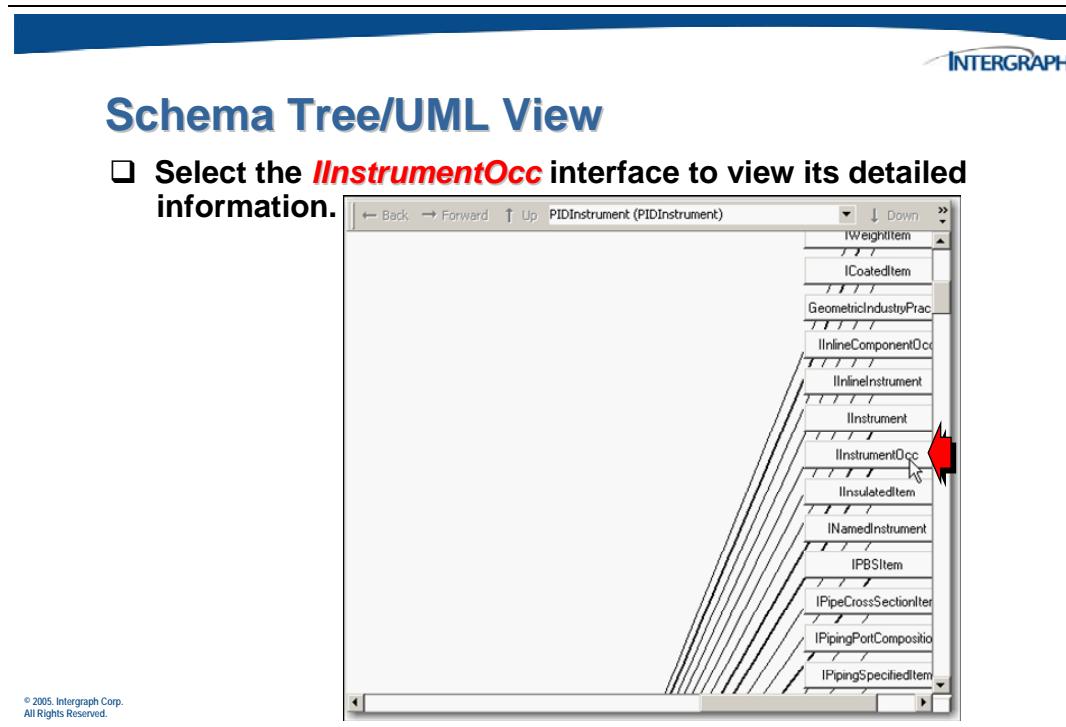


Schema Tree/UML View

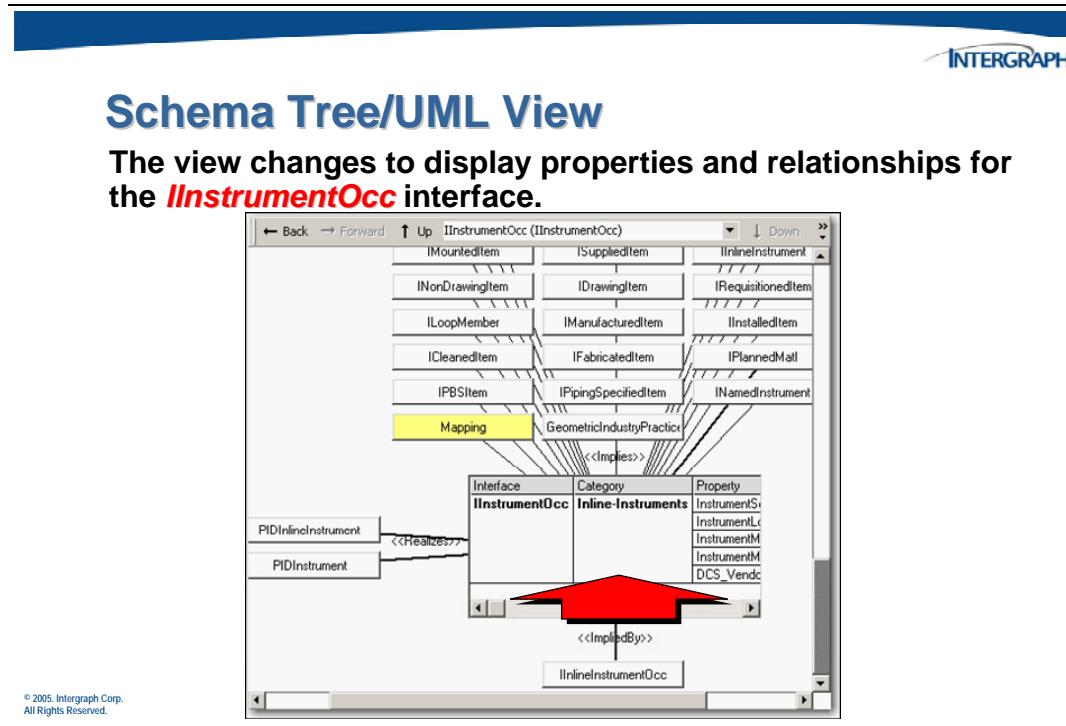
- The default UML diagram for the **PIDInstrument** object displays.



You can use the Tree/UML view to see details of any related edges.



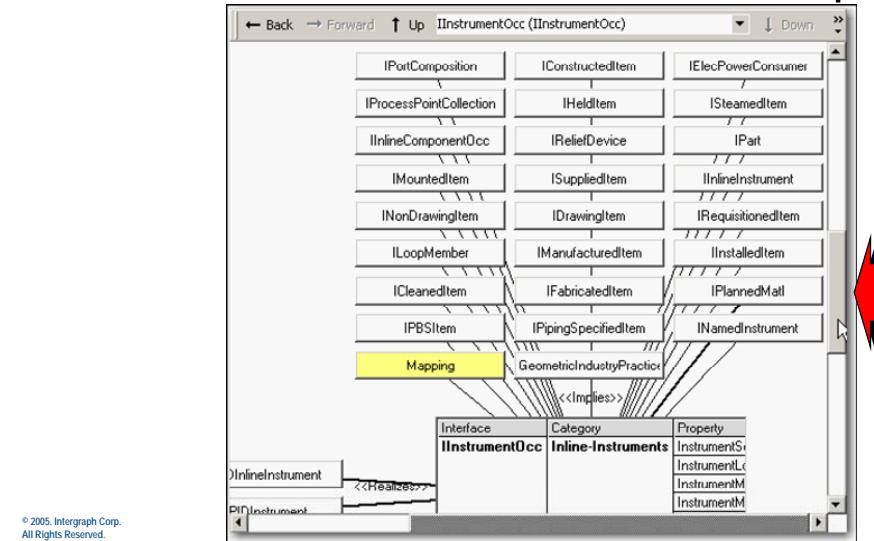
Each type of relationship, such as <<Realizes>> or <<Implies>>, is shown inside angle brackets between the selected object and other classes and interfaces in the UML view.





Schema Tree/UML View

- Use the scroll bar to view all of the relationship information.

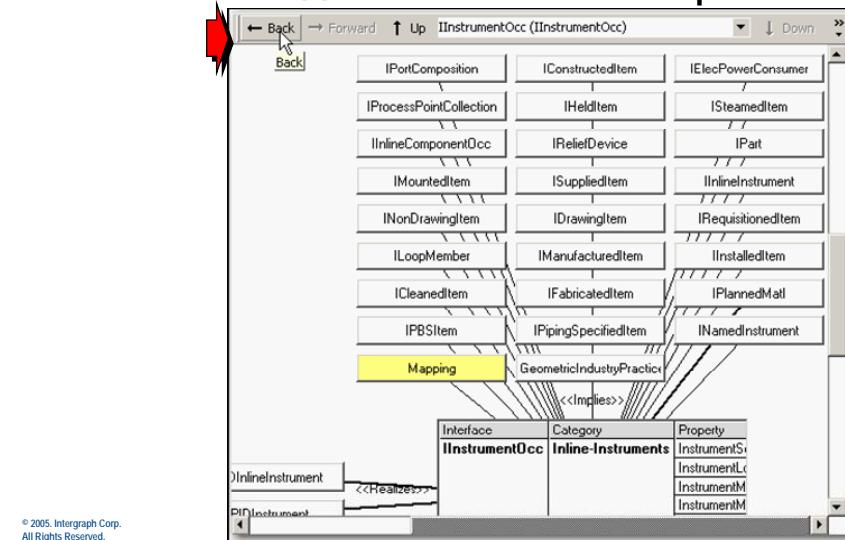


To go back to the last UML view, click **Back** on the toolbar.



Schema Tree/UML View

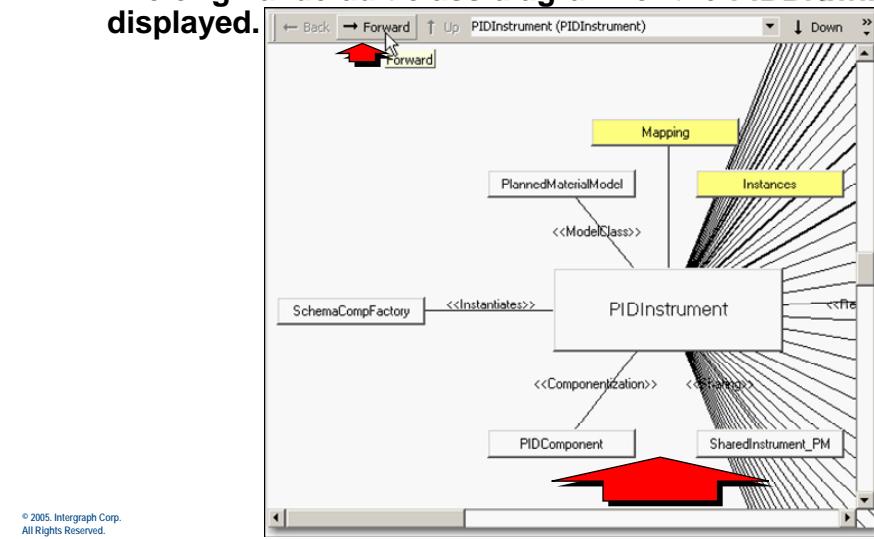
- Use the **Back** button to return to the previous diagram.





Schema Tree/UML View

The original default class diagram for the *PIDDrawing* object is displayed.

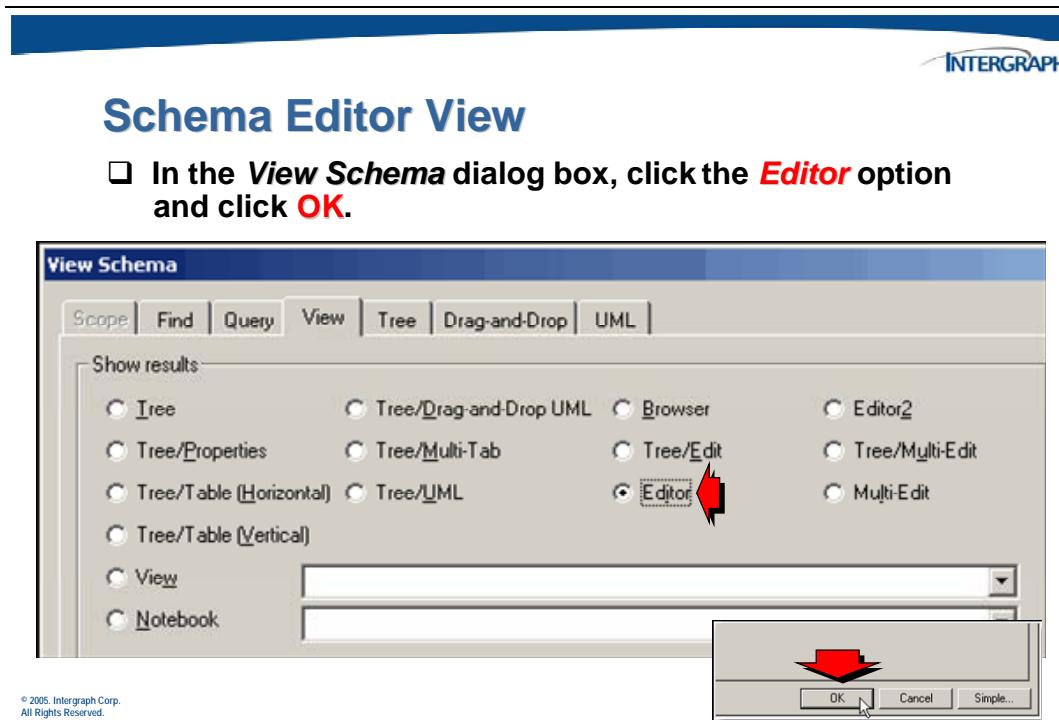


© 2005, Intergraph Corp.
All Rights Reserved.

2.1.7 Schema Editor View

In the Schema Editor, you can create new objects, such as classes, in the Framework schema. To add objects to the schema, you can use the **Editor** view.

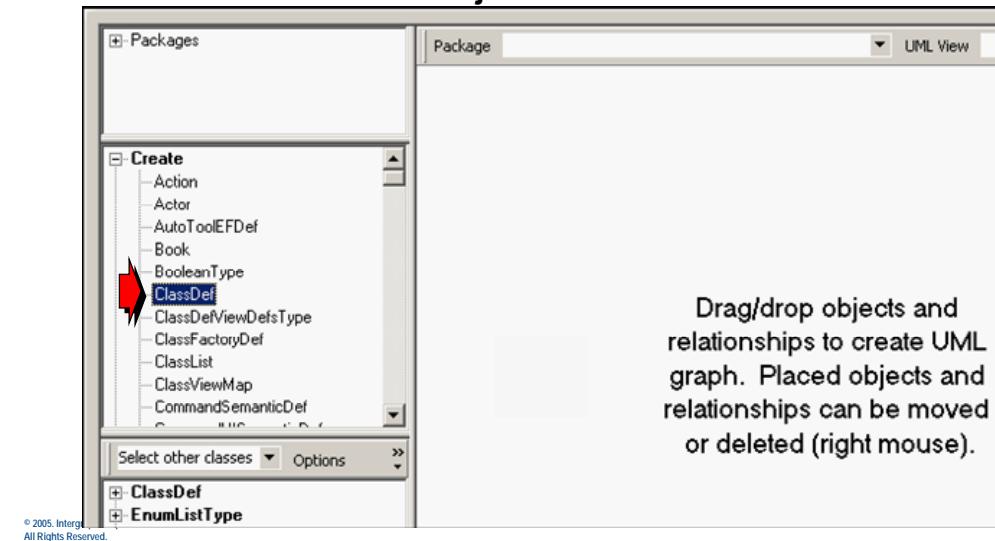
Click the **View** tab in the *View Schema* dialog box.





Schema Editor View

- Click the **ClassDef** object in the Create tree.

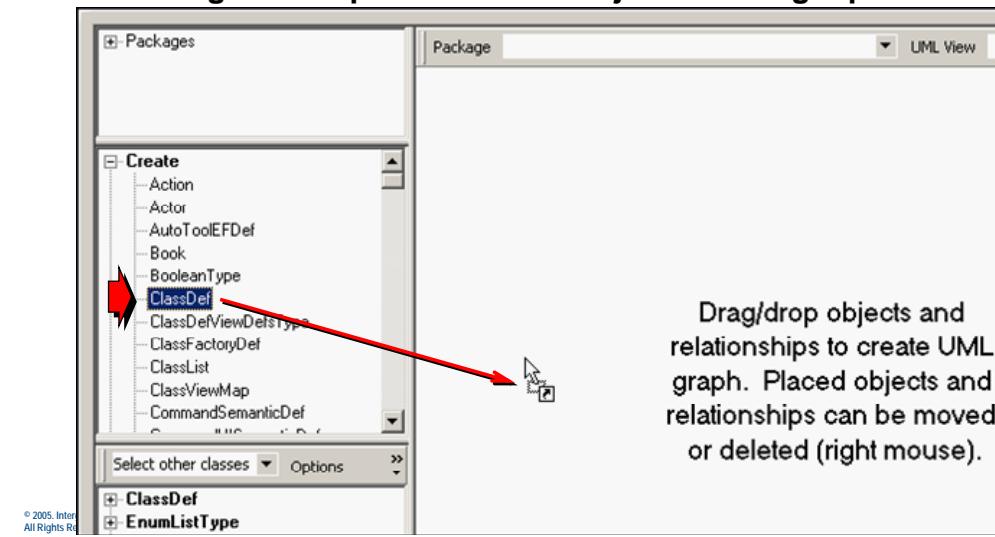


To create a new object in the editor view, drag the object type from the **Create** tree to the UML view.

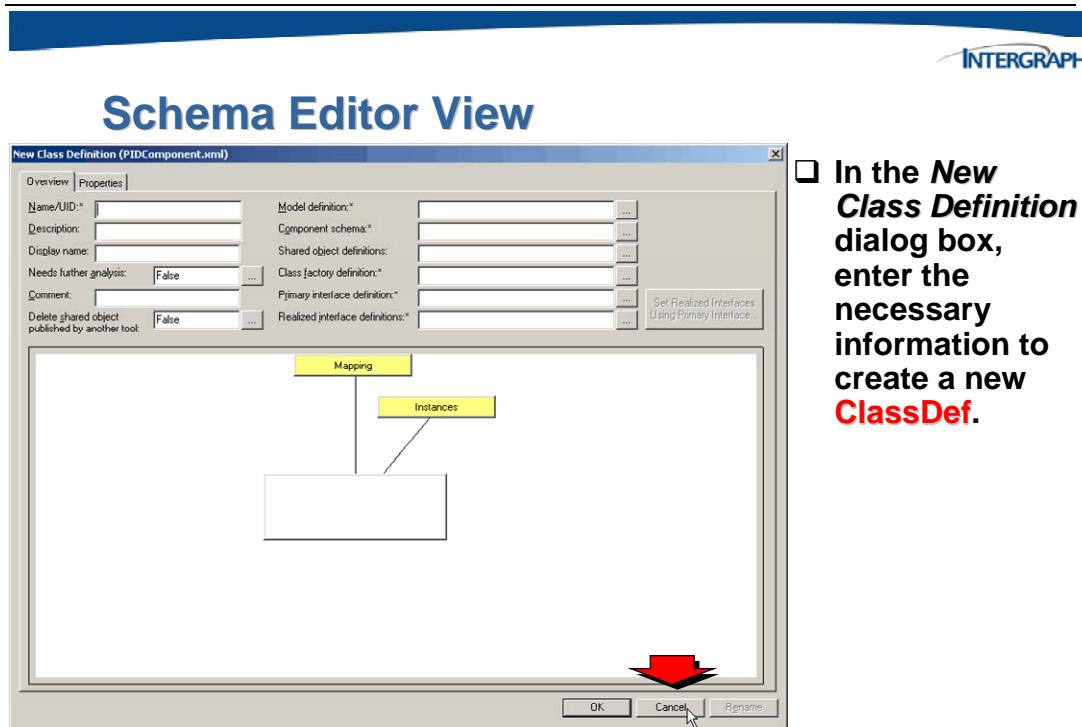


Schema Editor View

- Drag and drop the **ClassDef** object to the right pane.



When you drag an object type to the UML view, the **New** dialog box for that object type appears. Only those objects that have **New** dialog boxes defined for them in the Schema Editor appear in the **Create** tree.



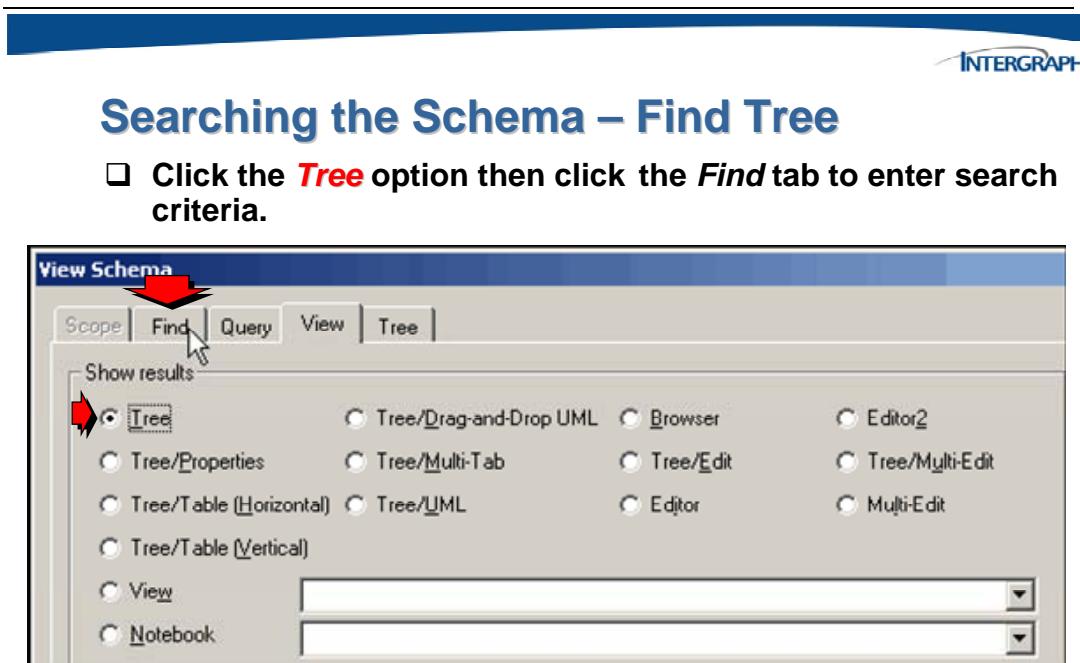
- In the **New Class Definition** dialog box, enter the necessary information to create a new **ClassDef**.

2.2 Finding Schema Objects

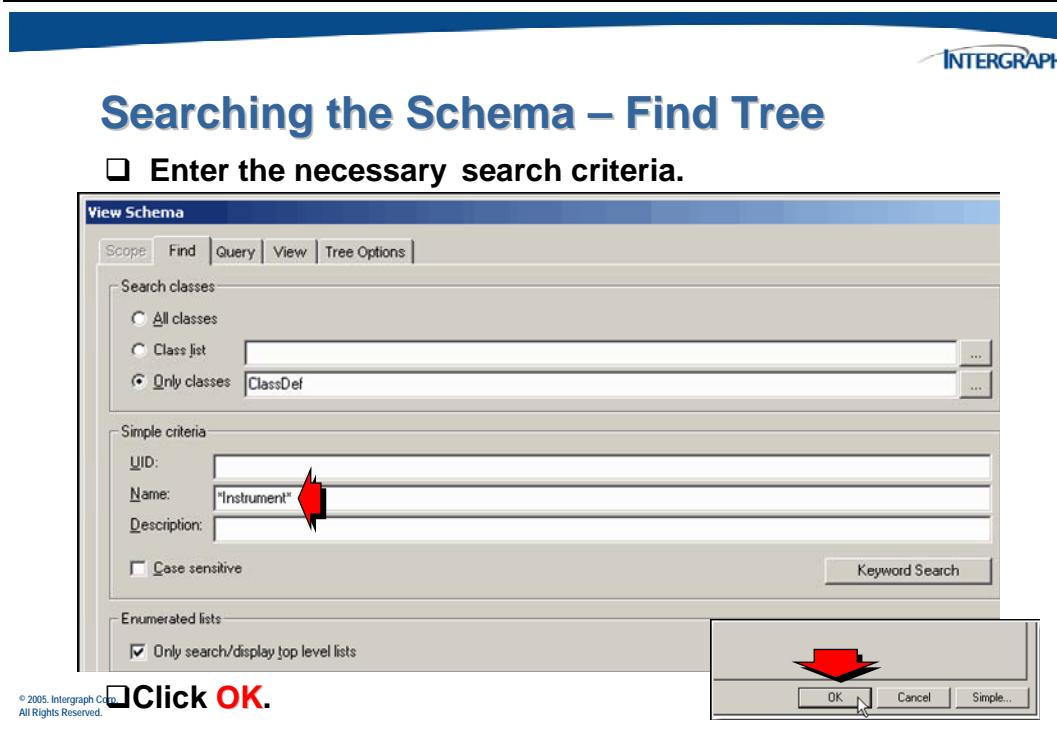
In the Schema Editor, you can locate particular objects in the schema, meta schema, tool schemas, or data files by defining search criteria for the object that you want to find, including the name and description of the object and its UID.

2.2.1 View Schema Find Tab

To find an object in the Schema Editor, click the **Find** tab from the *View Schema* dialog box. This allows you to define search criteria and to limit the scope of your search to the schema, meta schema, tool schemas, data files, or a combination of these.



The **View Schema** will change to display the **Find** dialog box. 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. The “*” character can be used to perform wildcard searches.



The following defines the fields used the *Find* dialog box:

- All classes** - Searches all classes in the Schema Editor. This is the only option available if you access the **Find** tab from the **Find** dialog box.
- Class list** - Allows you to select the class list that you want to include in your search. To select the class list that you want, click . This option is only available if you access the **Find** tab from one of the **View** dialog boxes.
- Only classes** - Allows you to select the individual classes that you want to include in your search. To select from a list of available classes, click . This option is only available if you access the **Find** tab from one of the **View** dialog boxes. This option is also selected by default when you access the **Find** tab from the one of the **View** dialog boxes.
- UID** - Type all or part of the unique identifier of the object that you want to find. You can use an asterisk (*) as a wildcard character in your search.
- Name** - Type all or part of the name of the object that you want to find. You can use an asterisk (*) as a wildcard character in your search.
- Description** - Type all or part of the description of the object that you want to find. You can use an asterisk (*) as a wildcard character in your search.

- Case sensitive** - Specifies whether you want the search to be case sensitive.
- Organize by class** - Displays the search results organized by class. If you clear this checkbox, the search results are presented in list format.
- Only display top level lists** - Specifies whether you want the software to display results from top-level lists only.

A Schema Tree view displays the results of the search. Expand nodes in the tree to see related edges.



Searching the Schema – Find Tree

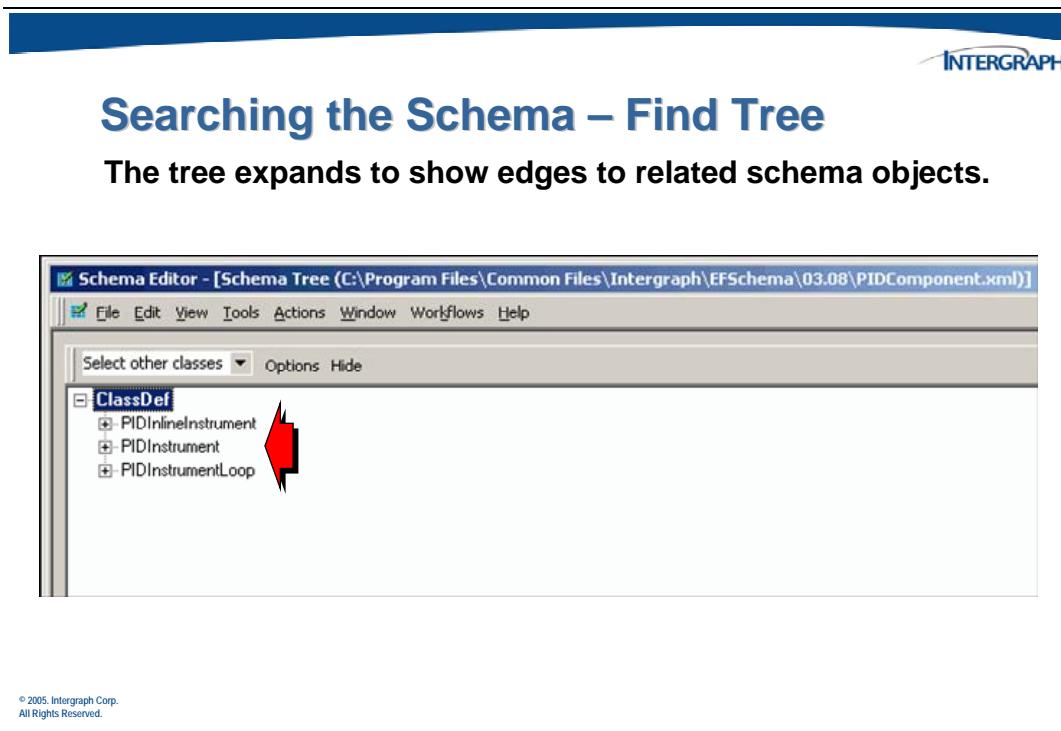
Click the + next to **ClassDef** to expand the tree.

Select other classes ▾ Options Hide

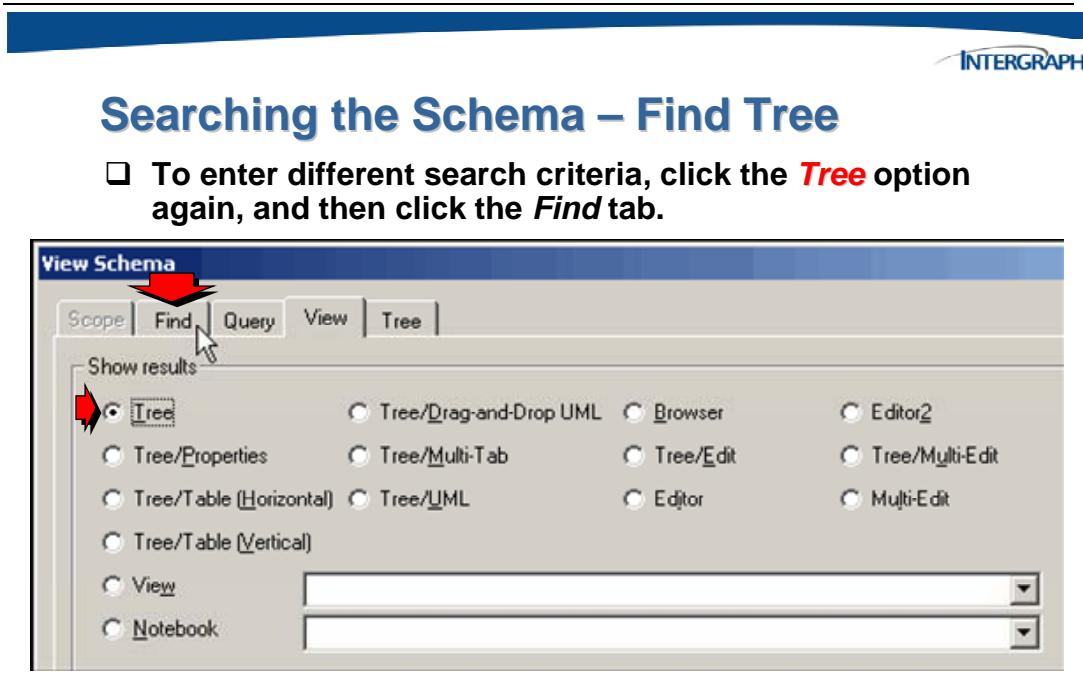
 **ClassDef**

© 2005. Intergraph Corp.
All Rights Reserved.

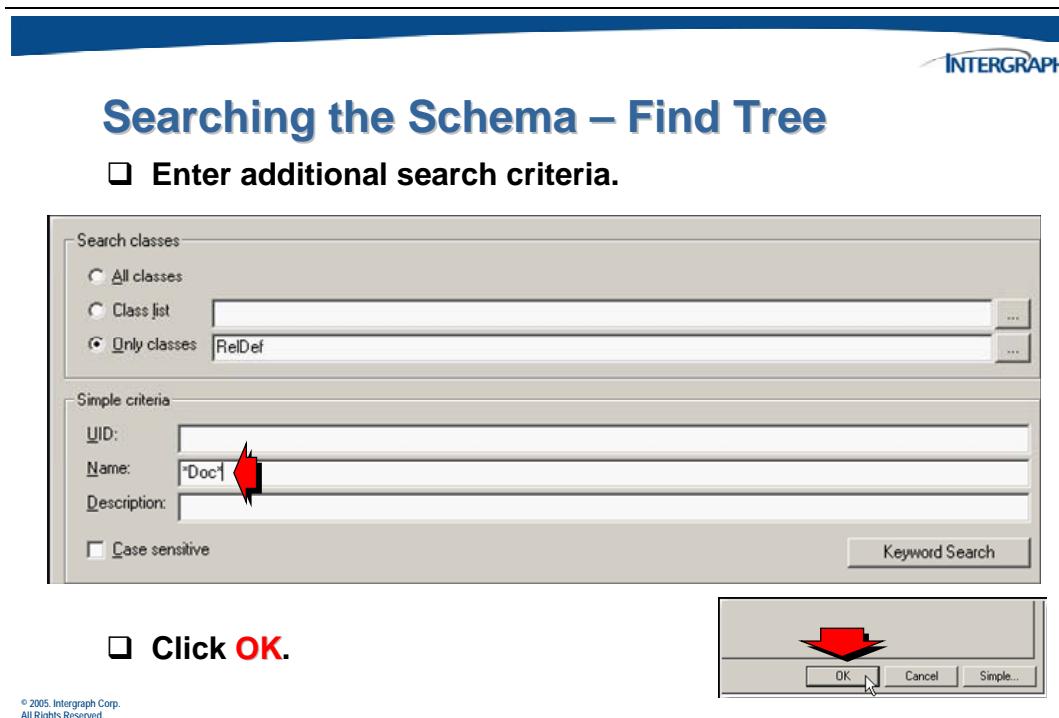
Continue to drill down in the tree view.



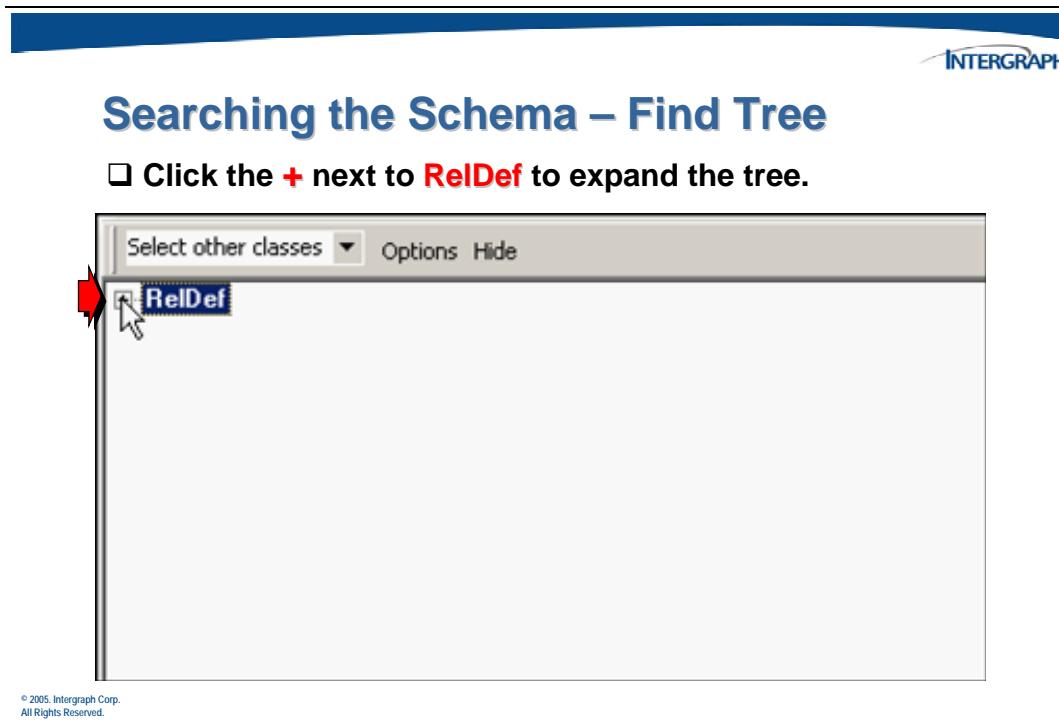
Close the *Schema Tree* view window and bring up the *View Schema* dialog box. Click the **Find** tab in the *View Schema* dialog.



Then, enter any additional search criteria in the *View Schema* dialog.



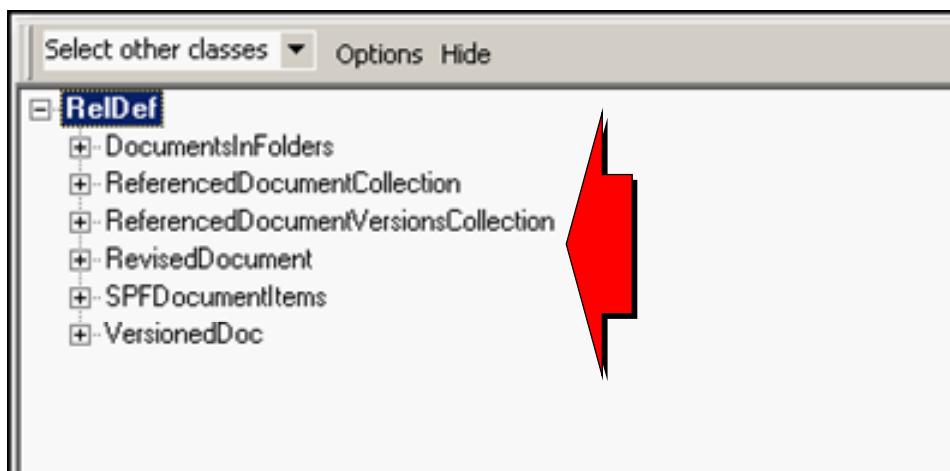
In this example, the search is limited to the **RelDef** object class and any name that contains the phrase **Doc**. A Schema Tree view displays the search results. Again, expand nodes in the tree to see related edges.





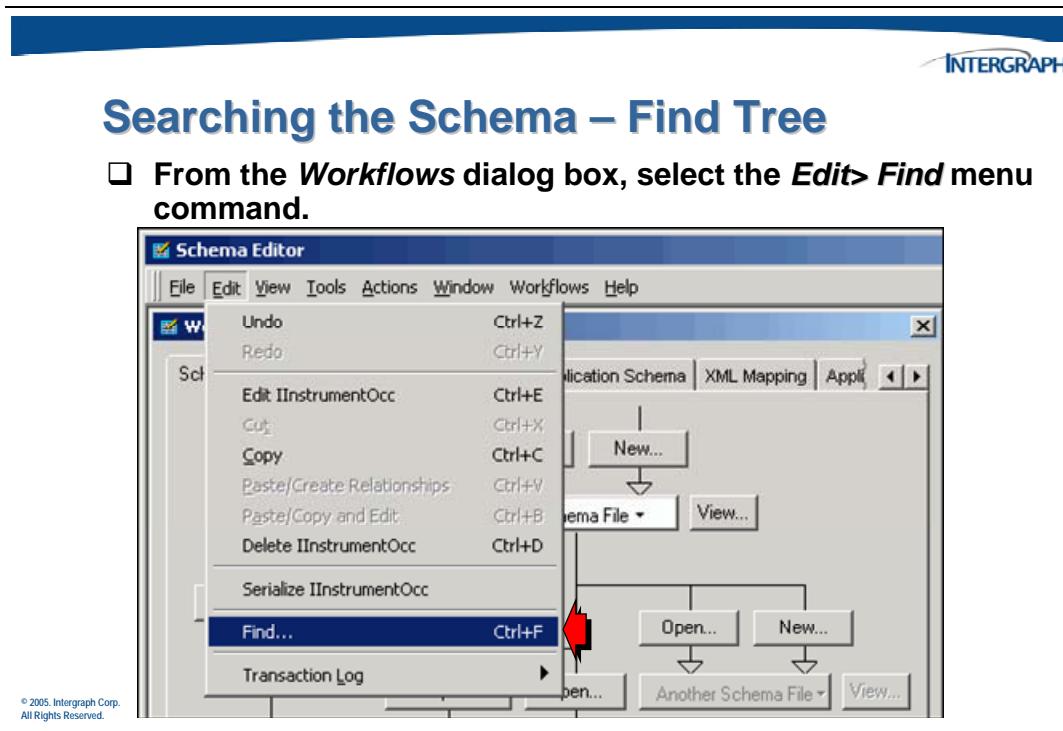
Searching the Schema – Find Tree

The tree expand to show edges to related schema objects.

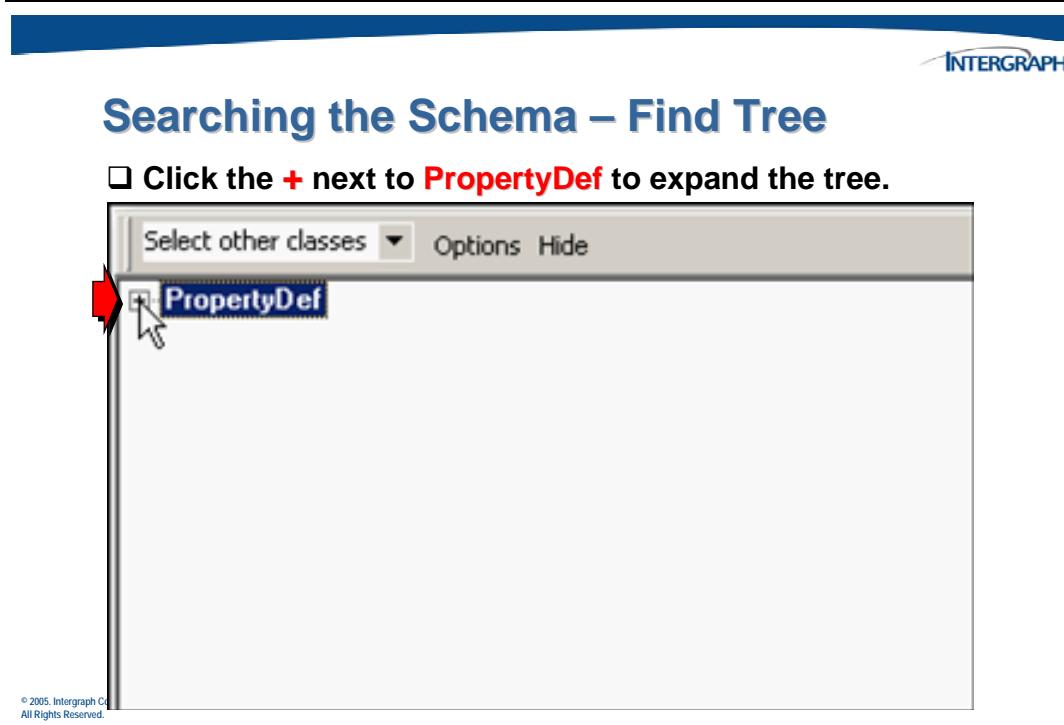
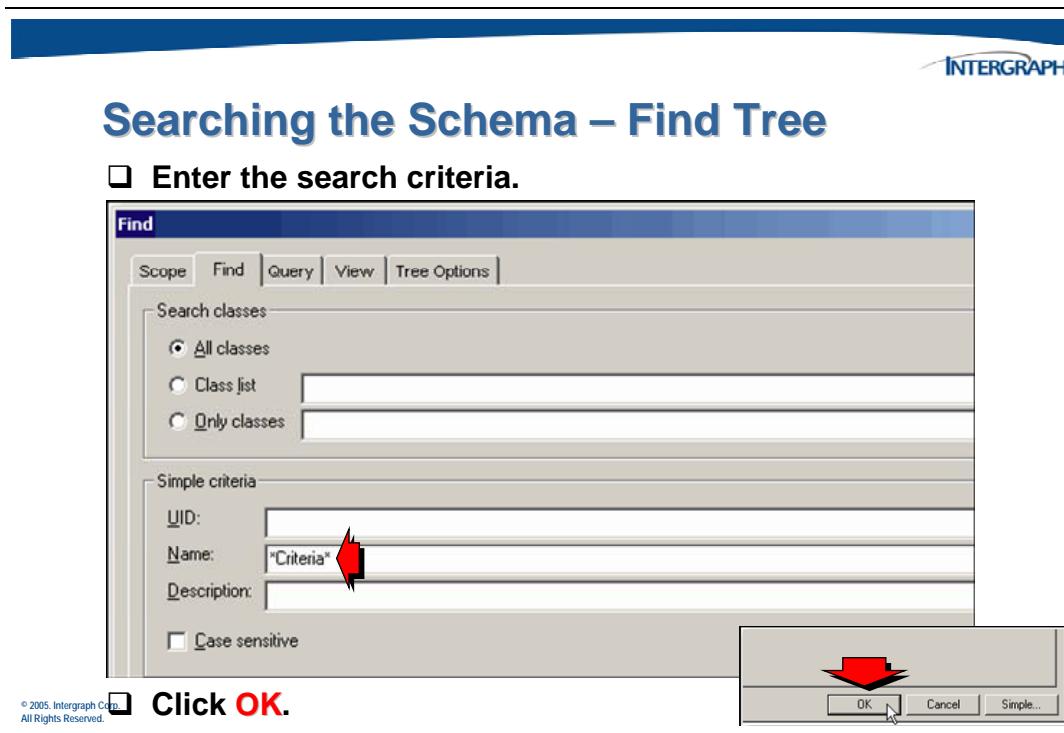


2.2.2 Finding Objects from the Workflows Dialog Box

An alternate method for searching for schema objects is to use the **Find** button in the *Workflows* dialog box.



A *Find* dialog box appears.



A Schema Tree view displays the search results.

INTERGRAPH

Searching the Schema – Find Tree

The tree expands to show edges to related schema objects.

Select other classes Options Hide

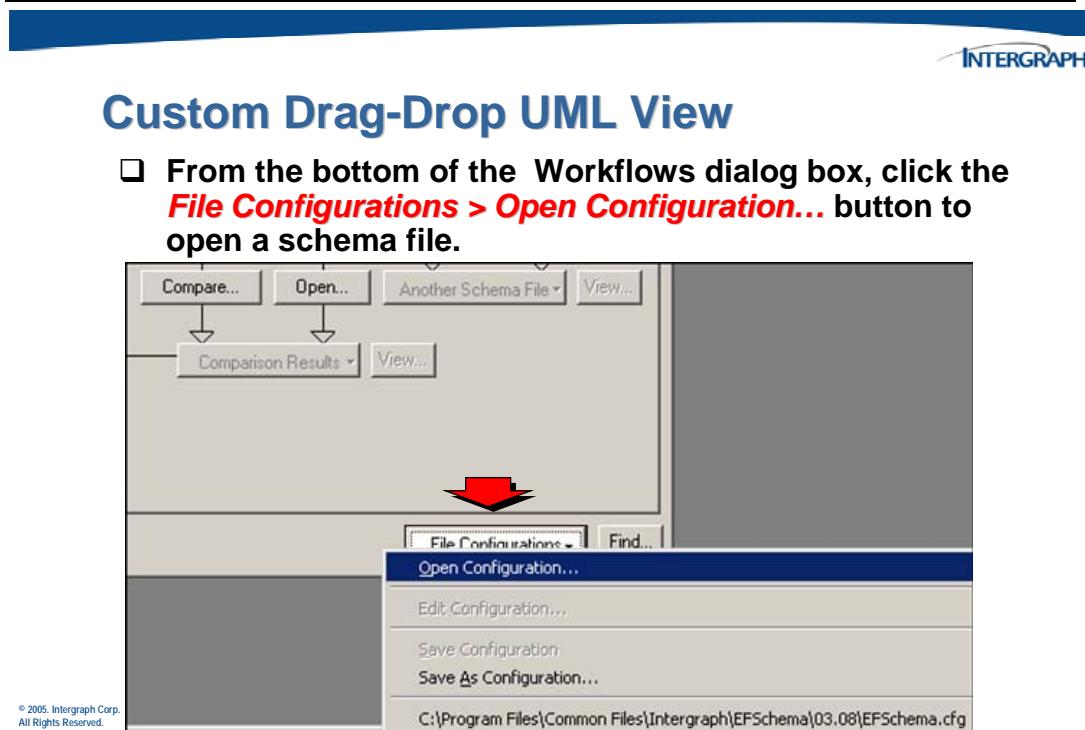
- [-] PropertyDef
 - [+] DesignCriteriaFluidState
 - [+] DesignCriteriaMassFlowRate
 - [+] DesignCriteriaMetalTemperature
 - [+] DesignCriteriaPower
 - [+] DesignCriteriaPressure
 - [+] DesignCriteriaPressure2
 - [+] DesignCriteriaShortTermDuration
 - [+] DesignCriteriaShortTermPressure
 - [+] DesignCriteriaShortTermTemperature
 - [+] DesignCriteriaSpecificGravityMassBasis
 - [+] DesignCriteriaSpeed
 - [+] DesignCriteriaSteamOutPressure
 - [+] DesignCriteriaSteamOutRequirement
 - [+] DesignCriteriaSteamOutTemperature
 - [+] DesignCriteriaTemperature
 - [+] DesignCriteriaTemperature2
 - [+] DesignCriteriaVacuumPressure
 - [+] DesignCriteriaVacuumTemperature
 - [+] DesignCriteriaVapourPressure
 - [+] DesignCriteriaVentingTemperature
 - [+] DesignCriteriaViscosity

© 2005, Intergraph Corp.
All Rights Reserved.

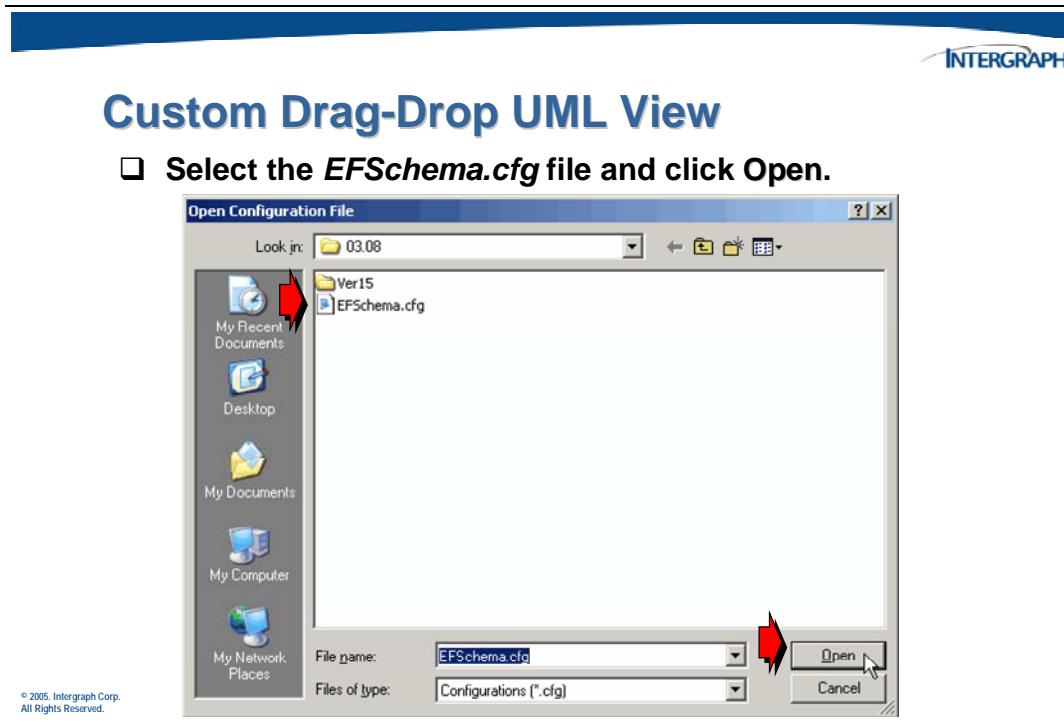
2.3 Custom Drag-Drop UML View

You can search for an object and then view the results in a Tree Drag-Drop UML diagram. You can create your own UML views, name them, and store them as part of a package. In the following example, a custom view will be defined to show the search results.

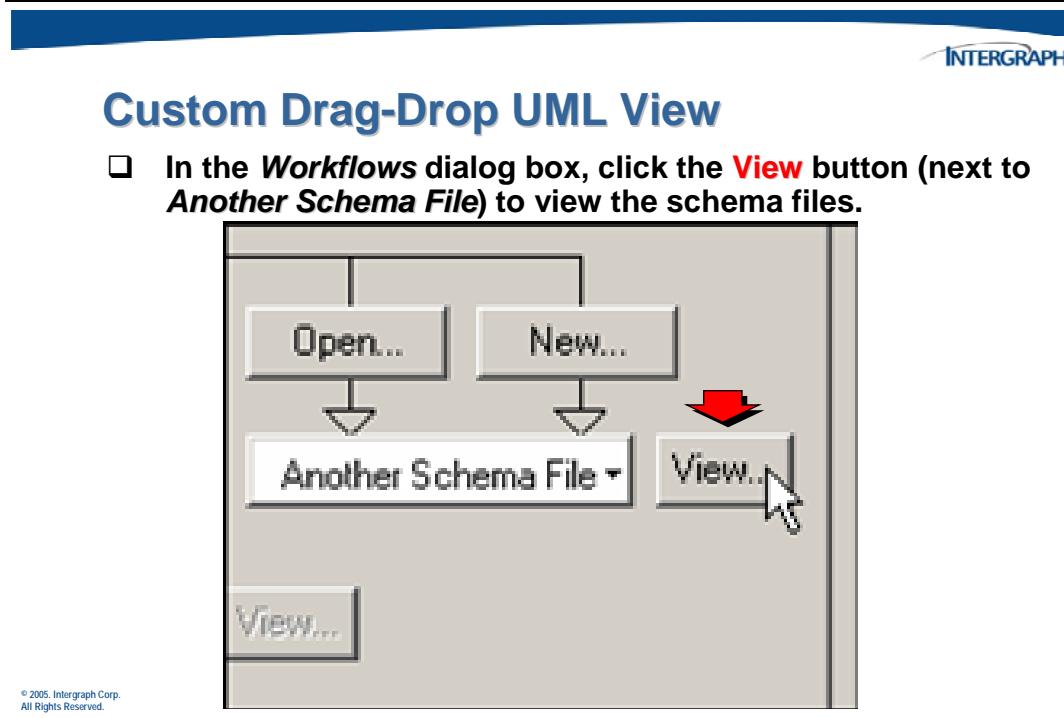
Since the *PIDComponent* schema is used as a read-only file, to save a custom view, open the SmartPlant master schema, **EFSchema.xml** using the *EFSchema.cfg* configuration.



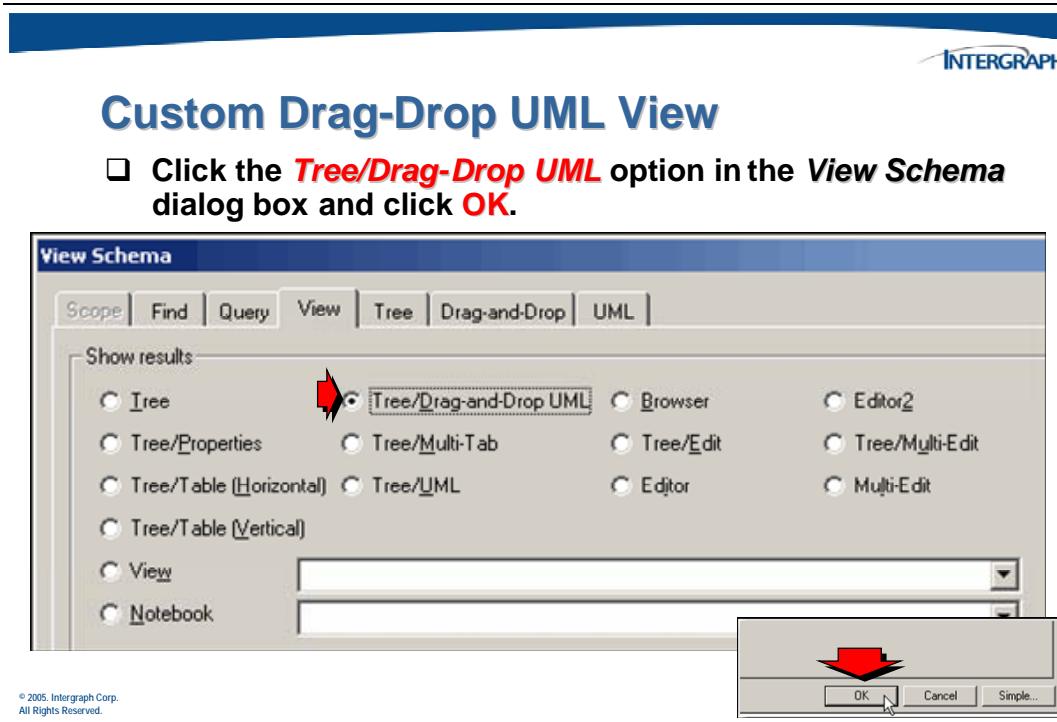
The *Open Configuration File* browser dialog will display.



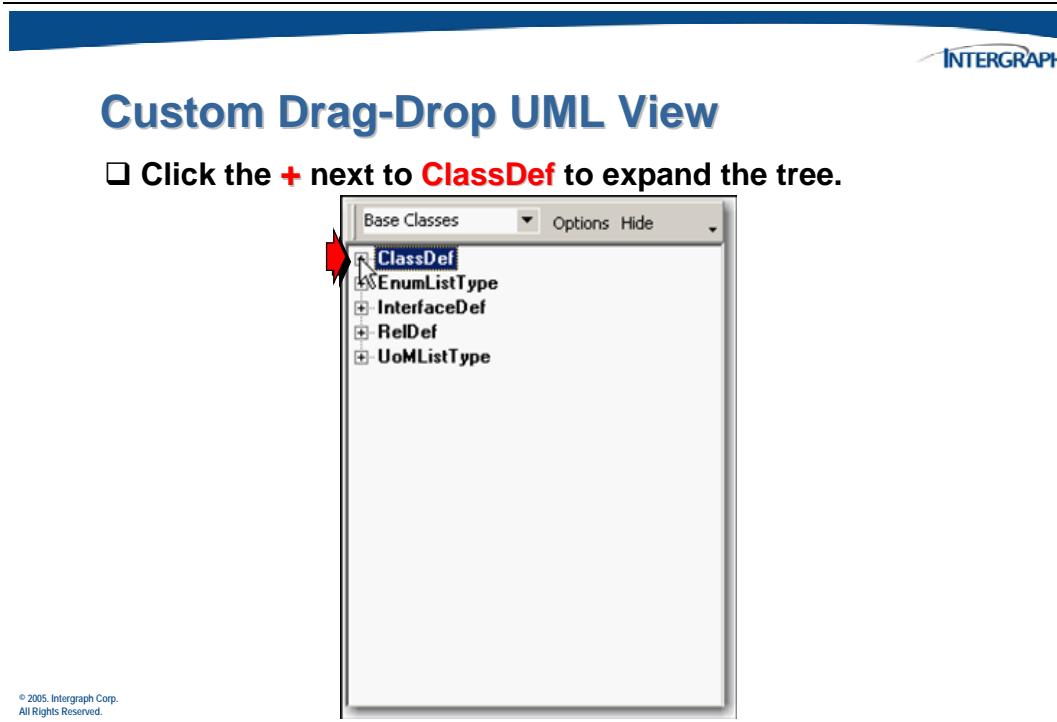
Once the schemas have been opened by the configuration file, select the viewing format.



Select the **Tree/Drag-Drop UML** option in the *View Schema* dialog box.



A Schema Tree/Drag-Drop UML view displays the results.

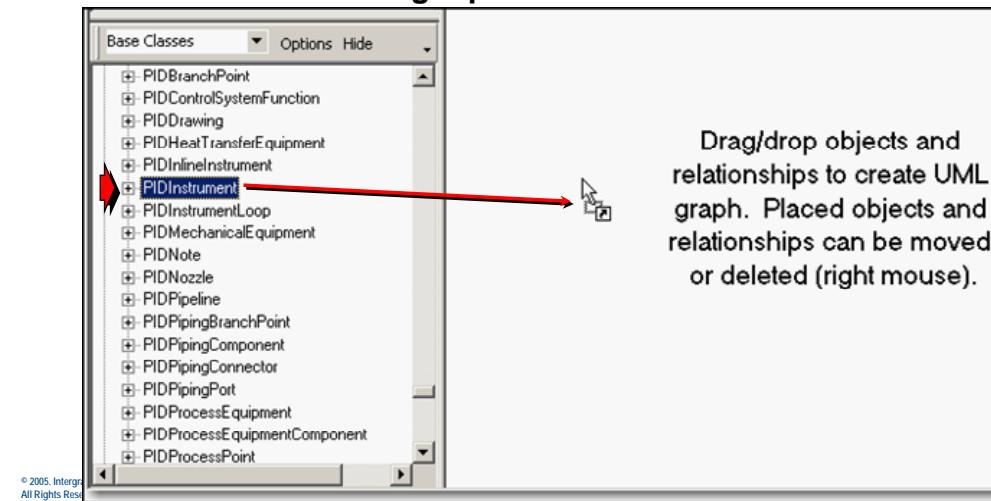


Select the object to be dropped into the UML view.



Custom Drag-Drop UML View

- Drag and drop the selected object, **PIDInstrument**, from the tree view to the right pane.

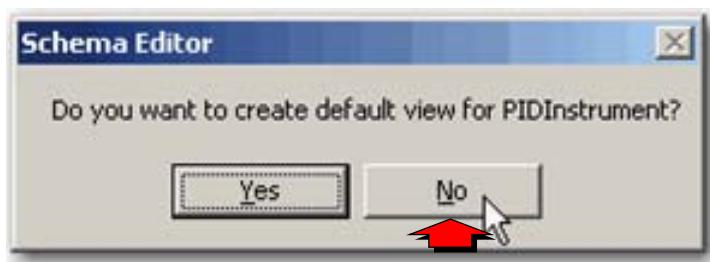


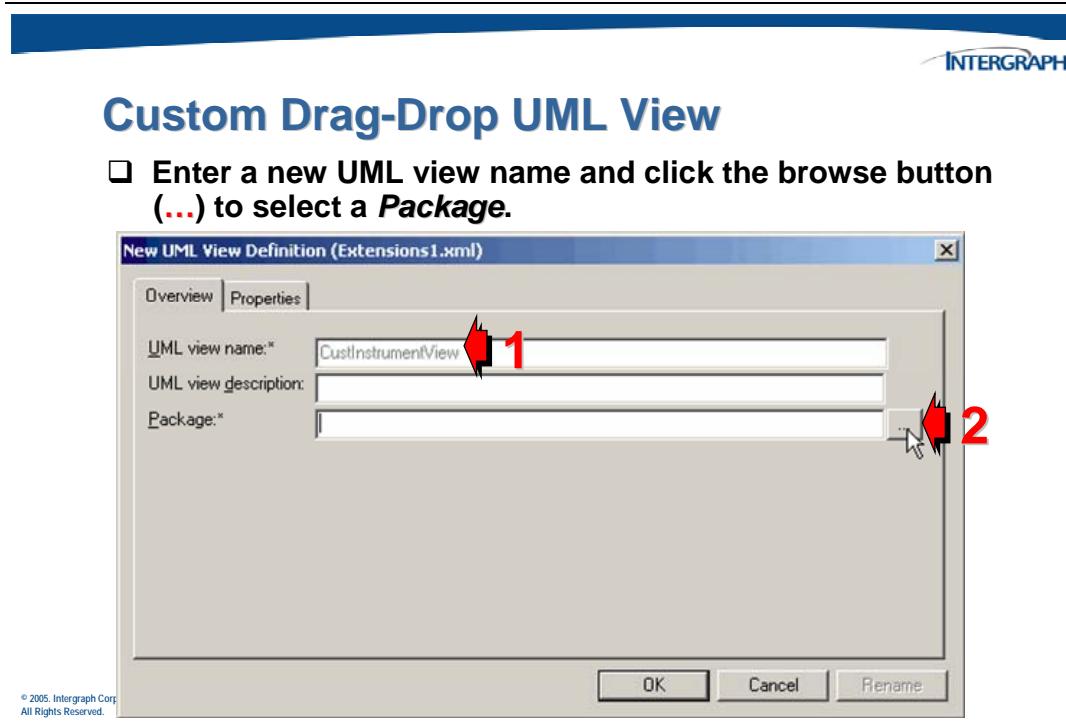
The **Schema Editor** dialog box appears. Click **No** to create a customized UML view.



Custom Drag-Drop UML View

- To create a new UML view, click **No**.

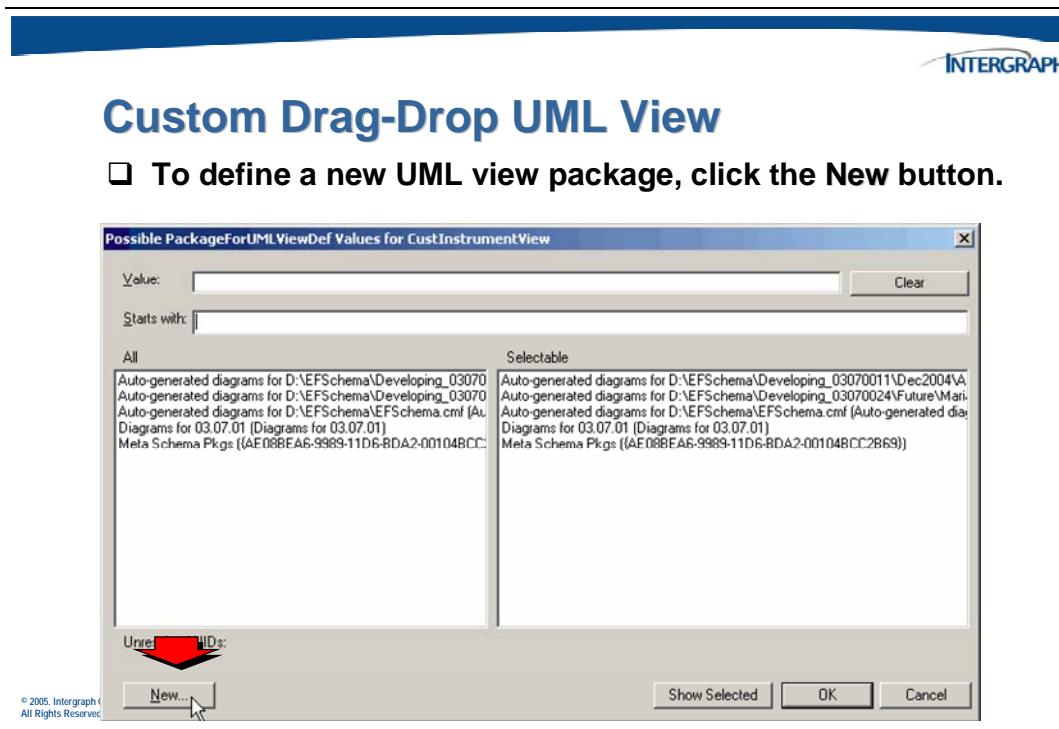




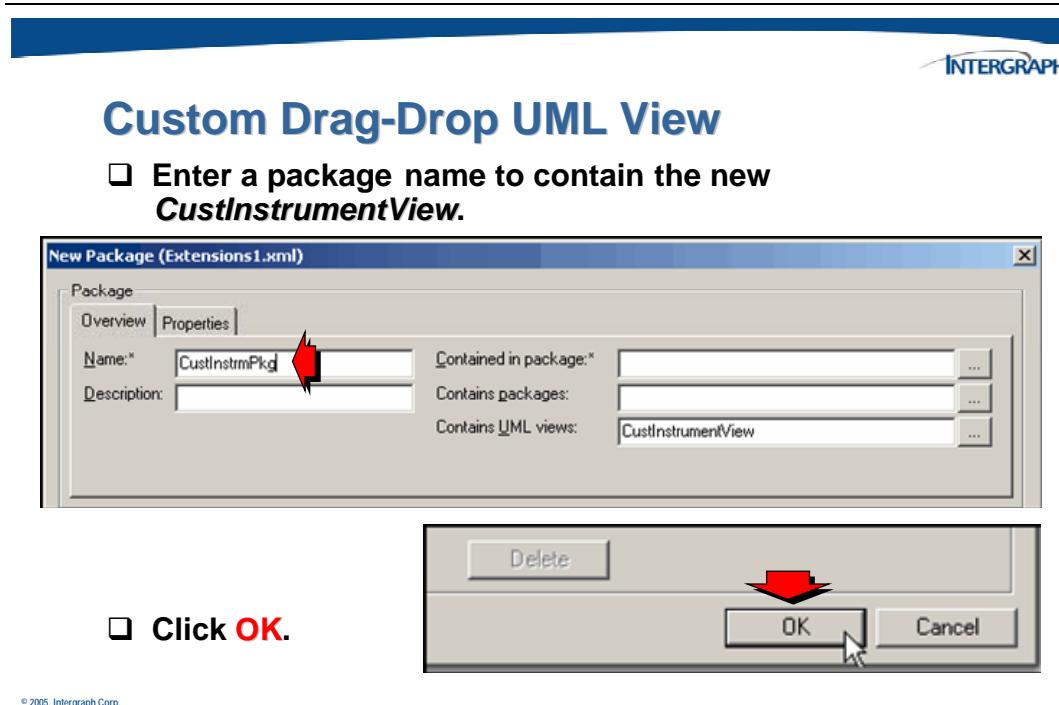
The following defines the fields used the *New UML View Definition* dialog box:

- ❑ **UML view name** – Specifies the name for the new custom UML view. This is a required field.
- ❑ **UML view description** – This is an optional field used to give the new view a description.
- ❑ **Package** – Use the ... to select an existing package in which to store the UML view. You can also create a new package to use in storing your custom UML view.

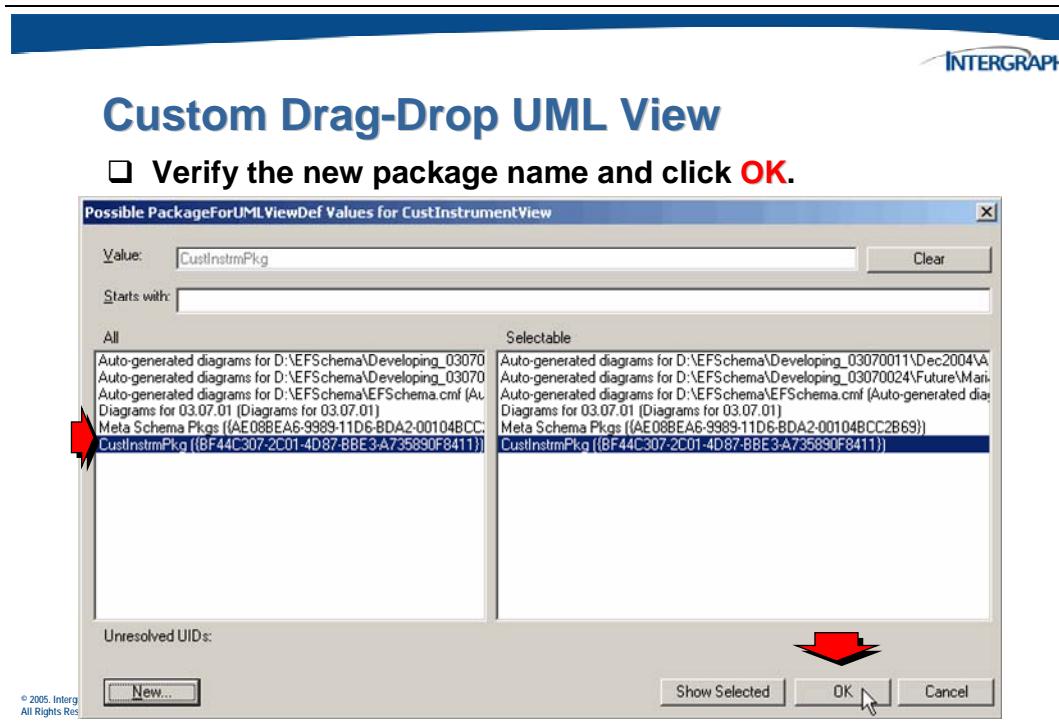
The *Possible PackageForUMLViewDef Values* dialog will display.



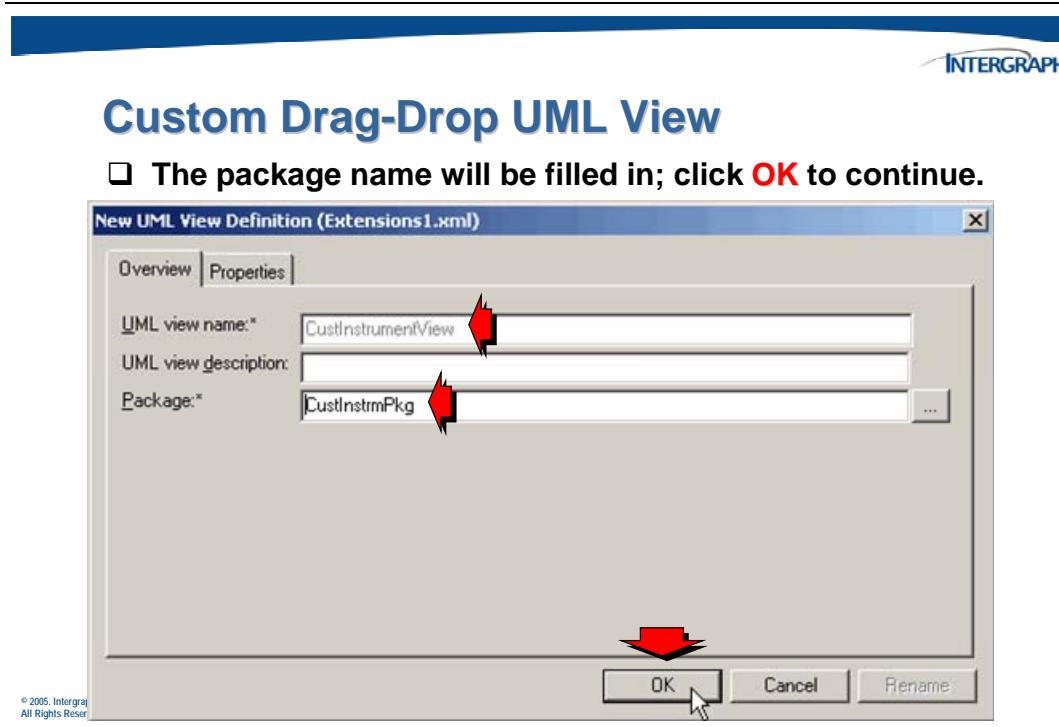
The *New Package* dialog box displays.



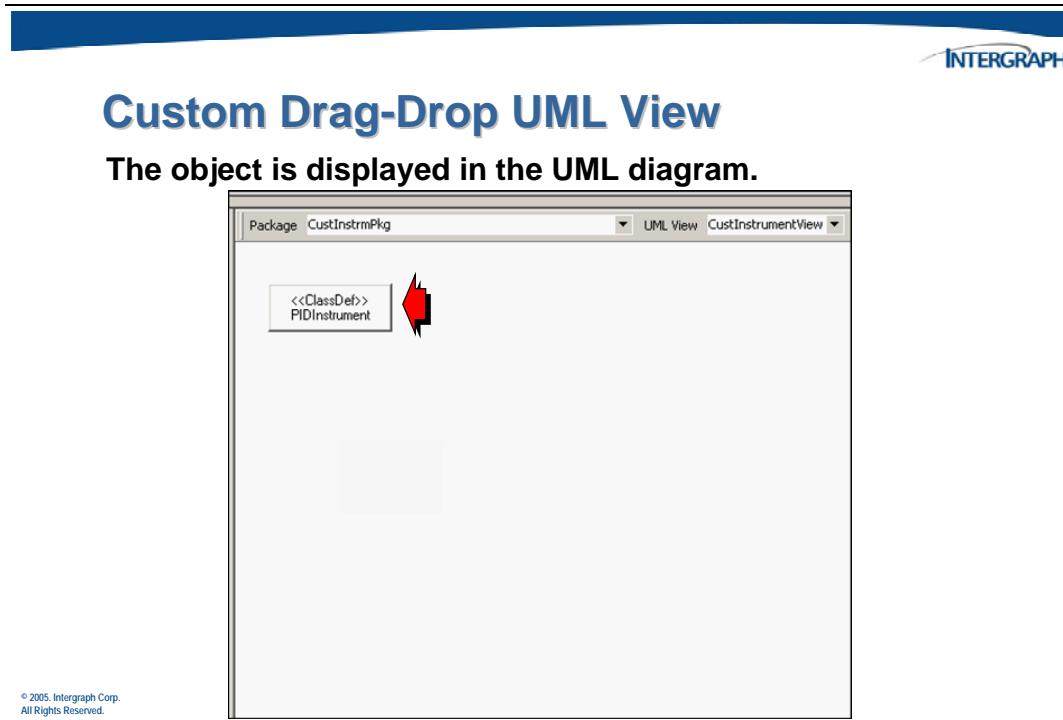
The new package displays in the *Possible PackageForUMLViewDef Values* dialog box.



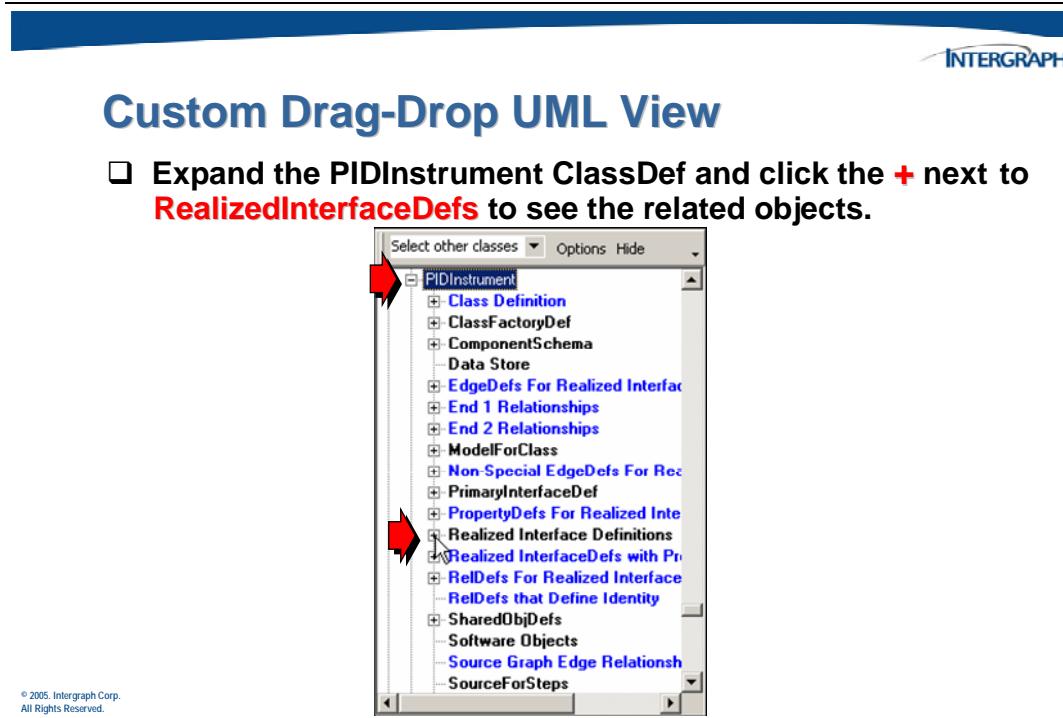
In the *New UML View Definition* dialog box, the new package name appears in the **Package** field.



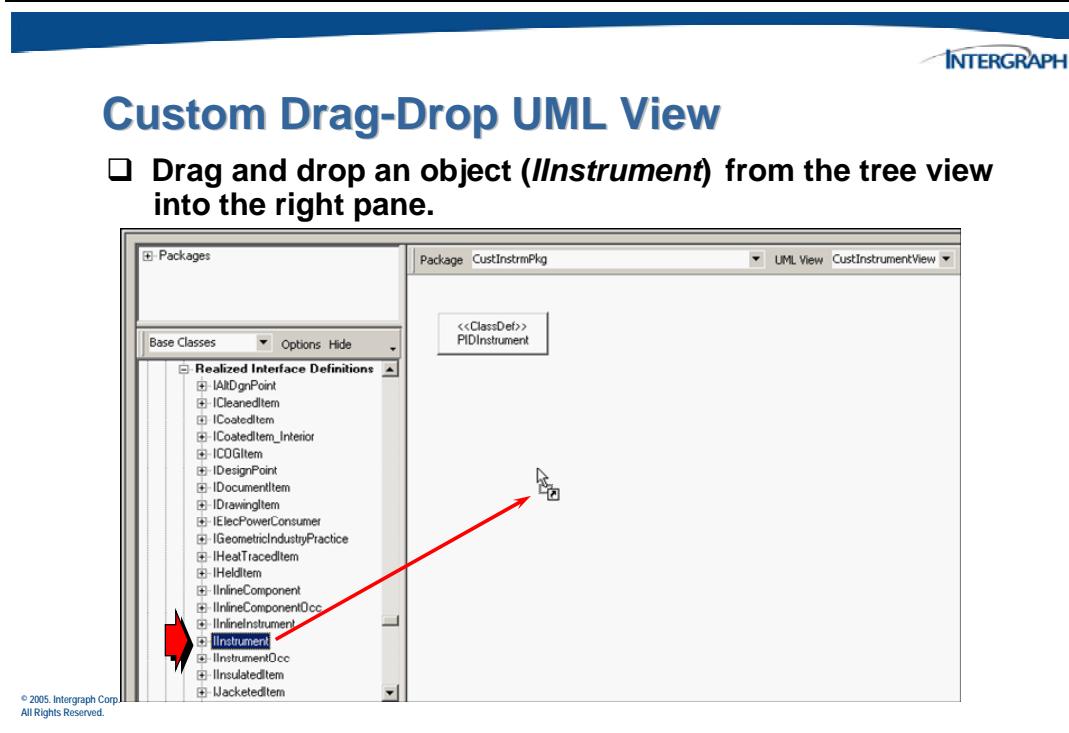
The dropped object, PIDInstrument, appears in the custom *CustInstrumentView* UML view.



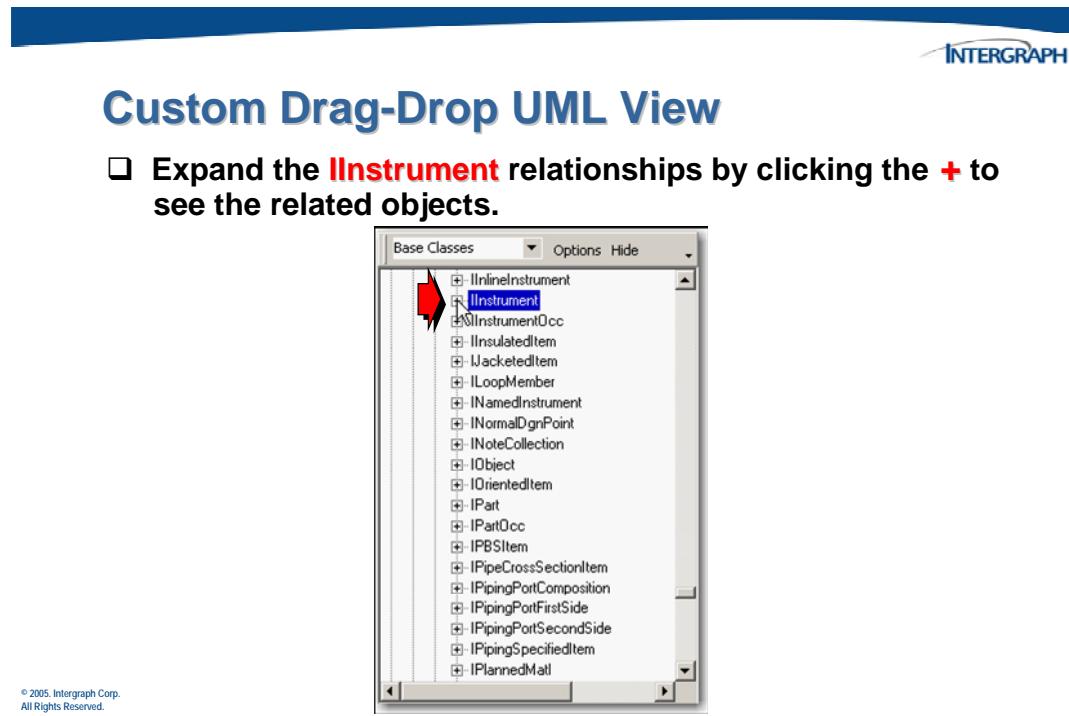
Use the tree to locate other objects to add to the UML view.



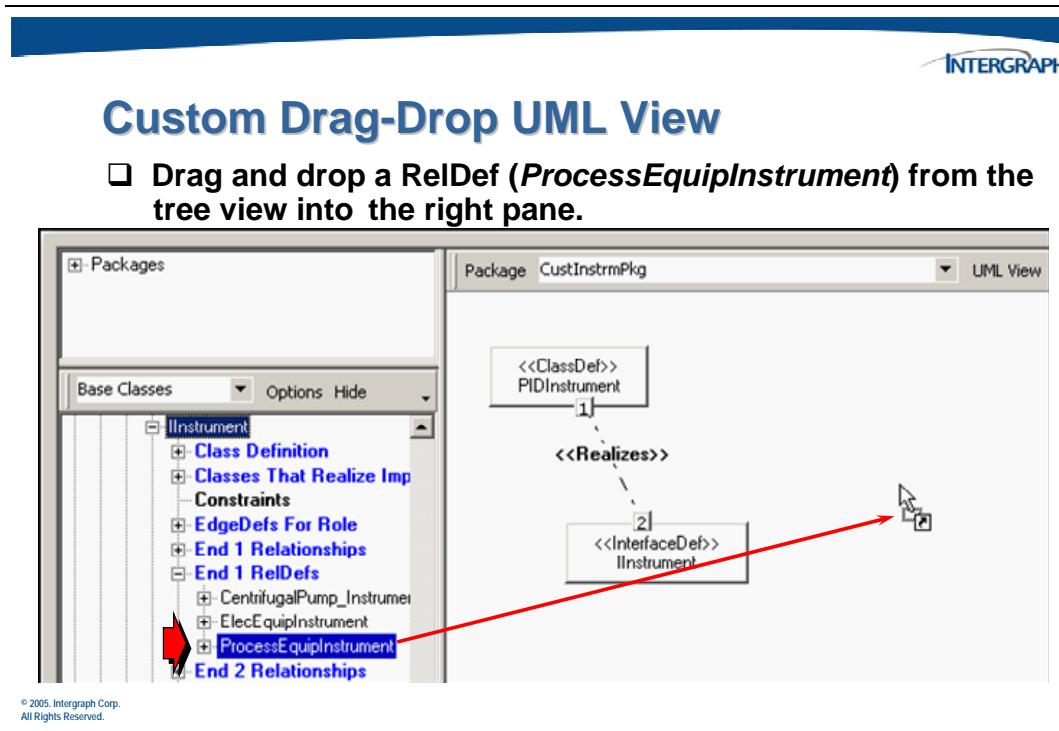
Expand nodes in the tree to see other objects. Select the object to be dropped into the UML view.



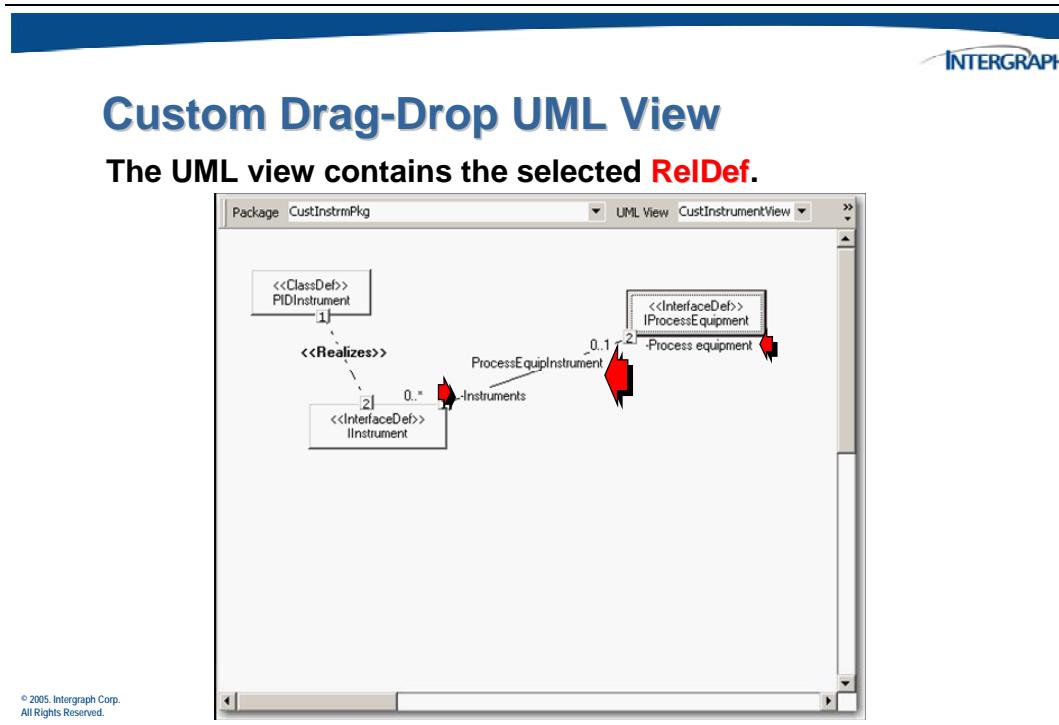
Expand nodes in the tree then select the object to be dropped into the UML view.



Continue to locate objects and drag them into the view.



By dragging and dropping objects from the tree, you are able to define your custom UML view objects and their relationships.

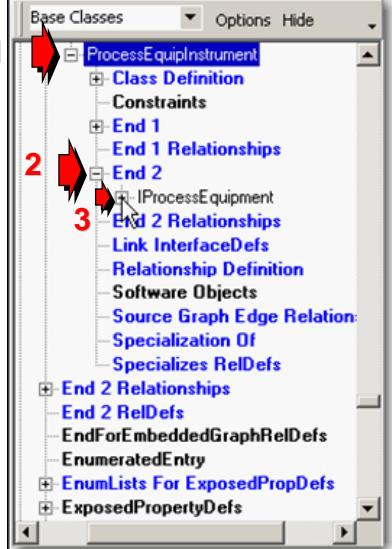


Continue to expand nodes in the tree so that additional objects can be dropped into the custom UML view.

Custom Drag-Drop UML View

Expand the **ProcessEquipInstrument** relationships by clicking the + to see the related objects.

Then click to expand End 2 and the **IProcessEquipment** End2 Interface.

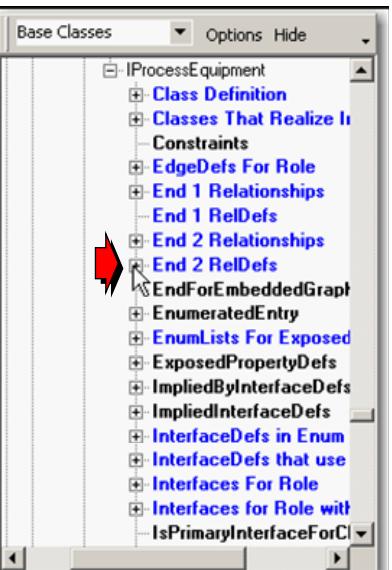


© 2005. Intergraph Corp.
All Rights Reserved.

Expand nodes in the tree to show the related edges.

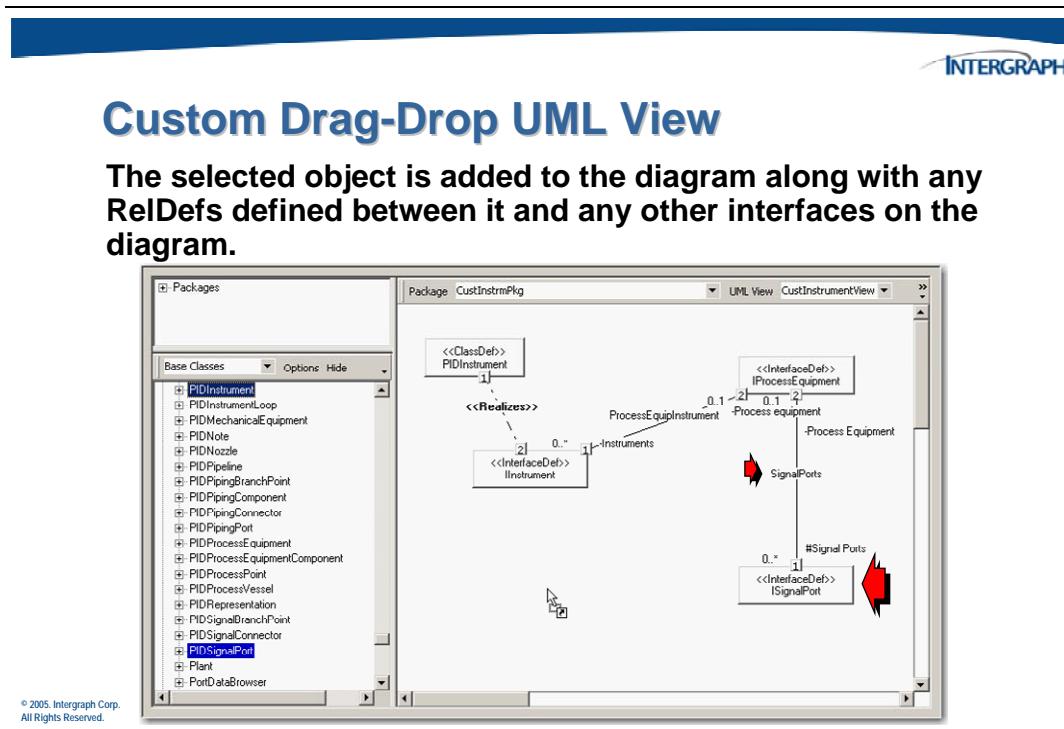
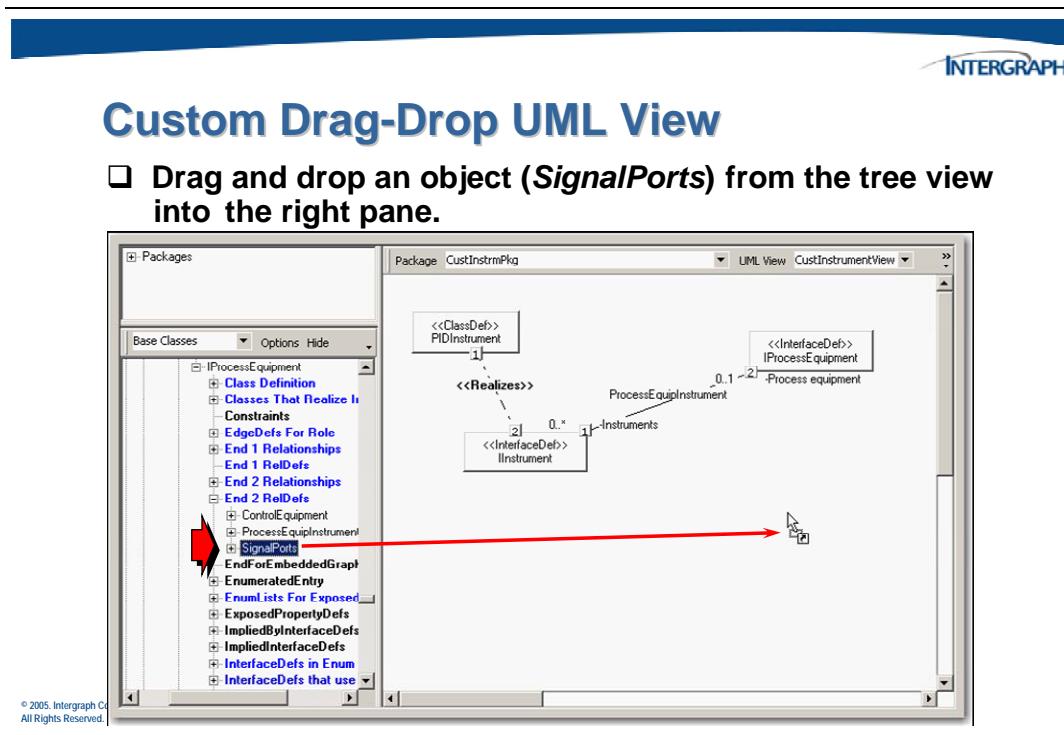
Custom Drag-Drop UML View

Expand the **End 2 RelDefs** relationship by clicking the + to see the related objects.



© 2005. Intergraph Corp.
All Rights Reserved.

Select the next object to be dropped into the UML view.

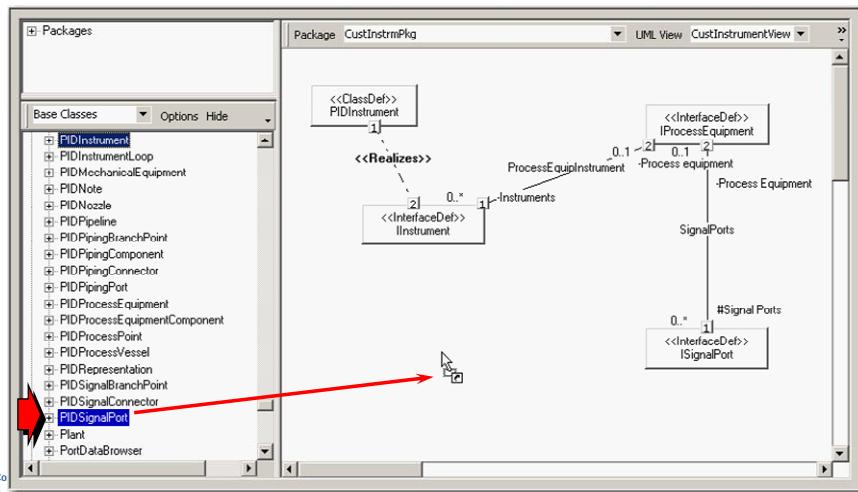


Select another object to be dropped into the UML view.



Custom Drag-Drop UML View

- Drag and drop another object (**PIDSignalPort**) from the tree view into the right pane.

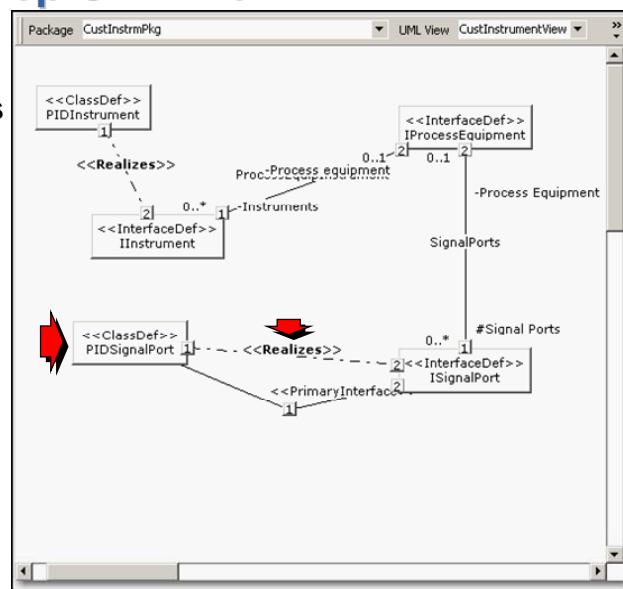


Note that you are building your own custom UML view diagram.



Custom Drag-Drop UML View

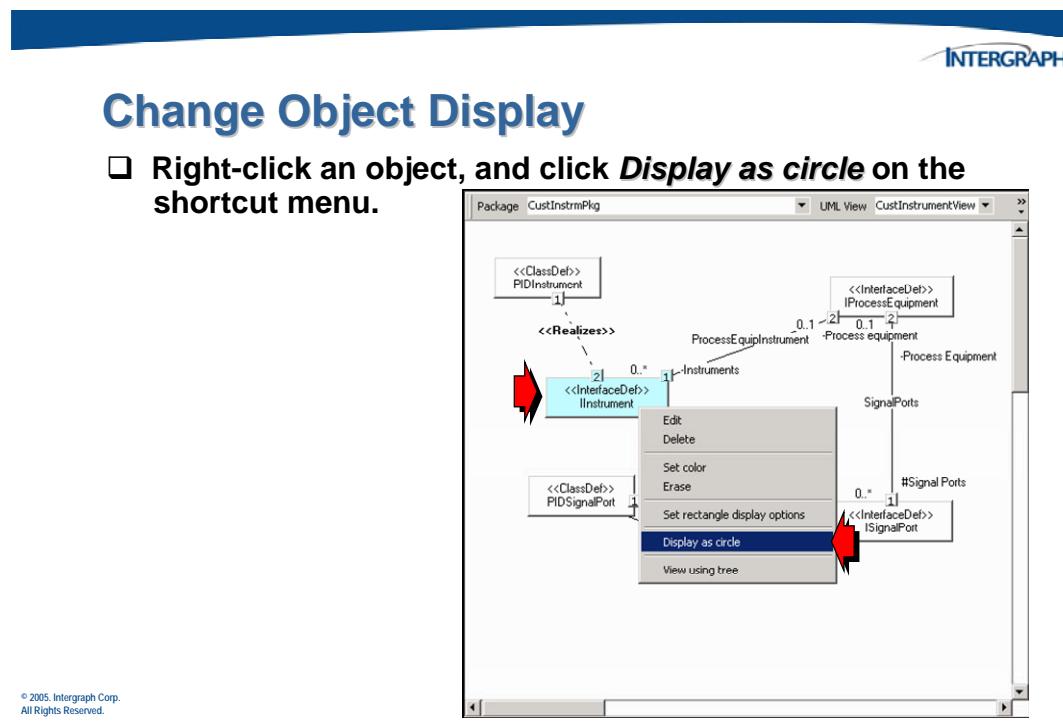
The selected object is added to the diagram along with its Realizes relationship.



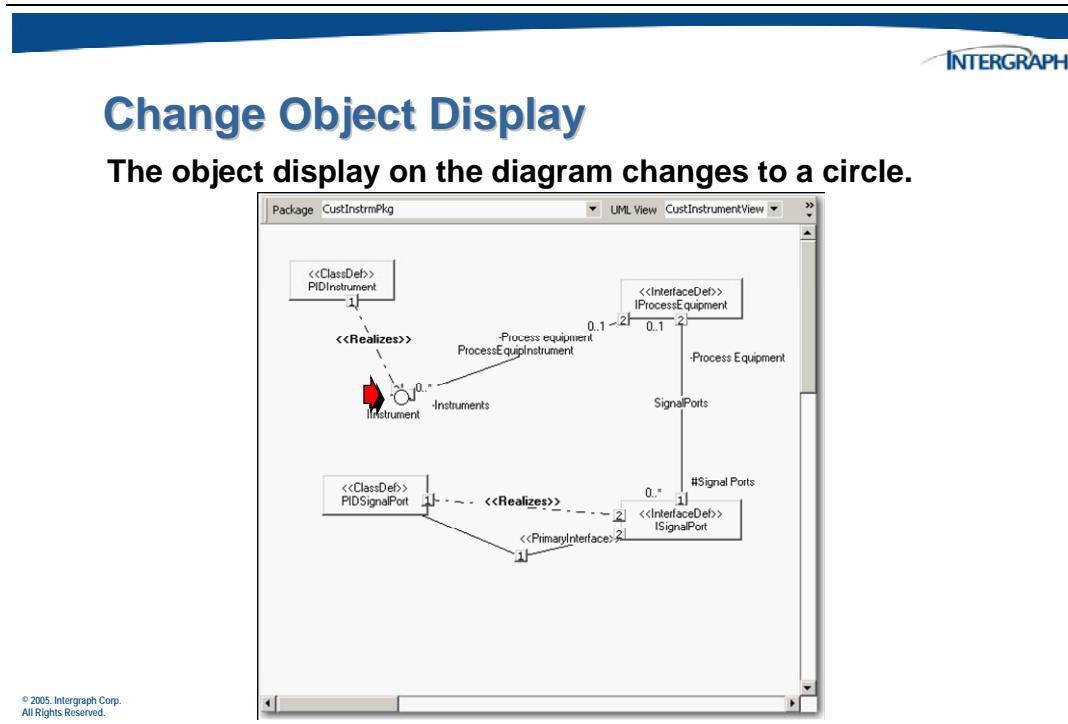
2.4 Change Object Display

By default when you drag and drop an object into a UML view, the dropped object will display as a rectangle. If you have a UML diagram with quite a few objects, the display can become cluttered. An option available to you is to change the display of the object from the default rectangle to a circle.

To do this, right-click an object to display the shortcut menu.



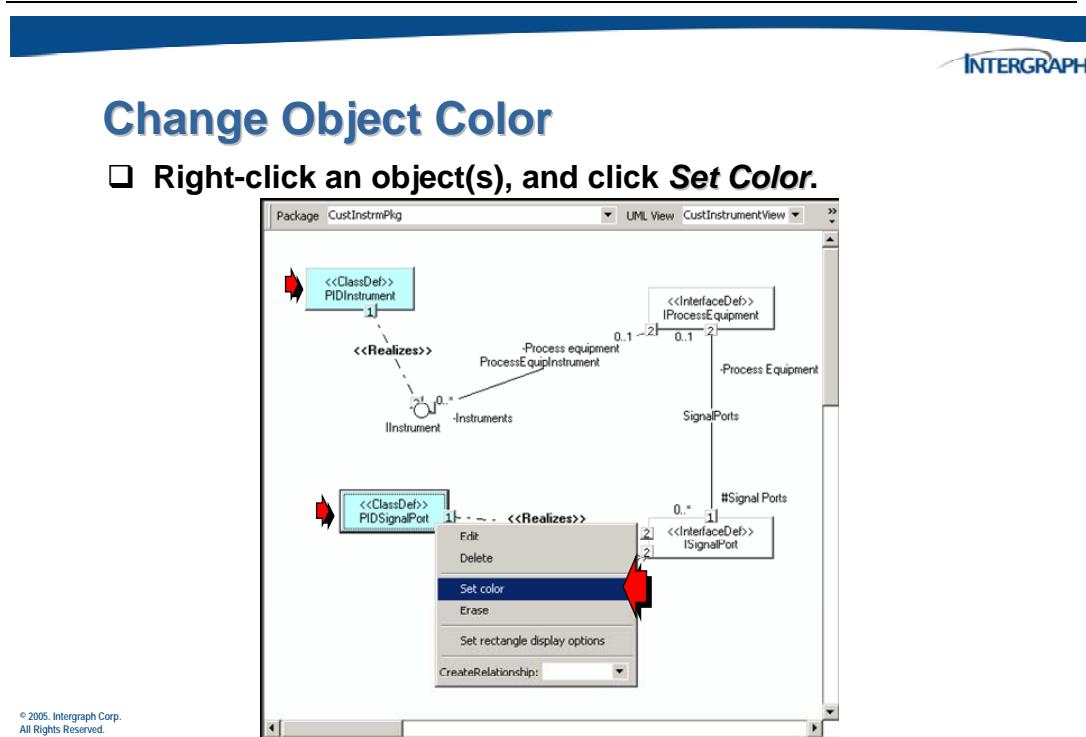
This will change the selected object display from a rectangle to a circle.



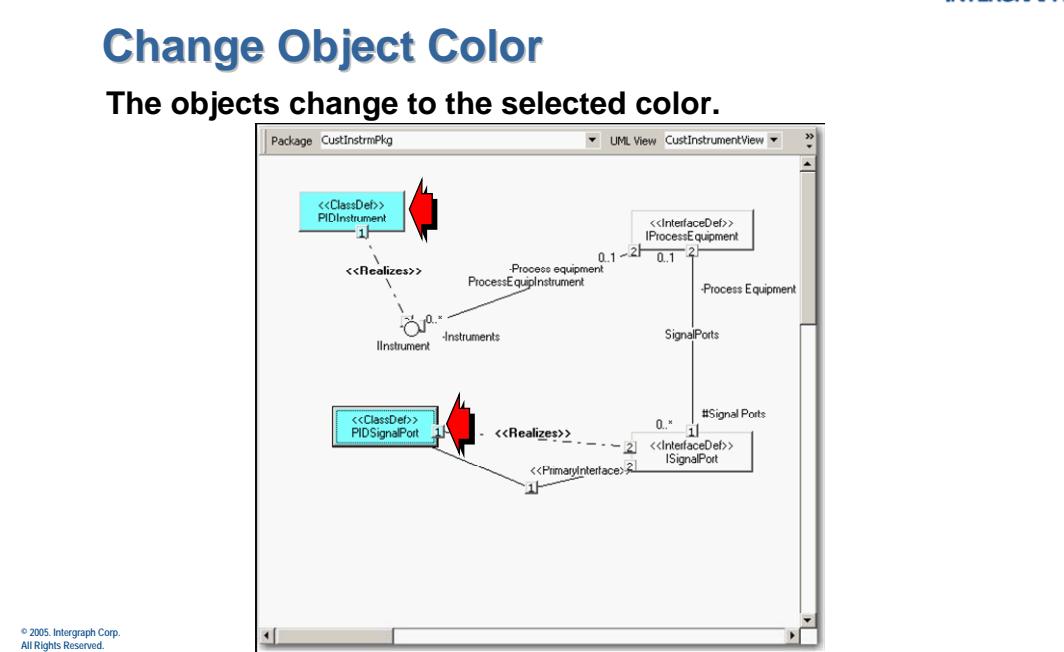
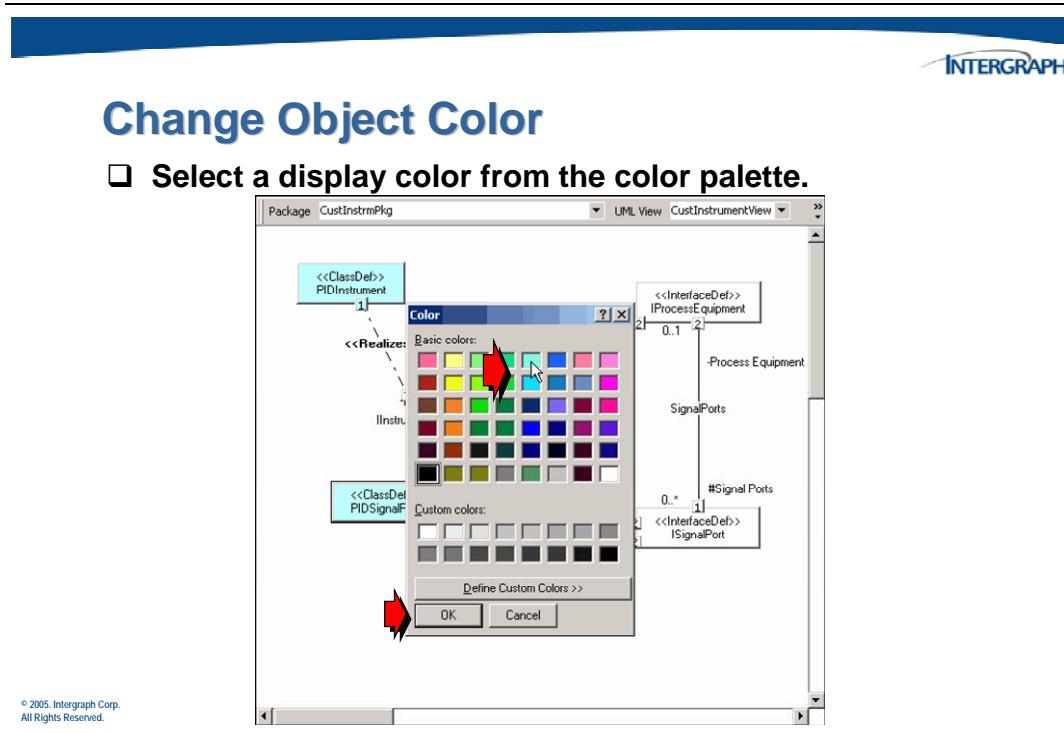
2.5 Change Object Color

To make certain objects easier to see on a UML diagram, you can change the color of the rectangle or circle representing the object.

To do this, right-click an object to display the shortcut menu.



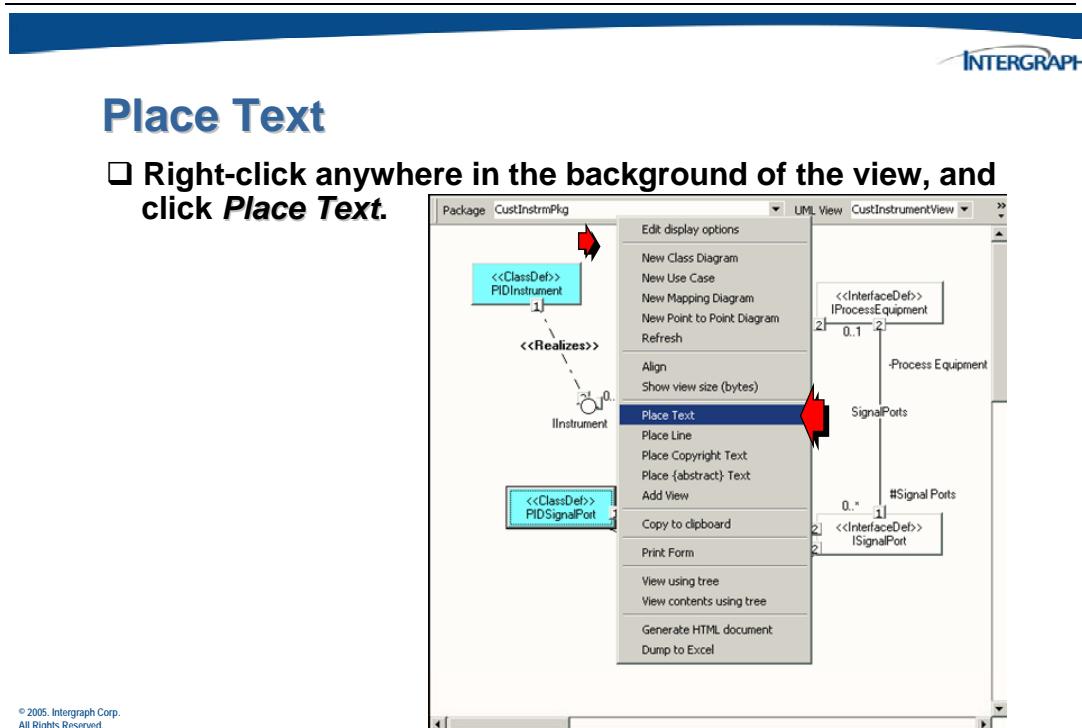
A color palette dialog box appears.



2.6 Placing Text in a View

A UML view diagram can be annotated with text. Use the **Place Text** command from the pop up menu to enter and format any text needed to clarify the diagram.

Right-click anywhere in the view except on an object to display the shortcut menu.

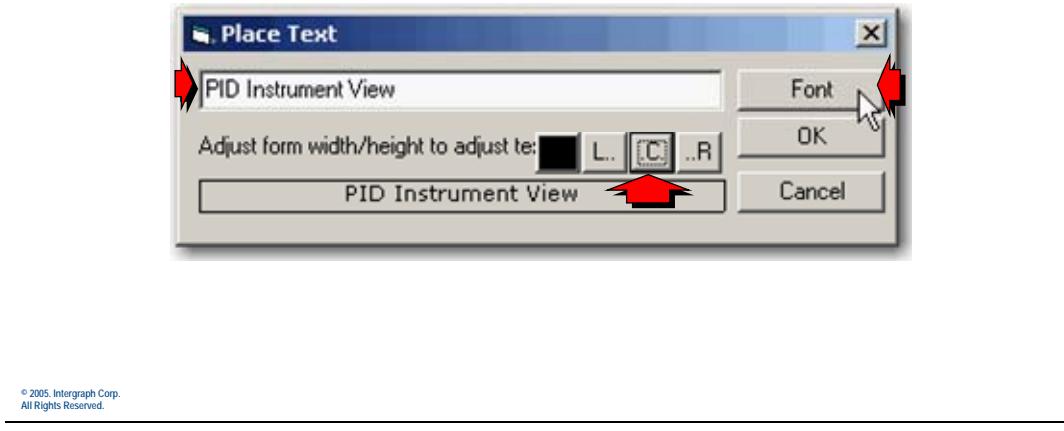


A *Text* dialog box appears.

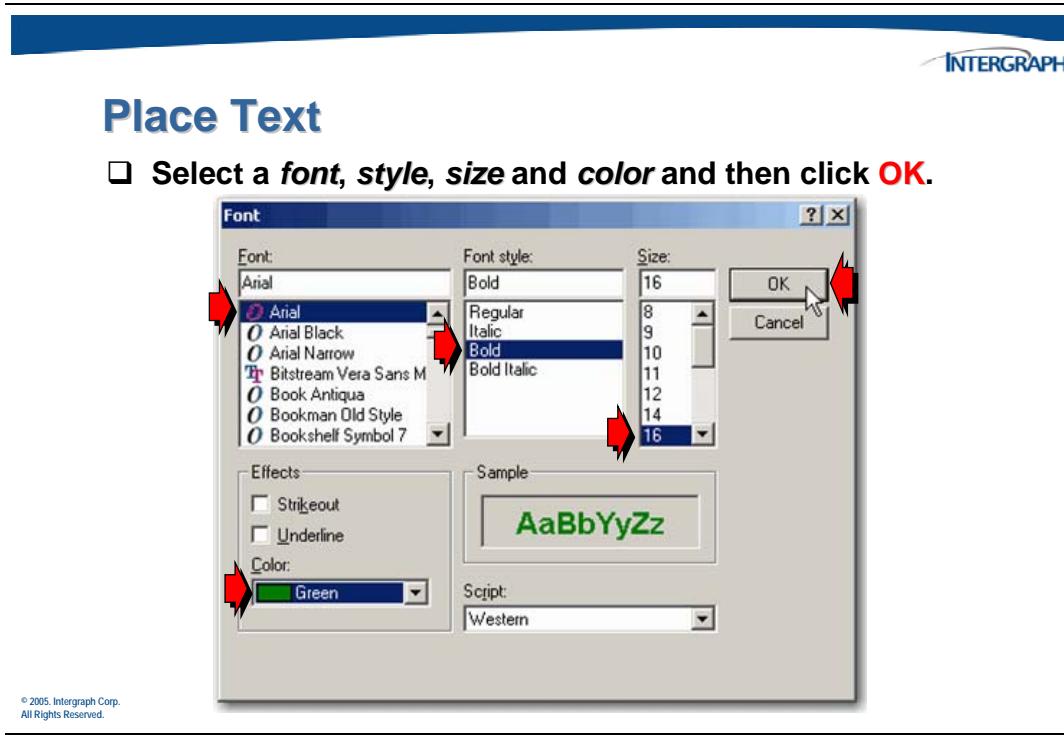


Place Text

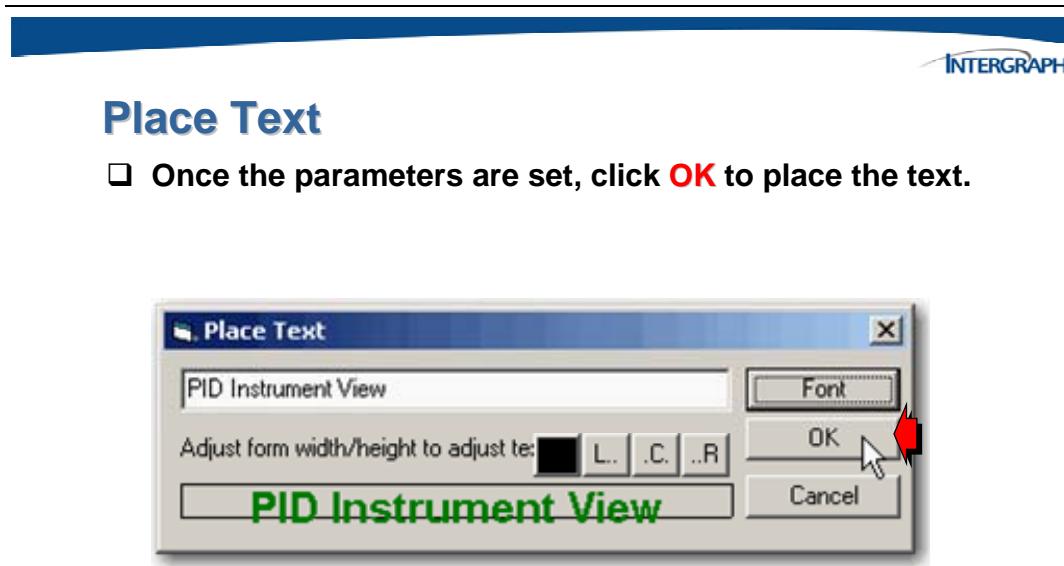
- Enter the text and select the desired parameters, such as text alignment and font.



The *Text* dialog box allows you to set the font color and alignment (left, center or right). Click the **Font** button to set the font parameters.

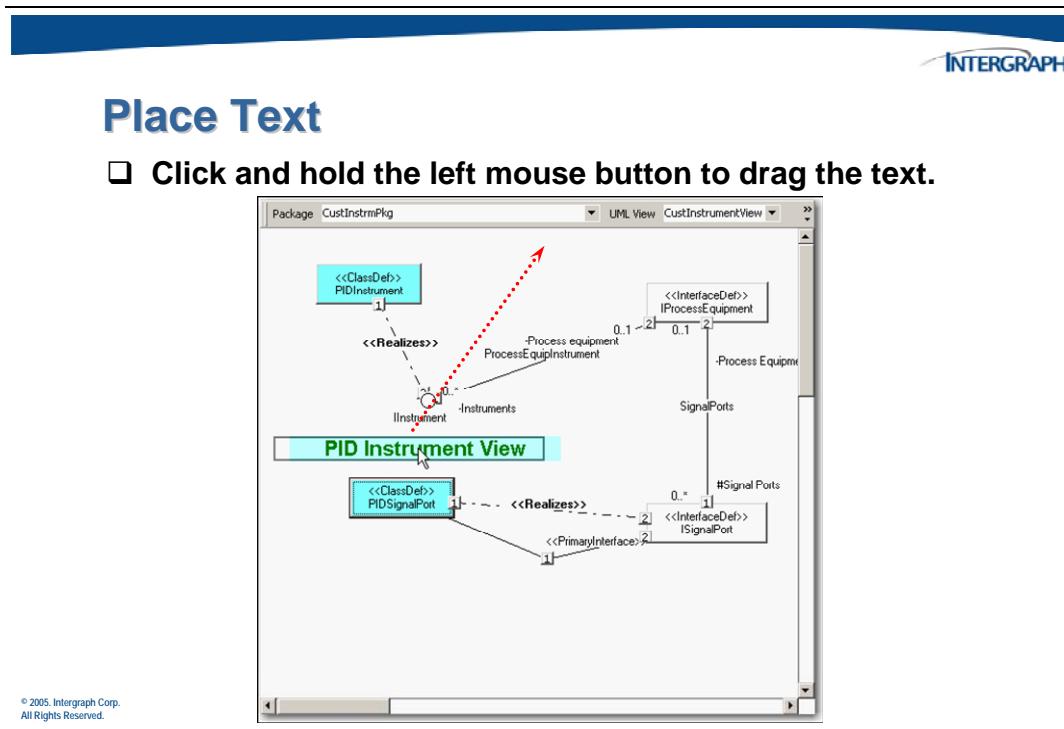


Note the approximation of the font display at the bottom of the Text dialog box.



© 2005, Intergraph Corp.
All Rights Reserved.

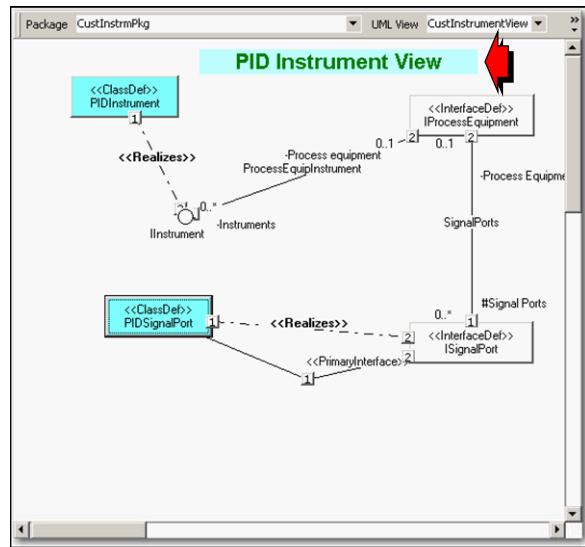
Use the mouse to position the text to the desired location.



INTERGRAPH

Place Text

- Position the text in the desired location and release the mouse button.

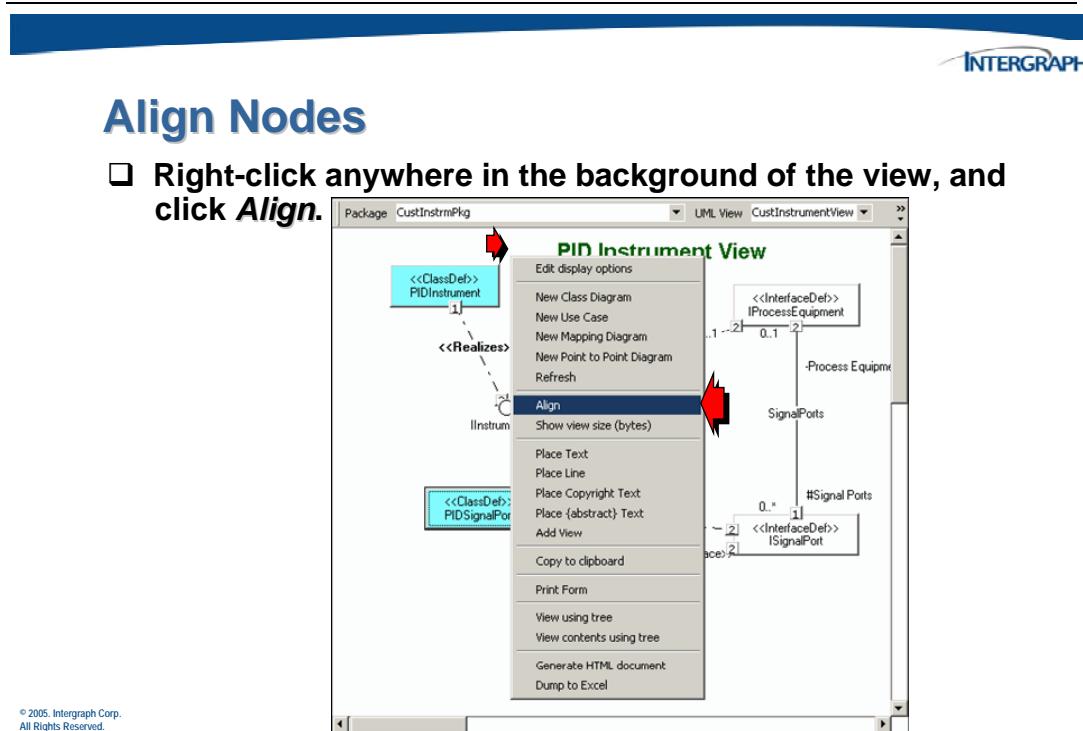


© 2005, Intergraph Corp.
All Rights Reserved.

2.7 Align Nodes

To dress up your custom UML view diagram, you can use the **Align** command on the shortcut menu.

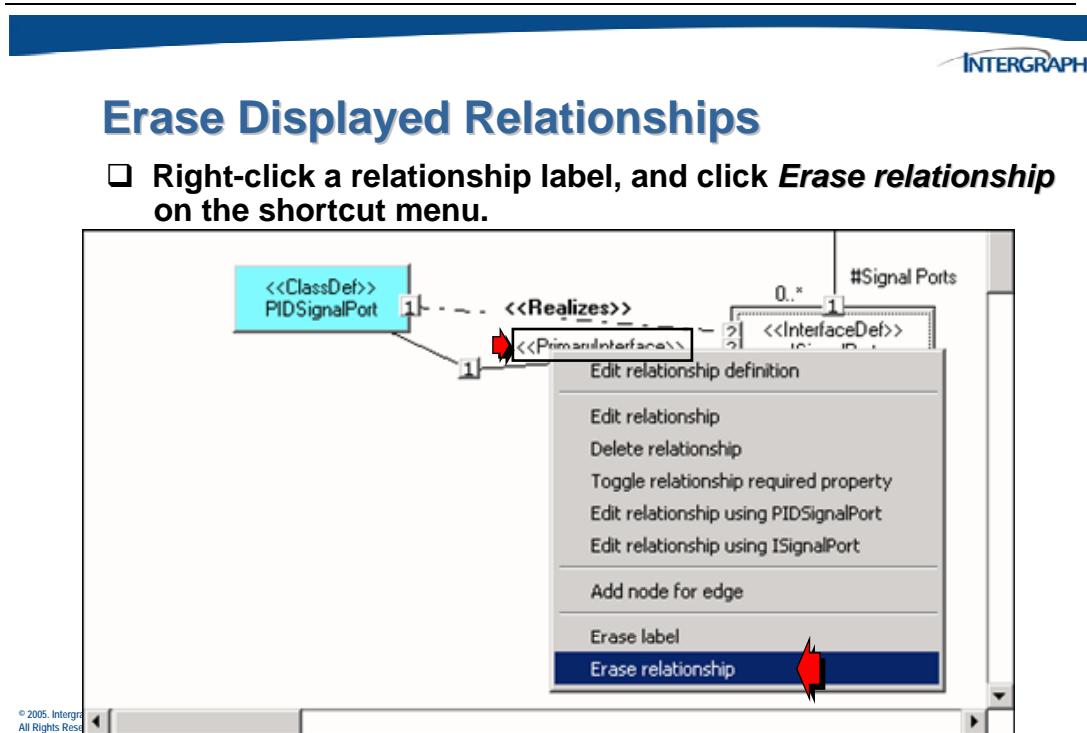
Right-click anywhere in the background of the Drag-Drop UML view.



2.8 Erasing Displayed Relationships

The Erase Relationship command allows you to remove a displayed object from a UML view. The erased object remains in the schema, but it is no longer shown on the displayed view.

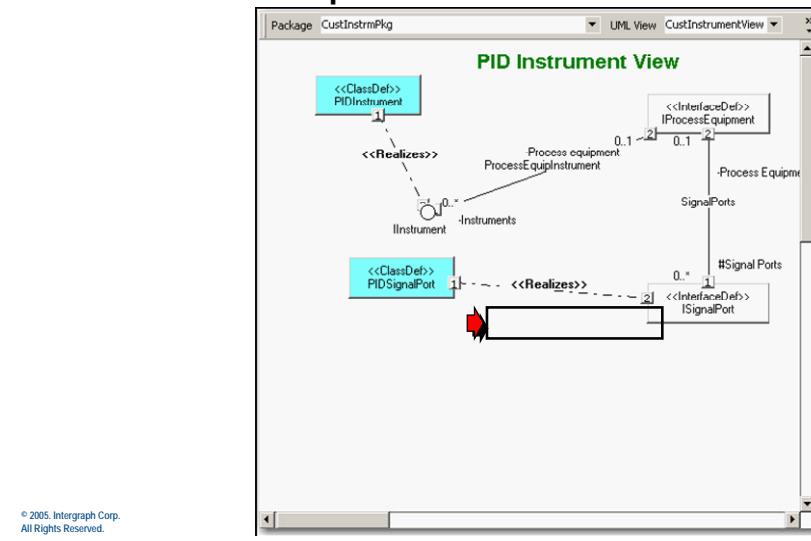
In the Drag-Drop UML view, right-click the object that you want to erase.





Erase Displayed Relationships

The relationship and label are erased from the UML display.

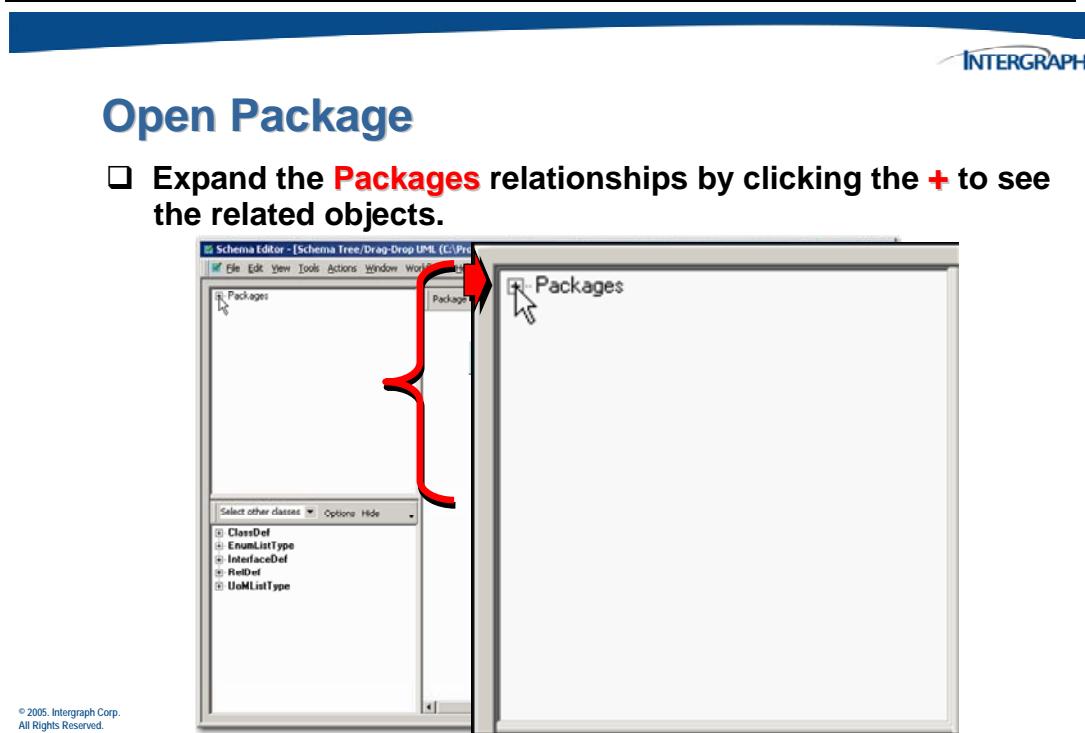


© 2005, Intergraph Corp.
All Rights Reserved.

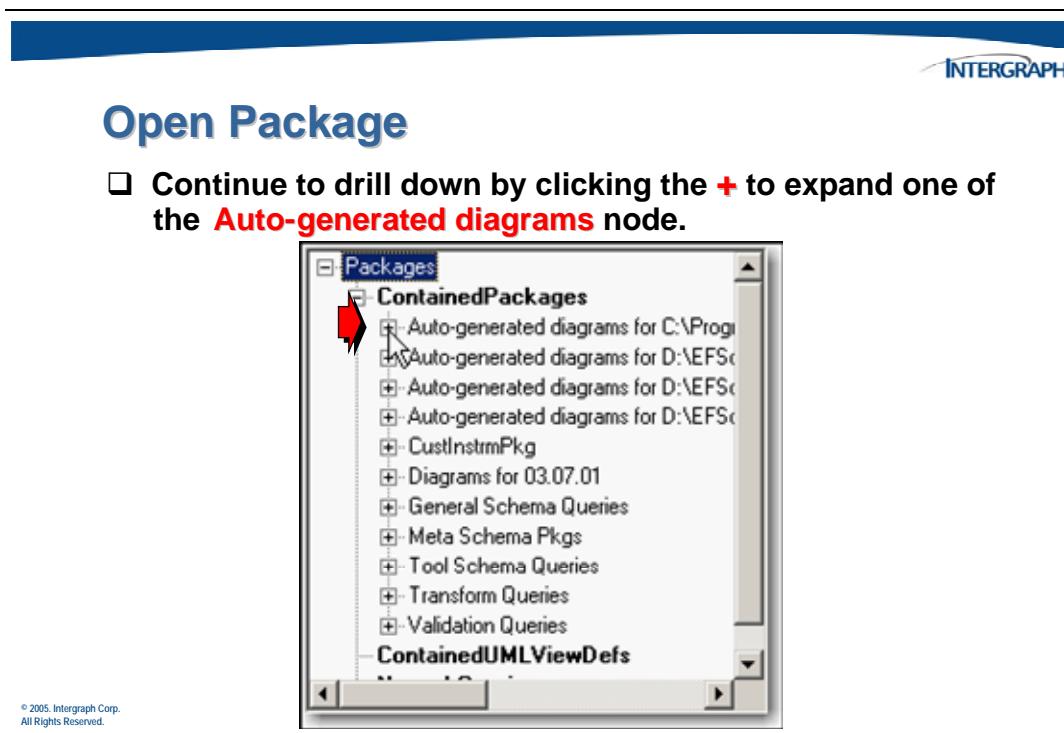
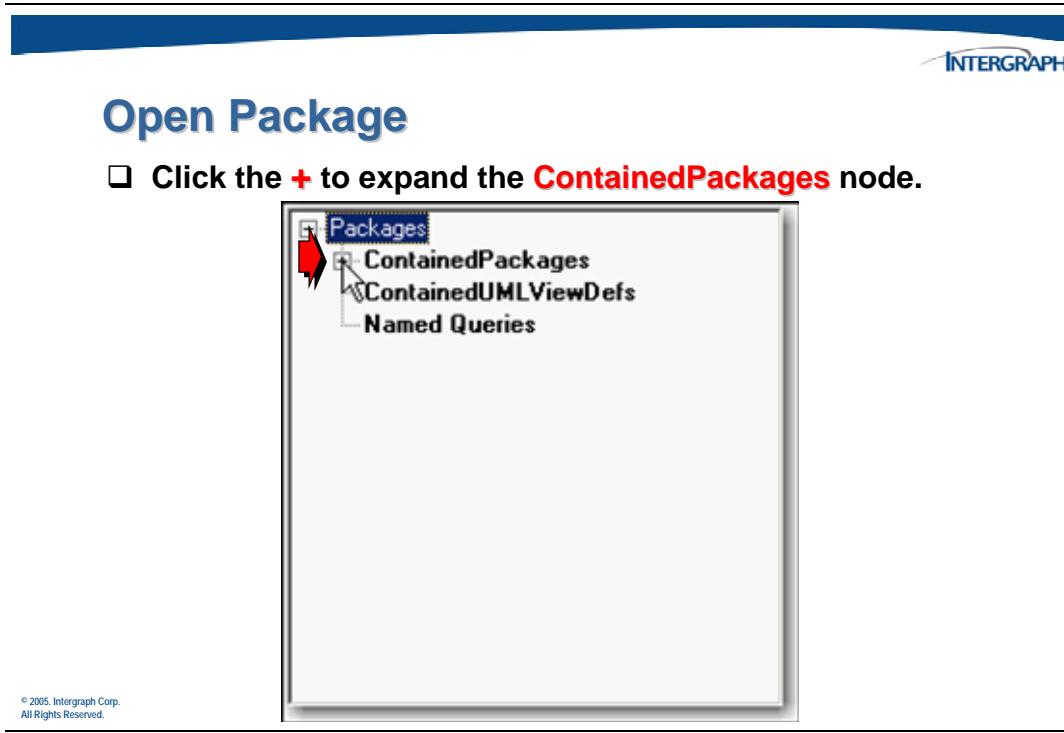
2.9 Open a Package

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.

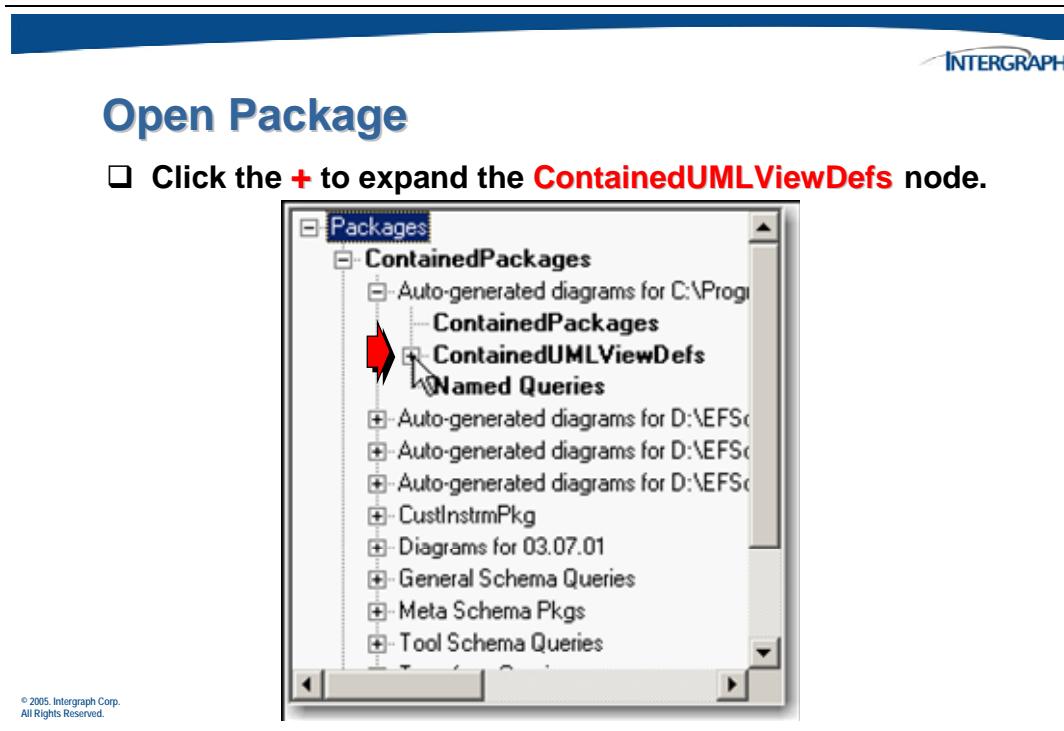
You can view existing UML views by expanding the **Packages** tree and then selecting the UML diagram that you want to see in the UML view.



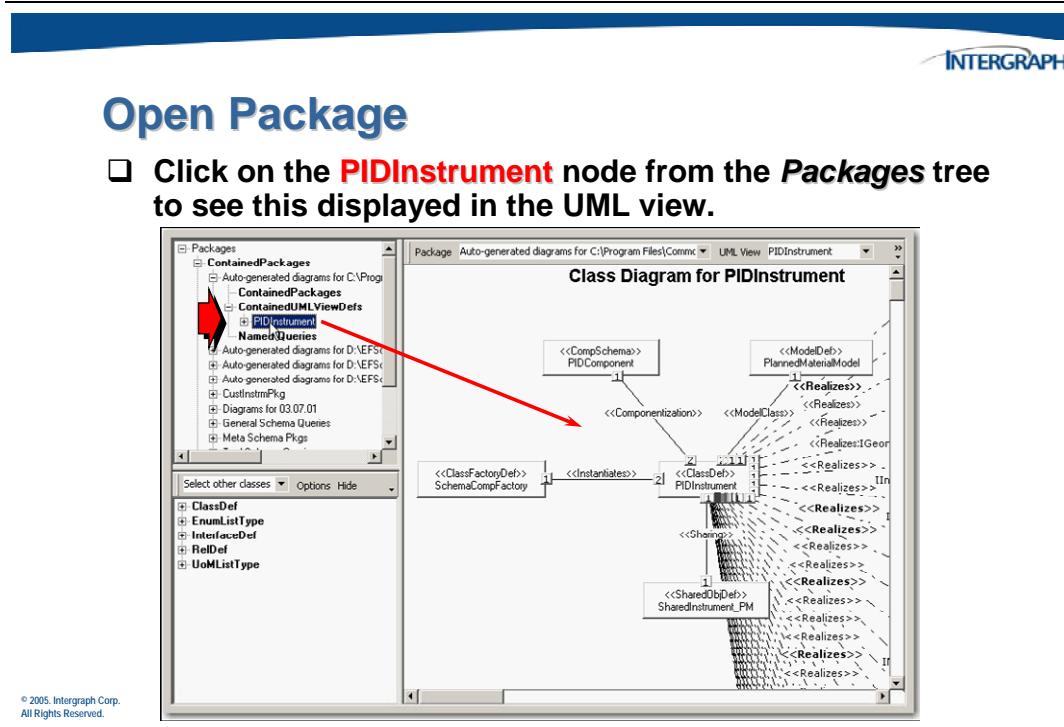
Expand nodes in the tree to see the package objects.



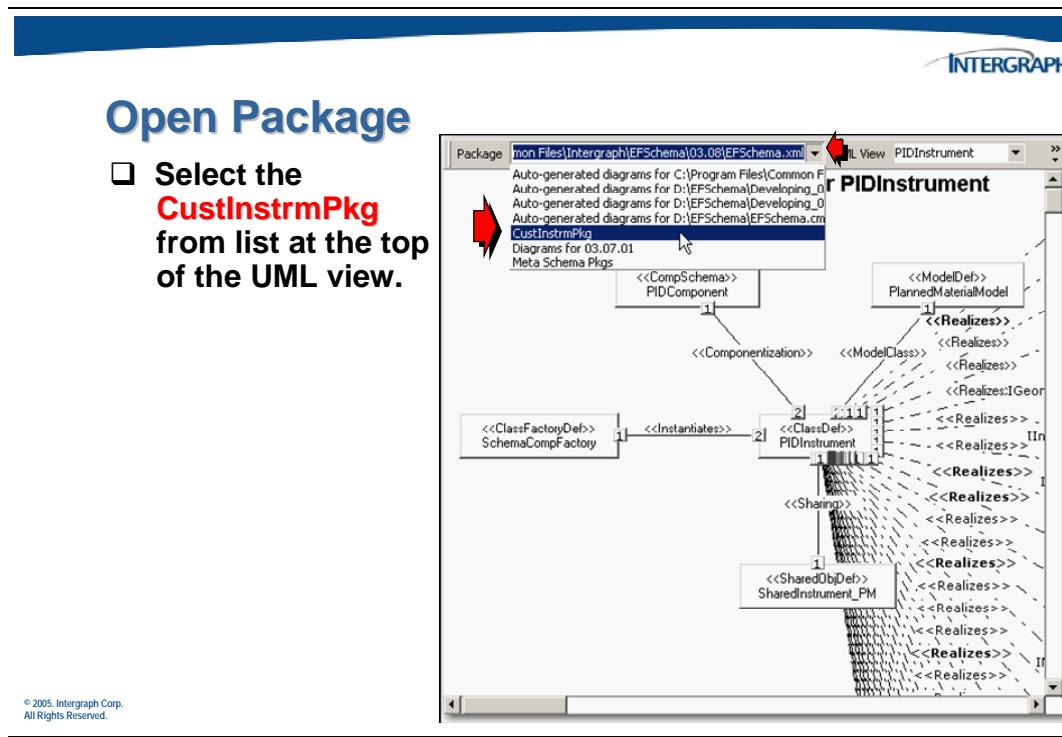
You will be looking for any of the custom UML views that you have defined.



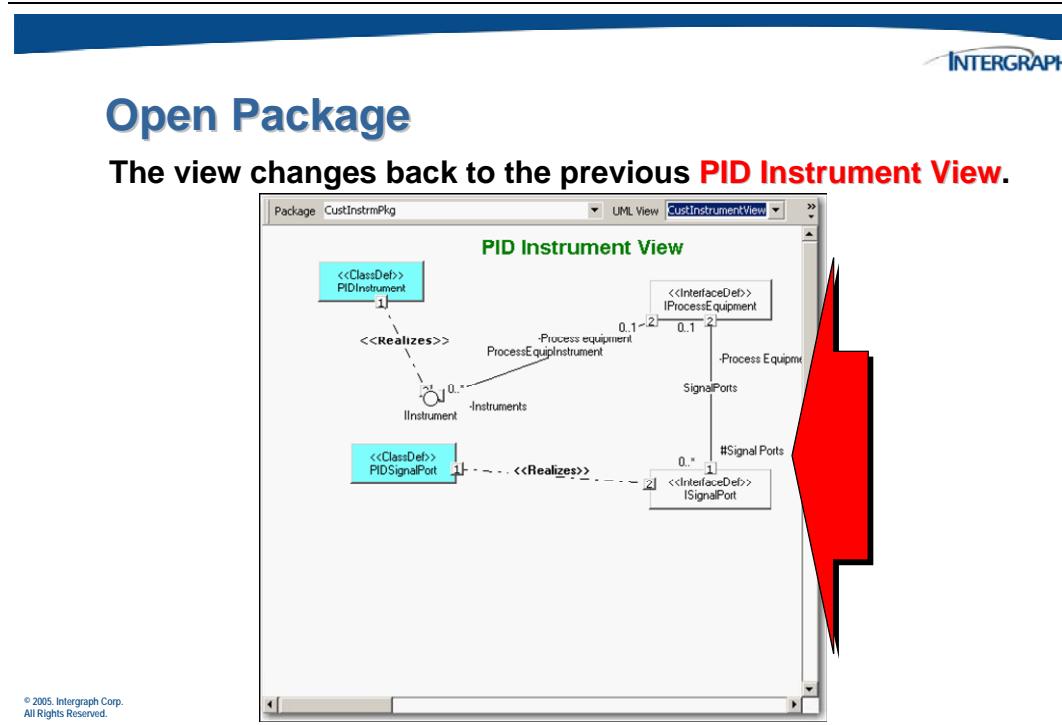
Select a saved view that you wish to display.



The displayed view can be changed to the custom UML view.



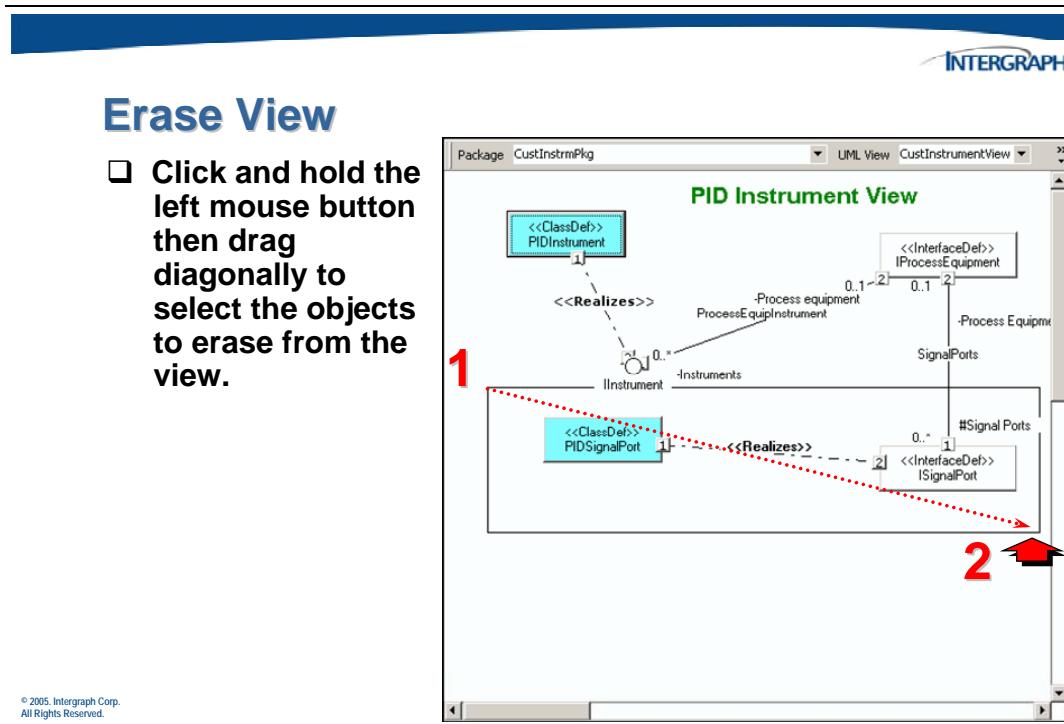
© 2005, Intergraph Corp.
All Rights Reserved.



© 2005, Intergraph Corp.
All Rights Reserved.

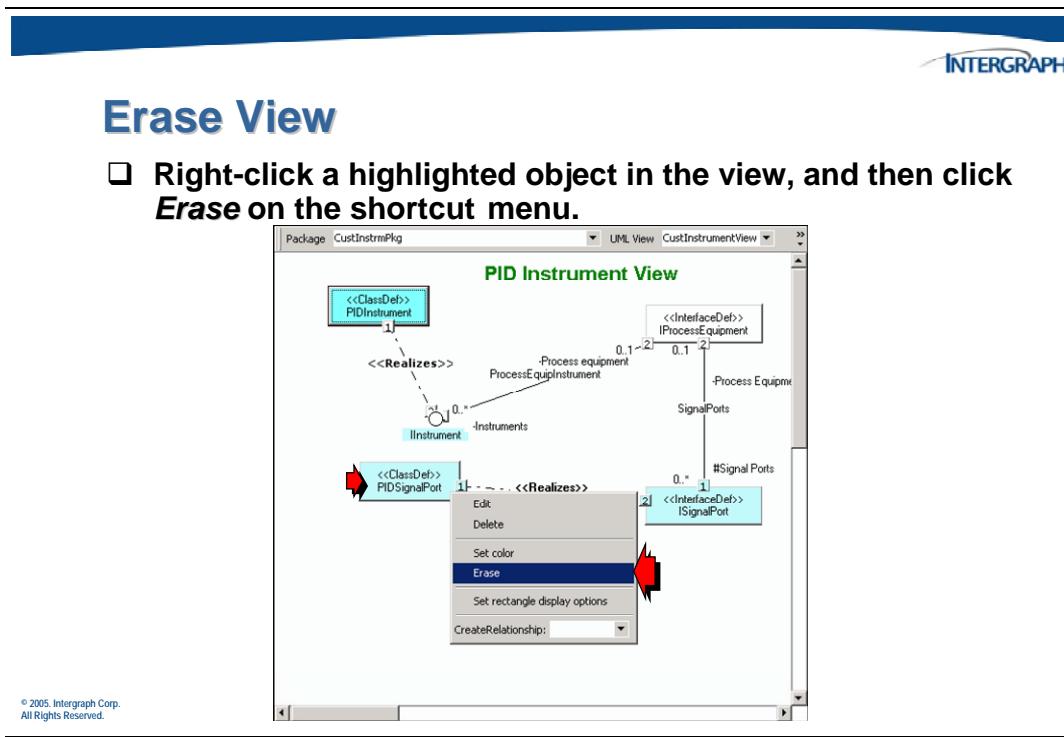
2.10 Erasing the View

The **Erase** command on the shortcut menu will allow you to remove a portion or all of the objects displayed in a view. Before selecting the **Erase** command, select the objects that you want to erase by placing a rectangle around them.

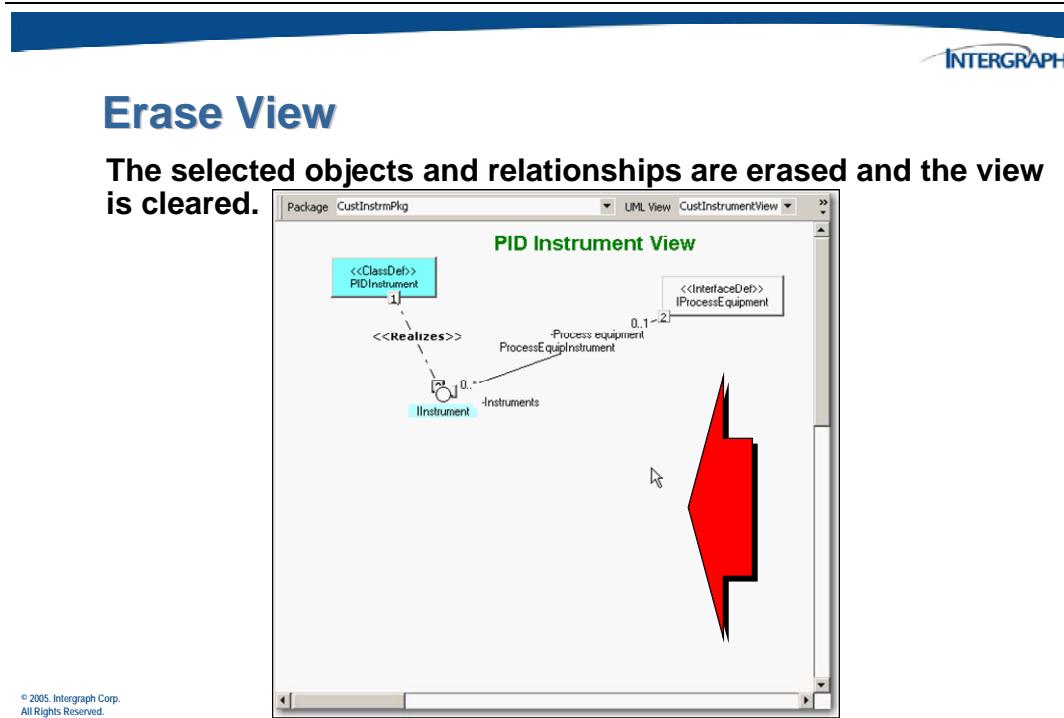


As you drag the mouse diagonally a dynamic rectangle will be displayed.

All of the objects completely inside of the dynamic rectangle will be selected.



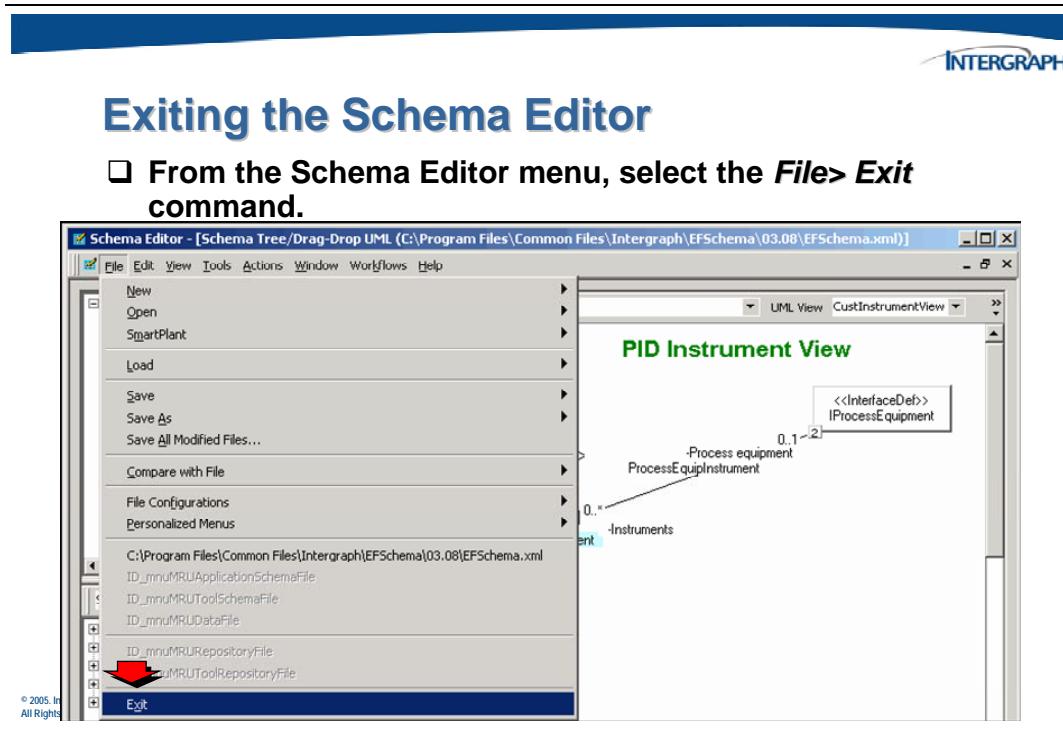
The selected objects will be erased but will remain in the schema.



Note: You can also clear the view by right-clicking in the background of the view and clicking *New UML view definition* on the shortcut menu.

2.11 Exiting the Schema Editor

Once the view windows have been closed, use the **File** command on the main menu to **Exit** from the Schema Editor.

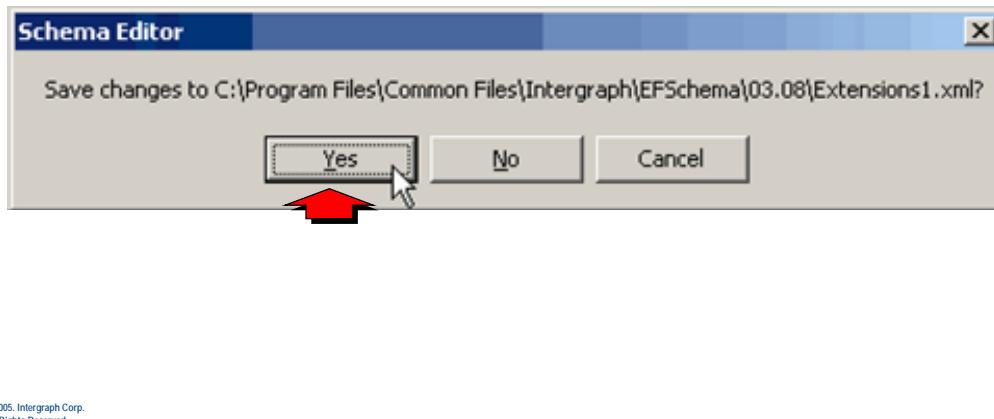


You will be prompted whether or not to save any changes that has been made to the extension schema.



Exiting the Schema Editor

- Click **Yes** to save the custom package/view in the schema file.



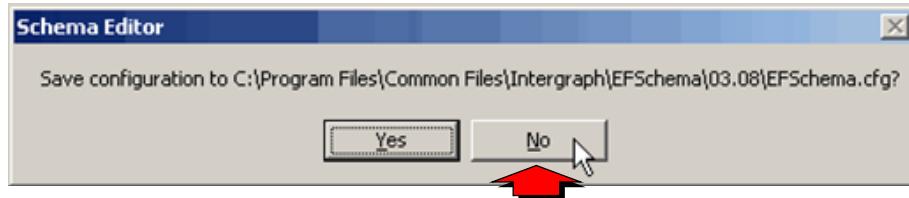
© 2005, Intergraph Corp.
All Rights Reserved.

A *Schema Editor* window may display to allow you to save the session configuration.



Exiting the Schema Editor

- Click **No** to save the session configuration information to a file.



© 2005, Intergraph Corp.
All Rights Reserved.

2.12 Activity 1 – Using the Schema Editor

The goal of this activity is to familiarize you with using the Schema Editor to view schema components. You will start the editor and make sure the PIDComponent schema is open. You will then use different types of views to familiarize yourself with the schema components.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
3. Open the PID component schema xml file.
 - Click **File > Open > Schema File** on the *Schema Editor* menu
 - Browse to **C:\Program Files\Common Files\Intergraph\EFSchema\03.08** in the *Open Schema File* dialog box.
 - Select **PIDComponent.xml** file and click **Open**

Viewing Schema Objects

4. View the open schema file components using the **Tree** view type.
 - In the *Workflows* dialog box, click the **View** button beside the **Schema File** button.
 - When the *View Schema* dialog box appears, select the **View** tab, choose the **Tree** option and click **OK**.
 - Expand the **ClassDef** object.
 - Expand the **PIDInstrument** object.
 - List some of the edges (relationships) that are displayed:

- Close the view window (click **X**).
5. View the open schema file components using the **Tree/Properties** view type.
- In the *Workflows* dialog box, click the *View* button beside the *Schema File* button.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/Properties** option and click **OK**.
 - Expand the **ClassDef** object.
 - Click the **PIDInstrument** object.
 - What happens to the right pane in the window?
-
- What three properties identify this object (based on the display in the right pane)?
-
-
-
- Close the view window.
6. View the open schema file components using the **Tree/Table (Horizontal)** view type.
- In the *Workflows* dialog box, click the *View* button beside the *Schema File* button.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/Table (Horizontal)** option and click **OK**.
 - Expand the **ClassDef** object.
 - Click the **PIDInstrument** object.
 - What happens to the right pane in the window?
-
- Close the view window.
7. View the open schema file components using the **Tree/Drag-Drop UML** view type.
- In the *Workflows* dialog box, click the *View* button beside the *Schema File* button.

- When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/Drag-Drop UML** option and click **OK**.
 - Expand the **ClassDef** object.
 - Click the **PIDInstrument** object.
 - Drag and drop the **PIDInstrument** object into the right pane.
 - When the *Schema Editor* dialog box appears, click the **Yes** button to create a default UML view.
 - Review the objects and relationships in the newly created UML view.
 - What does the acronym UML stand for?
-
- Close the view window.
8. View the open schema file components using the **Tree/UML** view type.
- In the *Workflows* dialog box, click the *View* button beside the *Schema File* button.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/UML** option and click **OK**.
 - Expand the **ClassDef** object.
 - Click the **PIDInstrument** object.
 - Review the objects and relationships in the default diagram.
 - Locate the **IIInstrumentOcc** interface and click it.
 - What happens to the display?
-
- Use the **Back** button to return to the previous diagram.
 - Close the view window.
9. View the open schema file components using the **Editor** view type.
- In the *Workflows* dialog box, click the *View* button beside the *Schema File* button.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Editor** option and click **OK**.
 - Click the **ClassDef** object in the middle of the left pane.
 - Drag and drop the **ClassDef** object into the right pane.

- Click **Cancel** in the *New Class Definition* dialog box.
- Close the view window.

Finding Schema Objects

10. Use the **Find** functionality to locate and view schema file components using the **Tree** view type.
 - In the *Workflows* dialog box, click the **View** button beside the **Schema File** button.
 - When the *View Schema* dialog box appears, select the **View** tab, choose the **Tree** option.
 - Select the **Find** tab.
 - In the *Only Classes* field, enter **ClassDef**.
 - In the *Name* box, type ***Instrument*** and click **OK**.
 - Expand the **ClassDef** object.
 - Expand the **PIDInlineInstrument** object.
 - Close the view window.
11. Use optional criteria to locate and view schema file components using the **Tree** view type.
 - In the *Workflows* dialog box, click the **View** button beside the **Schema File** button.
 - When the *View Schema* dialog box appears, select the **View** tab, choose the **Tree** option.
 - Click the **Find** tab.
 - In the *Only Classes* box, type **RelDef**.
 - In the *Name* box, type ***Doc*** and click **OK**.
 - Expand the **RelDef** object.
 - Close the view window.
12. Another way to execute a **Find** is to select it from the *Workflows* dialog box. Use optional criteria to locate and view schema file components.
 - In the *Workflows* dialog box, click the **Find** button.

- When the *Find* dialog box appears, in the *Name* box, type ***Pressure*** and click **OK**.
 - Expand the **PropertyDef** object.
 - Close the view window.
13. Close the schema editor and then re-start it in order to open the master schema configuration. Then locate and view schema file components using the **Tree/Drag-Drop UML** view type in order to build a custom UML view.
- Click the **File Configurations > Open Configuration** button in the lower right corner of the application window.
 - When the *Open Configuration File* dialog box displays, select the name **EFSchema.cfg** and click **Open**.
 - In the *Workflows* dialog box, click the **View** button beside the **Another Schema File** button.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/Drag-Drop UML** option and click **OK**.
 - Expand the **ClassDef** object.
 - Drag and drop the **PIDInstrument** object into the right pane.
 - When the *Schema Editor* dialog box appears, click the **No** button to create a custom UML view.
 - When the *NewCustInstrument UML View Definition* dialog box appears, type the *UML view name* **CustInstrumentView** and click the select (...) button.
 - When the *Possible PackageForUMLViewDef* dialog box appears, click **New**.
 - When the *New Package* dialog box appears, type the package *Name* **CustInstrmPkg** and click **OK**.
 - Click **OK** on the *Possible PackageForUMLViewDef* dialog box.
 - Verify that the *Package* that you created (CustInstrmPkg) displays on the *New UML View Definition* dialog box and click **OK**.
 - What happens to the view pane?
-
-
-
- Expand the **PIDInstrument** object and the **Realized Interface Definitions** object.

- Drag and drop the **IInstrument** object into the right pane.
- What happens in the view pane?

- Expand the **IInstrument** object and the **End1 RelDefs** object.
- Drag and drop the **ProcessEquipInstrument** object into the right pane.
- What happens in the view pane?

- Expand the **ProcessEquipInstrument** object, the **End2** object, the listed interface, and the **End2 RelDefs** object.
- Drag and drop the **SignalPorts** object into the right pane.
- Locate and drag and drop the **PIDSignalPort** object into the right pane.

14. Use the shortcut menu to change some of the diagram characteristics.

- Right-click **IInstrument**.
- Click the **Display as circle** command on the shortcut menu.
- What happens in the view pane?

15. Change the colors of the objects displayed in the diagram.

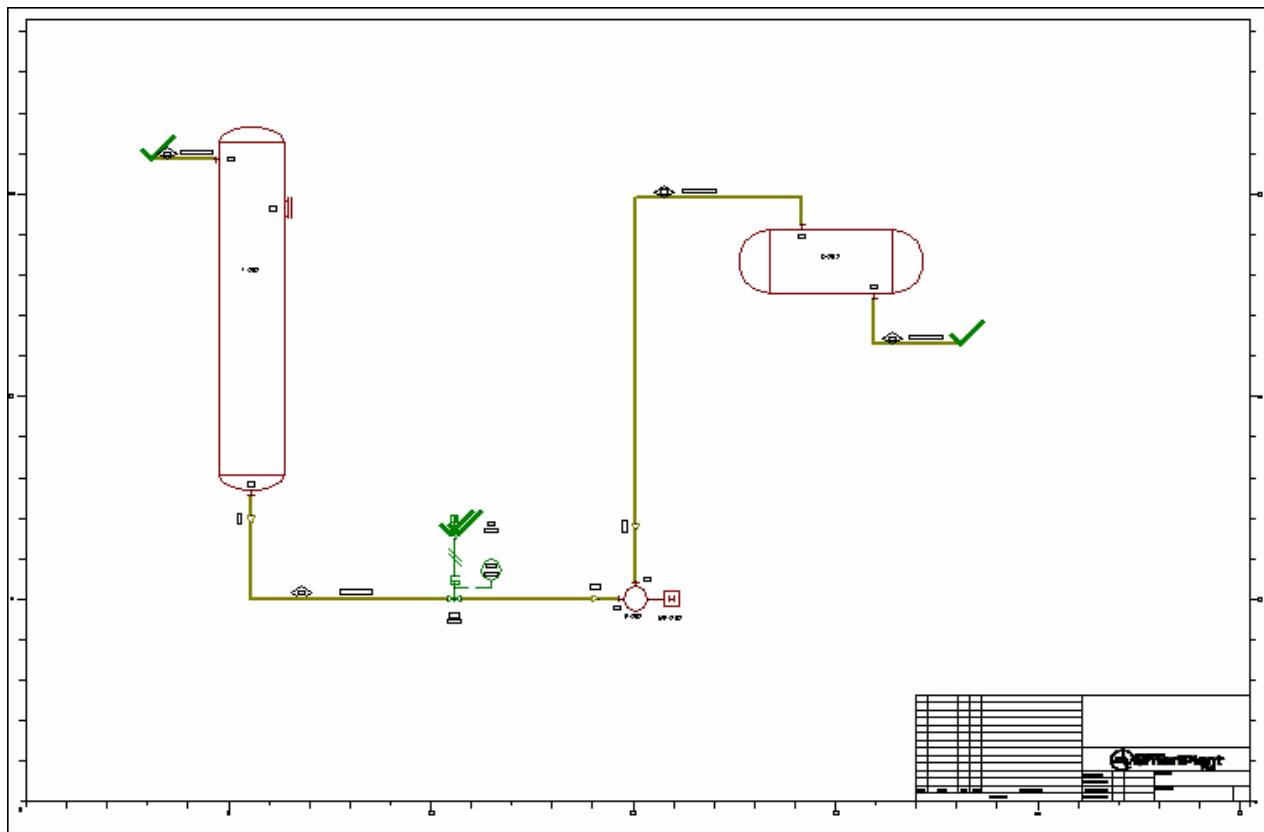
- Right-click one of the objects.
- Click the **Set Color** command on the shortcut menu.
- Pick a color of your choice from the color palette.
- Repeat the process for some of the other objects in the view.

16. Add a text heading to your diagram.

- Right-click the white space in your diagram (not an object).
- Click the **Place Text** command on the shortcut menu.
- Type the text of your choice in the *Text* dialog box.
- Use the **Font** button to set your text style parameters.
- Click **OK** and then drag to position your text.
- Right-click the white space in your diagram and select **New Class Diagram** from the pop-up menu.

(Continued on the next page.)

17. The following is a P&ID Drawing that we want to represent in the schema.



It consists of *Equipment*, *Nozzles*, *Piping Ports*, and *Pipelines* along with other components.

Use the Schema Editor to create the custom UML view shown on the next page. You will want to use the *Tree Drag-Drop UML* view option. Create a custom view called **EquipPipelineView** and put it in a custom package called **EquipPipelinePkg**.

The **ClassDef's** to use in creating this custom view are:

- PIDProcessEquipment
- PIDNozzle
- PIDPipeline

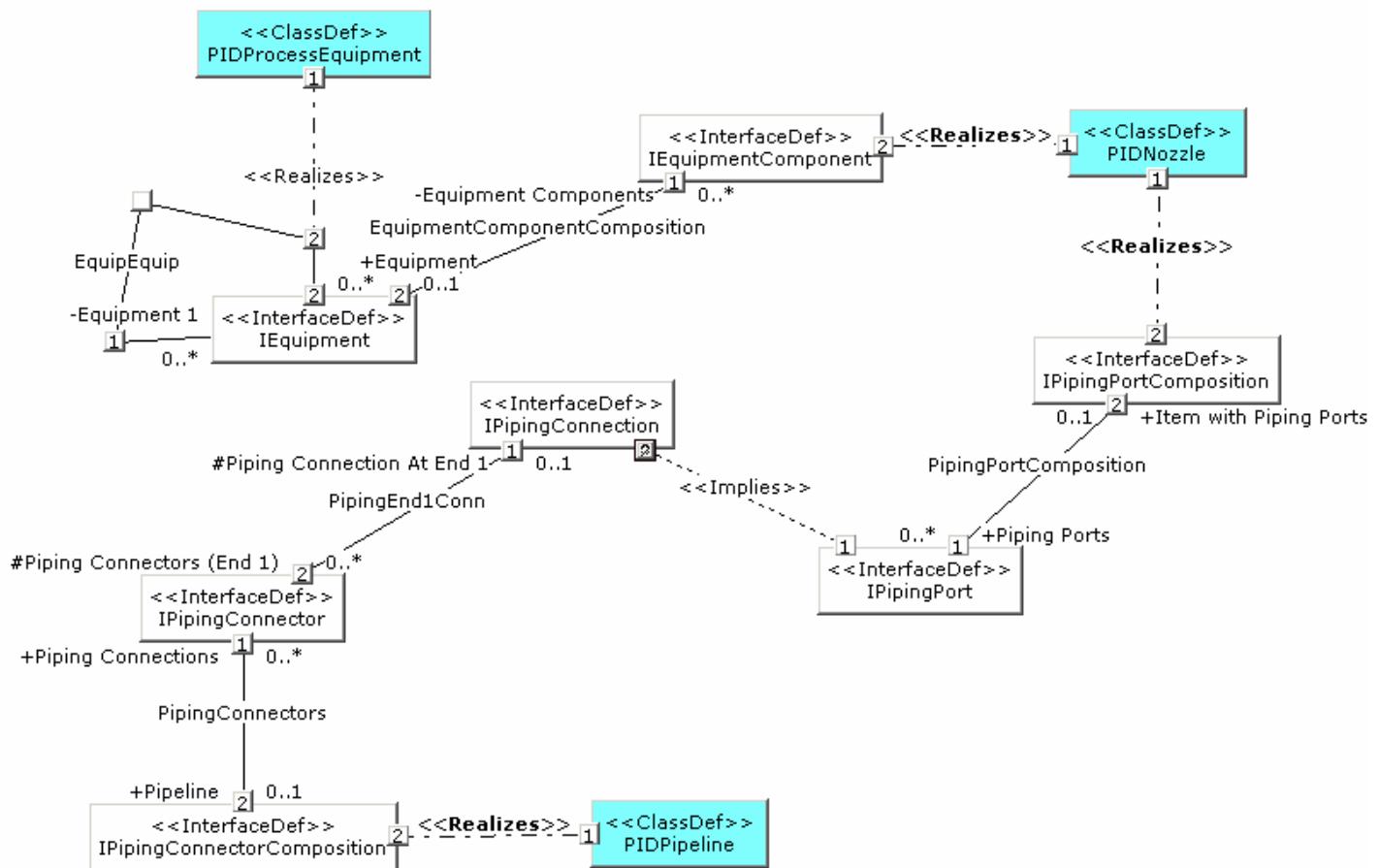
The **Realized Interface** definitions are:

- IEquipment
- IPipingPortComposition
- IPipingConnectorComposition

The **RelDefs For Role** to use are:

- EquipmentComponentComposition
- PipingPortComposition
- PipingEnd1Conn
- PipingConnectors

Note: Some of the relationships that you will see when you drag and drop them into the view have been turned off (Erase Relationship) in the following diagram.



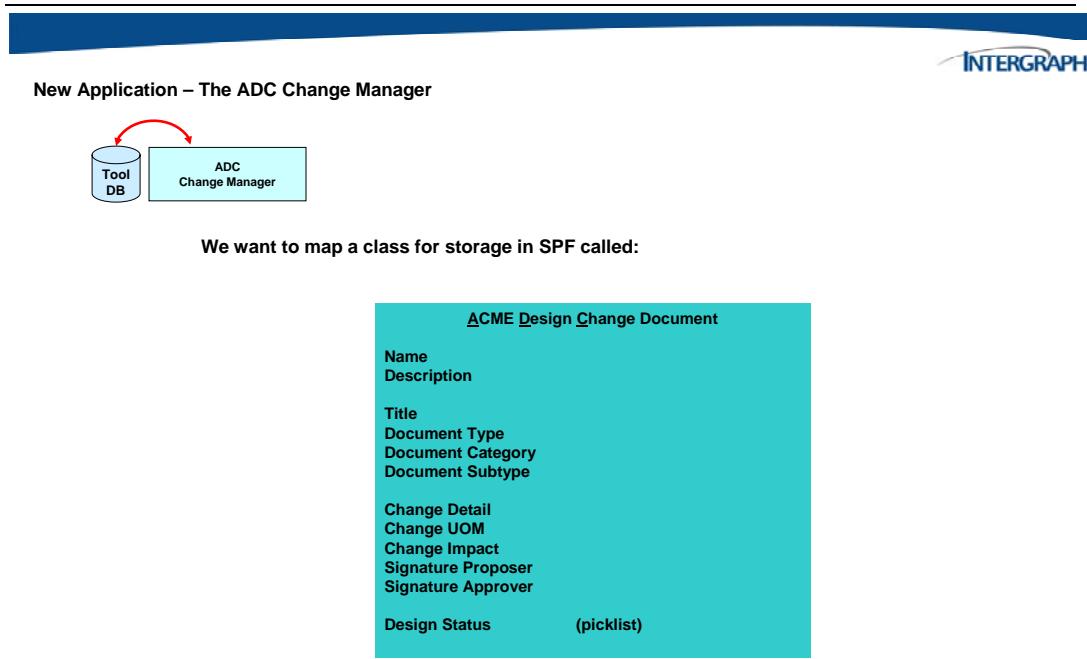
18. Exit the Schema Editor and save your work.
19. Once you have closed the Schema Editor, you may take a short break until the other students have finished this activity.

Summary:

In this activity, you accessed the Schema Editor and familiarized yourself with viewing and finding schema components.

2.13 Creating a Project Schema

As part of the **Schema Modeling and Mapping** course, you will create the necessary class, interfaces, properties and relationships in the hands on sessions in chapters 3, 4 and 5. Below is an example of a **Design Change Document** that will be modeled and eventually implemented.

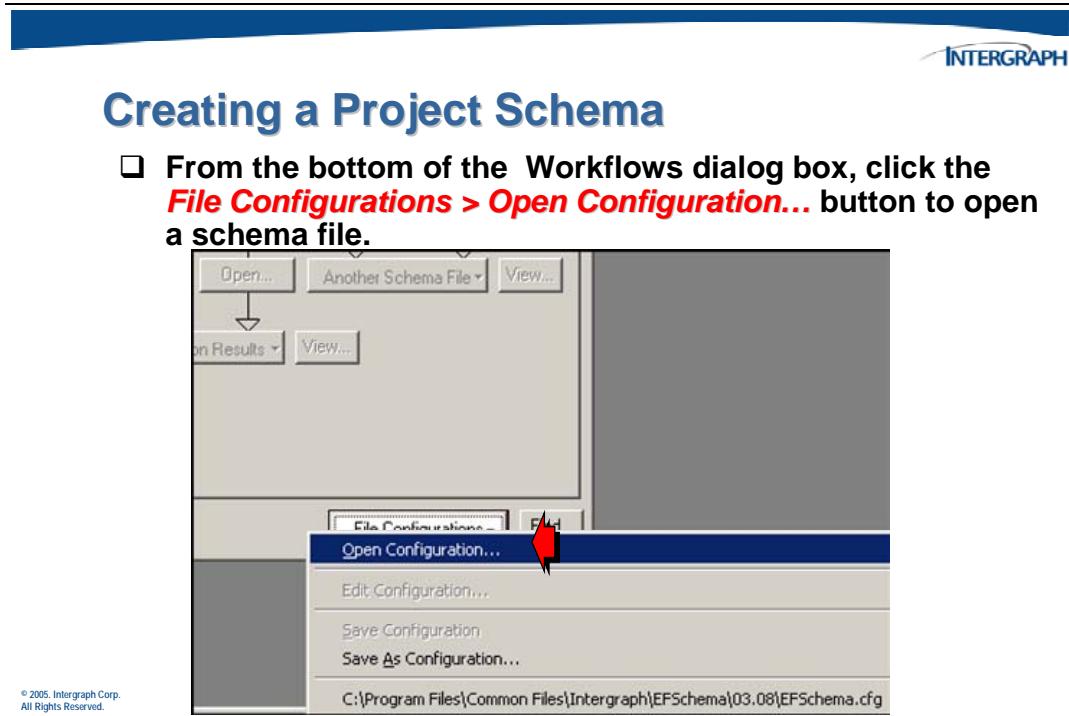


© 2005, Intergraph Corp.
All Rights Reserved.

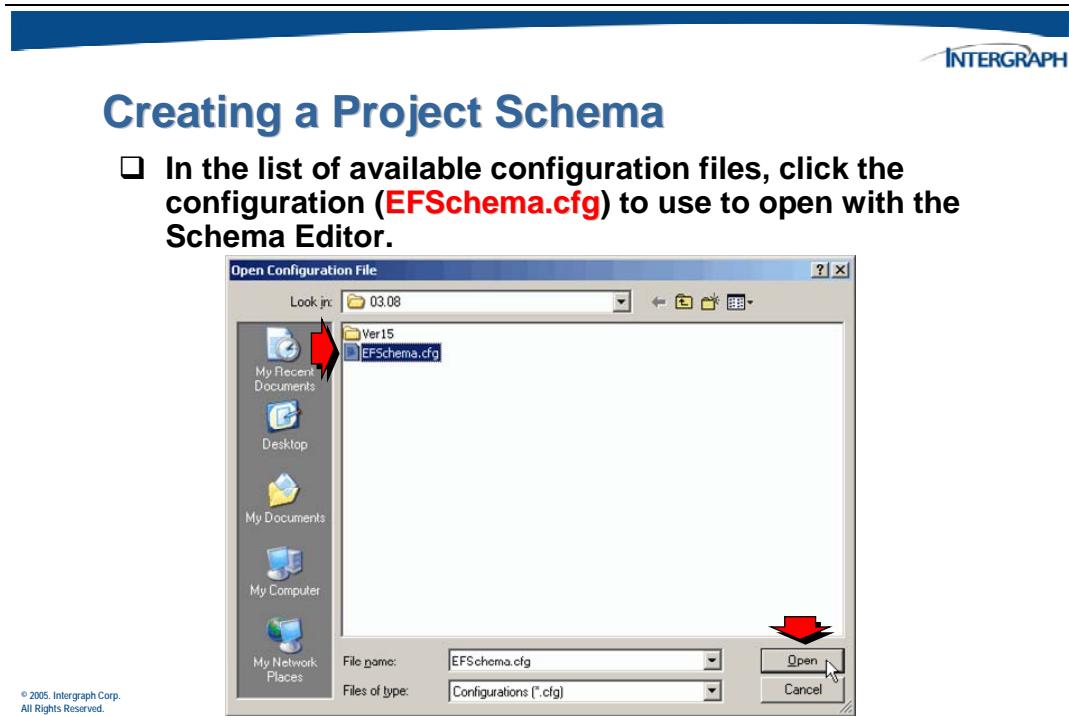
When extending the delivered schema structure, it is recommended that any site specific extensions be made in a “*Project*” schema.

This is done to make it easier to carry site specific extensions forward when you upgrade your SmartPlant Enterprise system. In future releases, just change the configuration of the project schema file with your custom model extensions to use the new version of the EFSchema.xml file.

Begin by setting up a configuration in the Schema Editor to create extensions to the delivered model. Startup the Schema Editor and load the EFSchema.cfg file from the **C:\Program Files\Common\Intergraph\EFSchema\03.08** folder.



The *Open Configuration File* dialog will display.

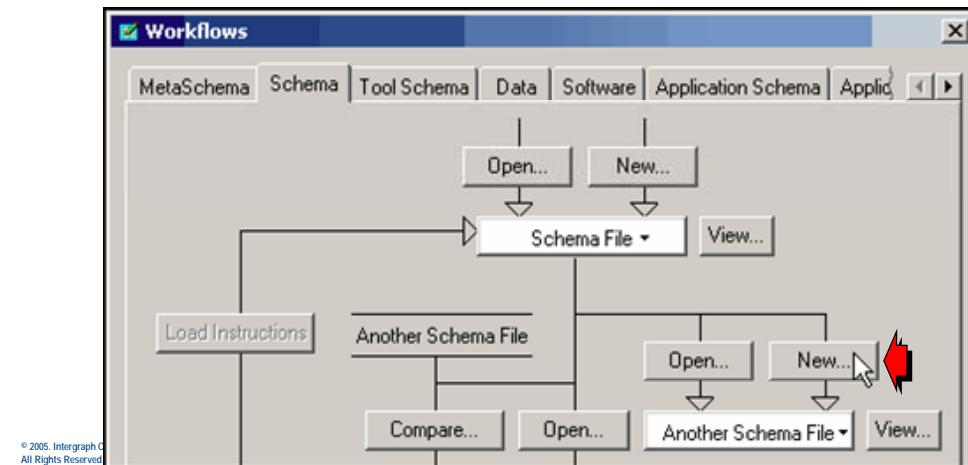


To create a new project schema and to insure the EFSchema file is secure during editing, perform the following steps. Create the project schema using the *Another Schema File New* button.



Creating a Project Schema

- From the Workflows dialog box, create a new project schema by selecting the New button above *Another Schema File*.

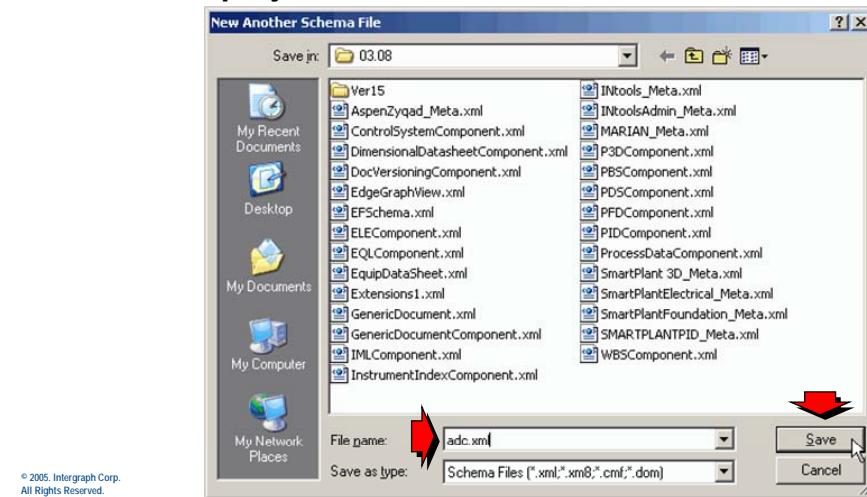


Enter a name for the new project schema file, such as **adc.xml**.

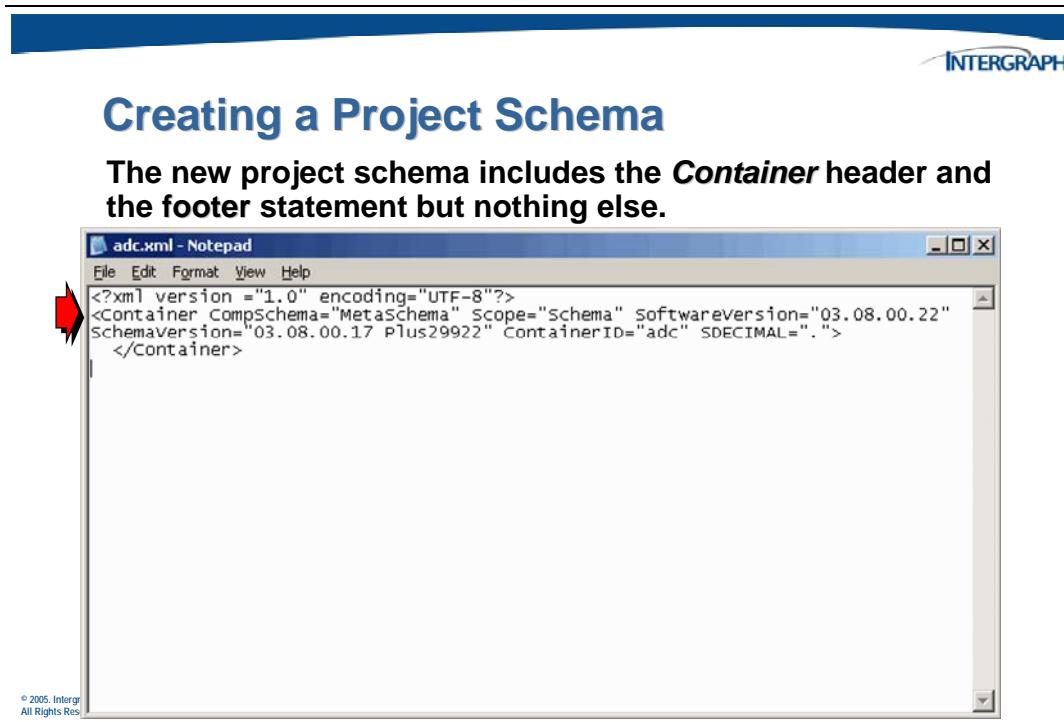


Creating a Project Schema

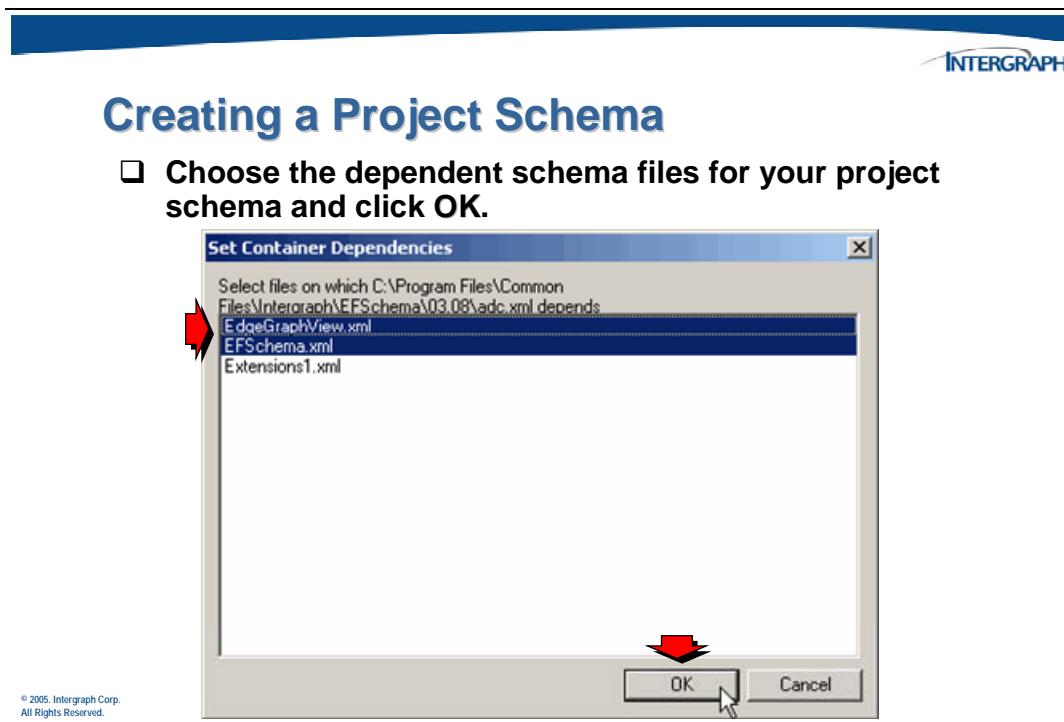
- In the *New Another Schema File* dialog, enter the name for the project schema file and select **Save**.



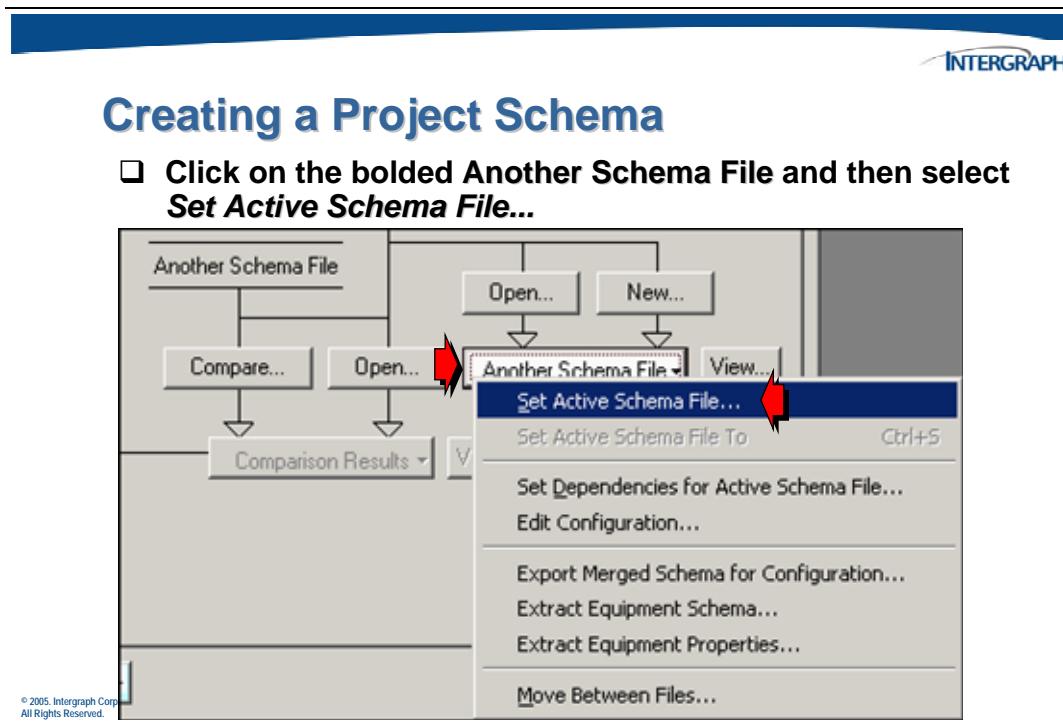
This will create an “empty container” xml that will store all site specific modeling.



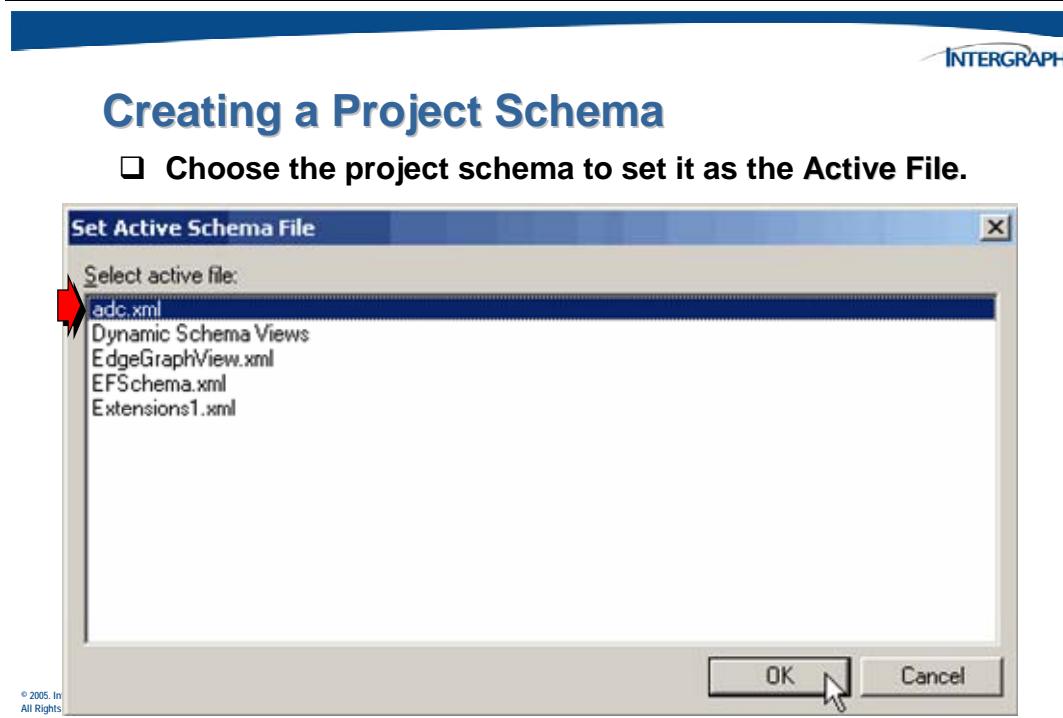
This will display a *Set Container Dependencies* dialog with the **EFSchema.xml** file listed. Select the *master* schema file and the **adc.xml** will be dependent on the EFSchema.xml file.



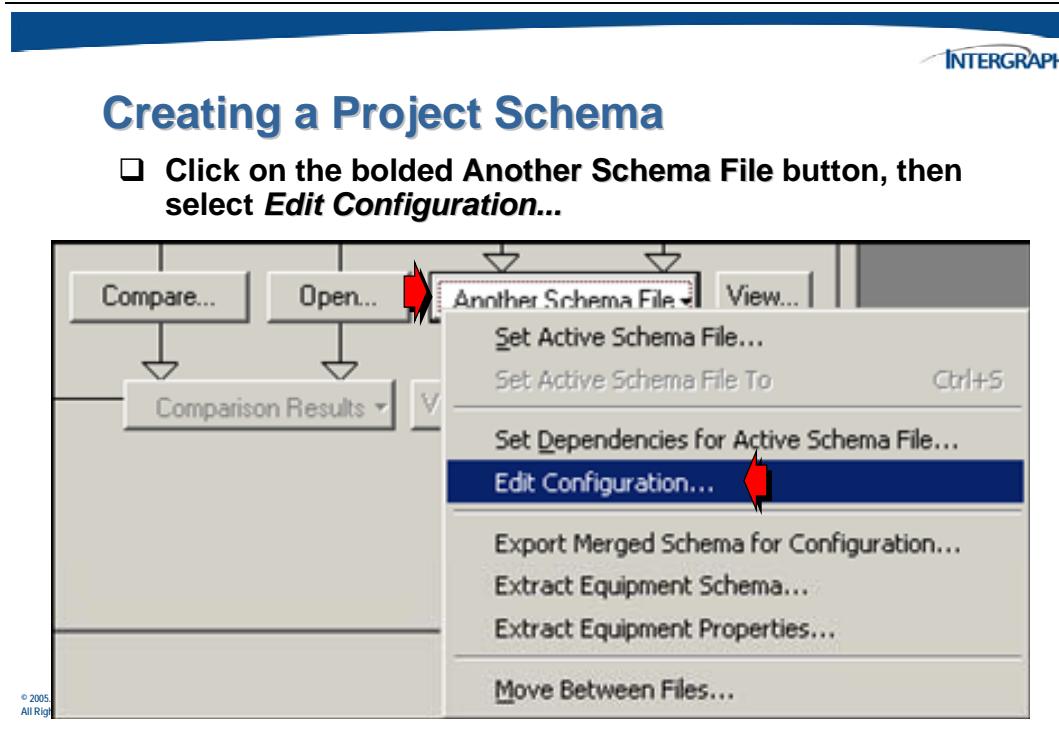
Next, set the file properties for the two files to work together by setting the **Active File**.



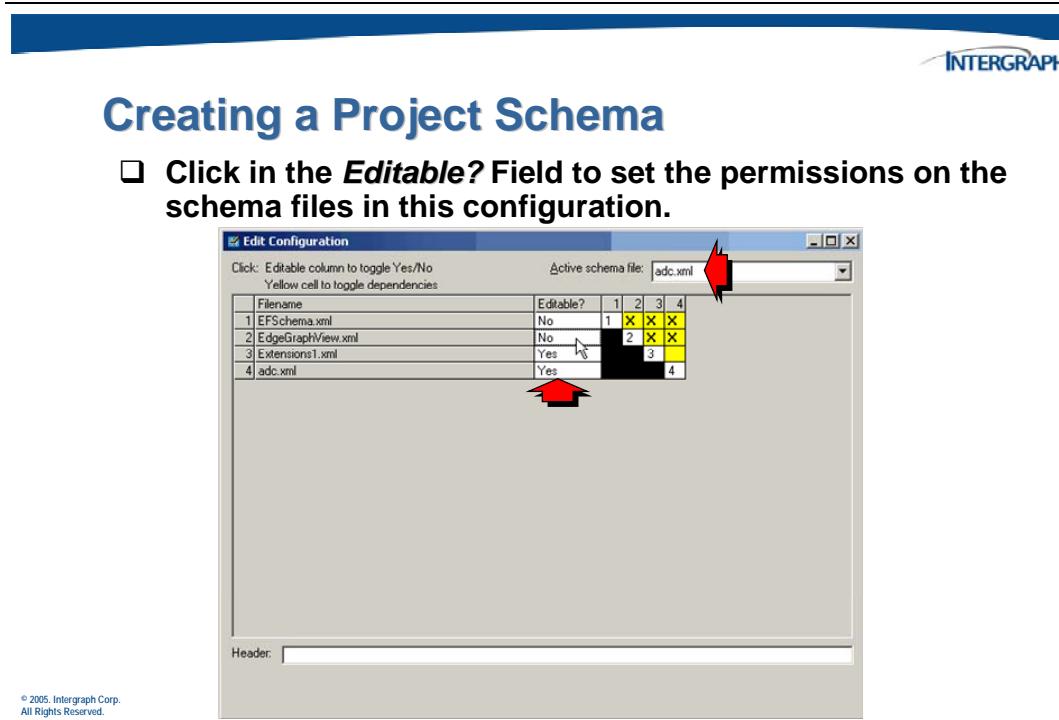
The *Set Active File* dialog will be displayed.



Finally select the *Another Schema File* button again and select the ***Edit Configuration*** option.



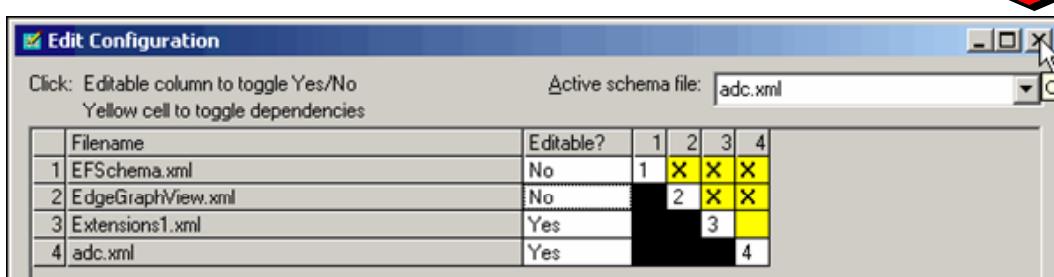
With this dialog you need to make the *EFSchema.xml* and *EdgeGraphView.xml* files **Read Only** and the *adc.xml* **Read/Write**.





Creating a Project Schema

- Click the X to close the *Edit Configuration* dialog.



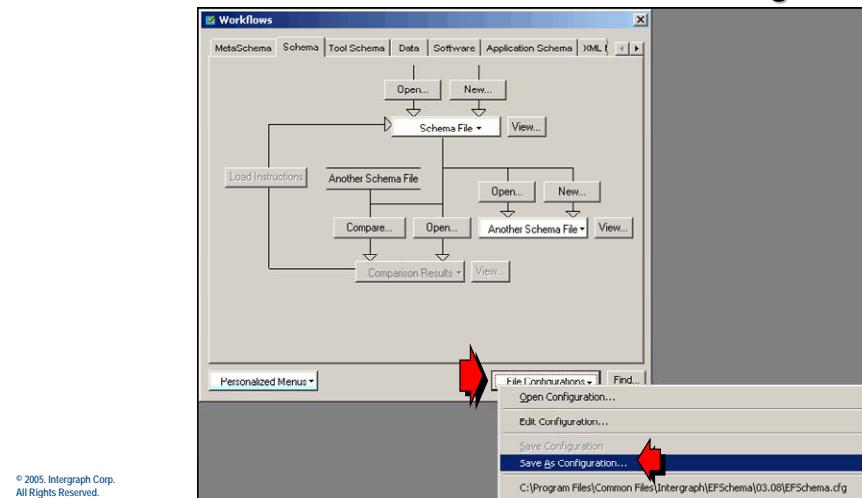
© 2005, Intergraph Corp.
All Rights Reserved.

Now save the configuration for use in future editing sessions.



Creating a Project Schema

- Click the File Configuration button in the lower right corner of the window and choose *Save As Configuration*.

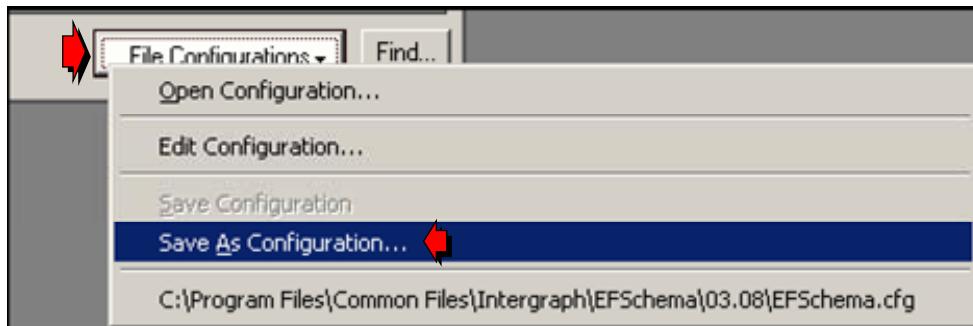


© 2005, Intergraph Corp.
All Rights Reserved.



Creating a Project Schema

Zoom in view of the Save As Configuration.



© 2005, Intergraph Corp.
All Rights Reserved.

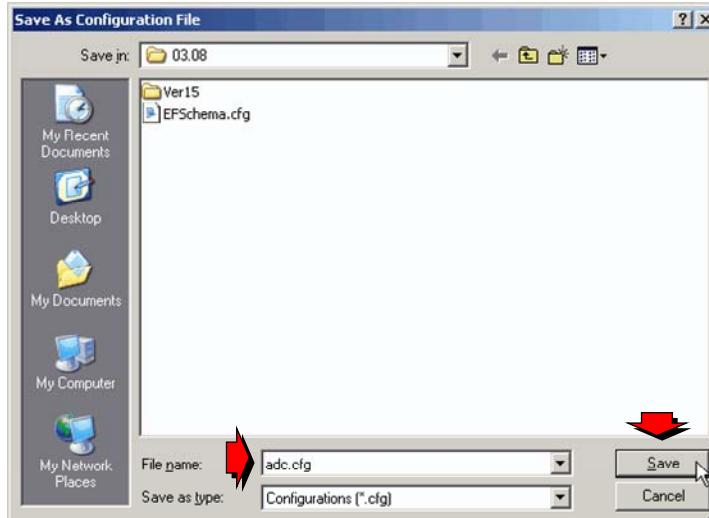
A *Save As Configuration File* window will display to allow you to save the session configuration. Save the configuration as **adc.cfg**.



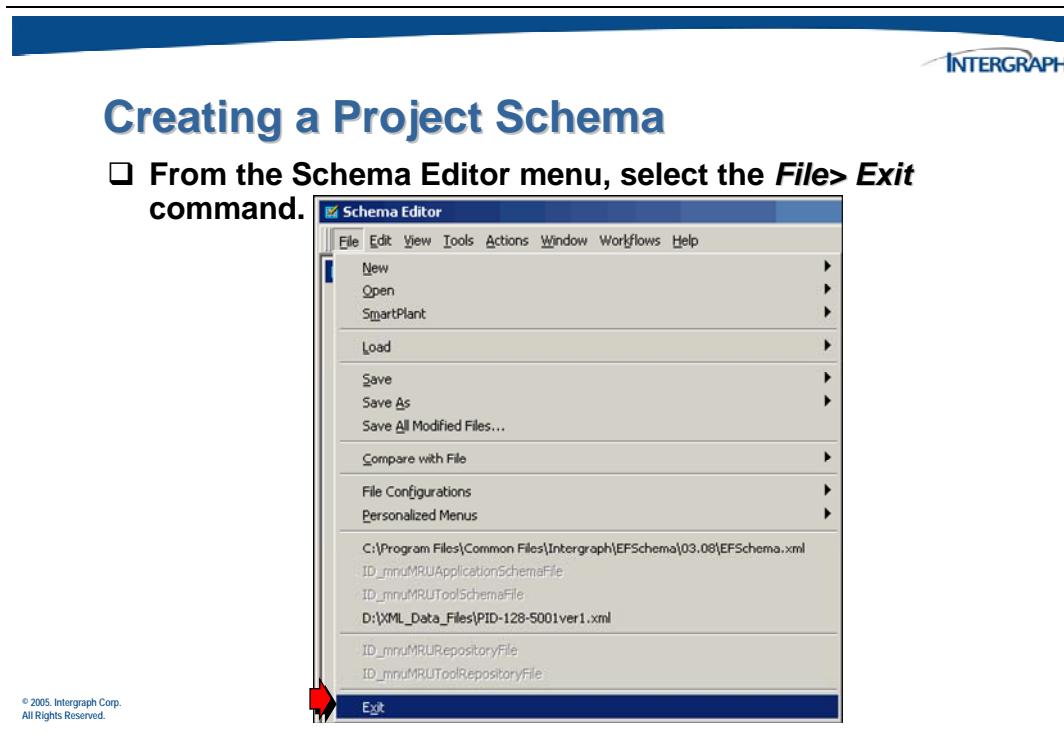
Creating a Project Schema

- Enter a name for the new file configuration and click **Save**.

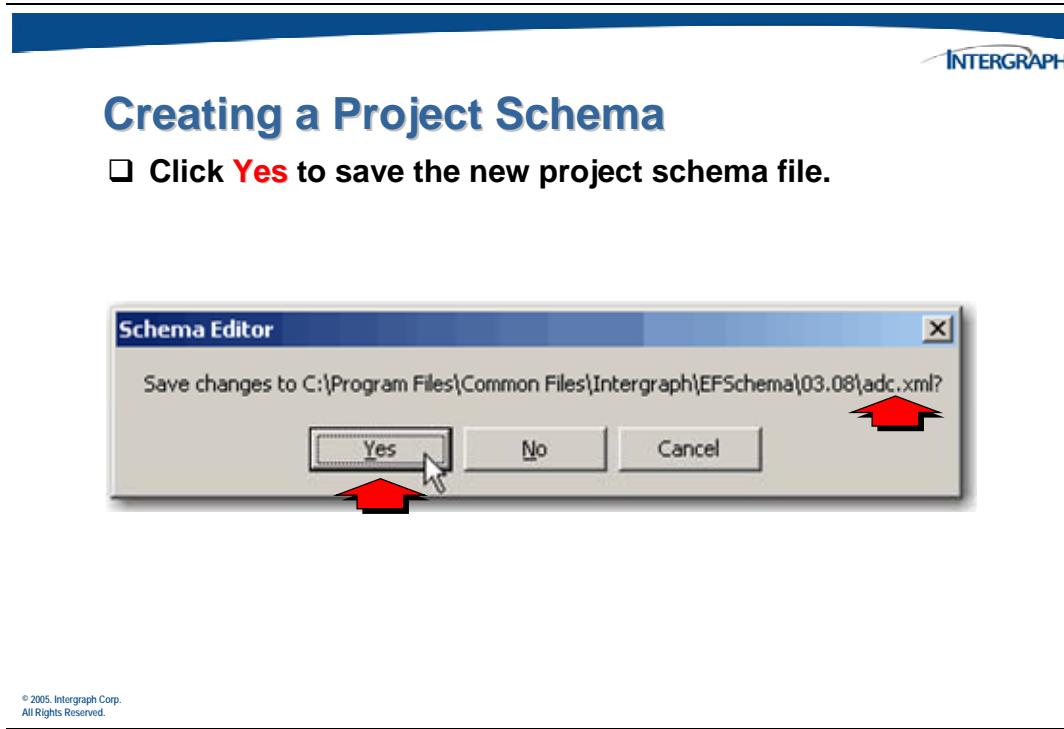
© 2005, Intergraph Corp.
All Rights Reserved.



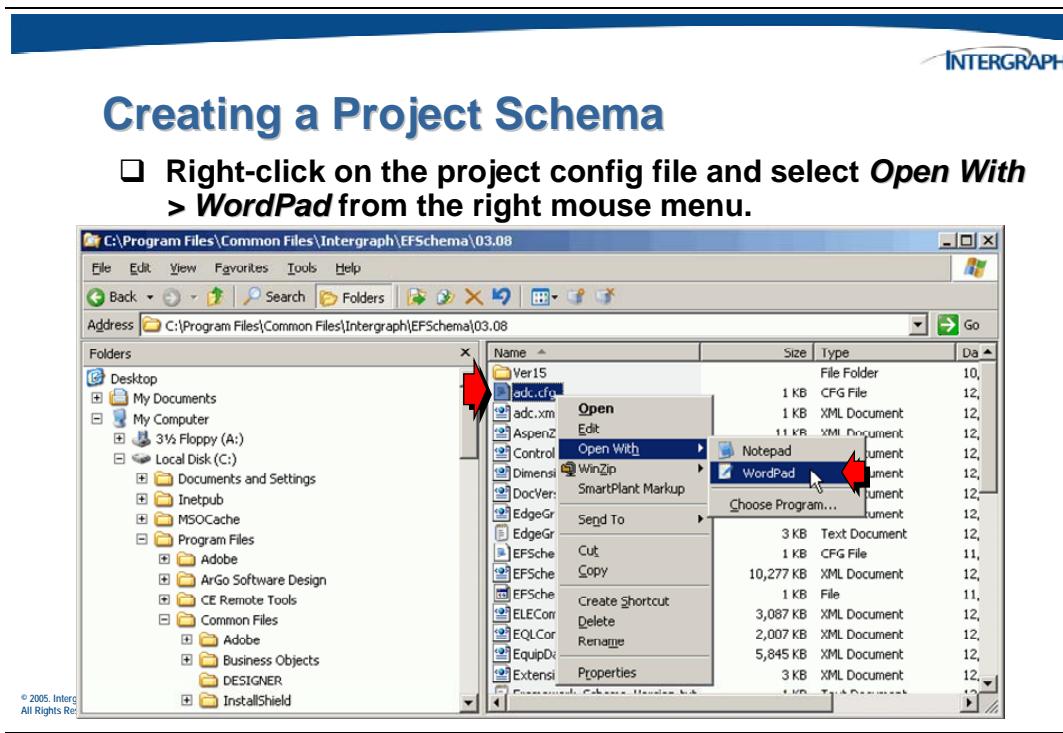
Use the **File** command on the main menu to **Exit** from the Schema Editor.



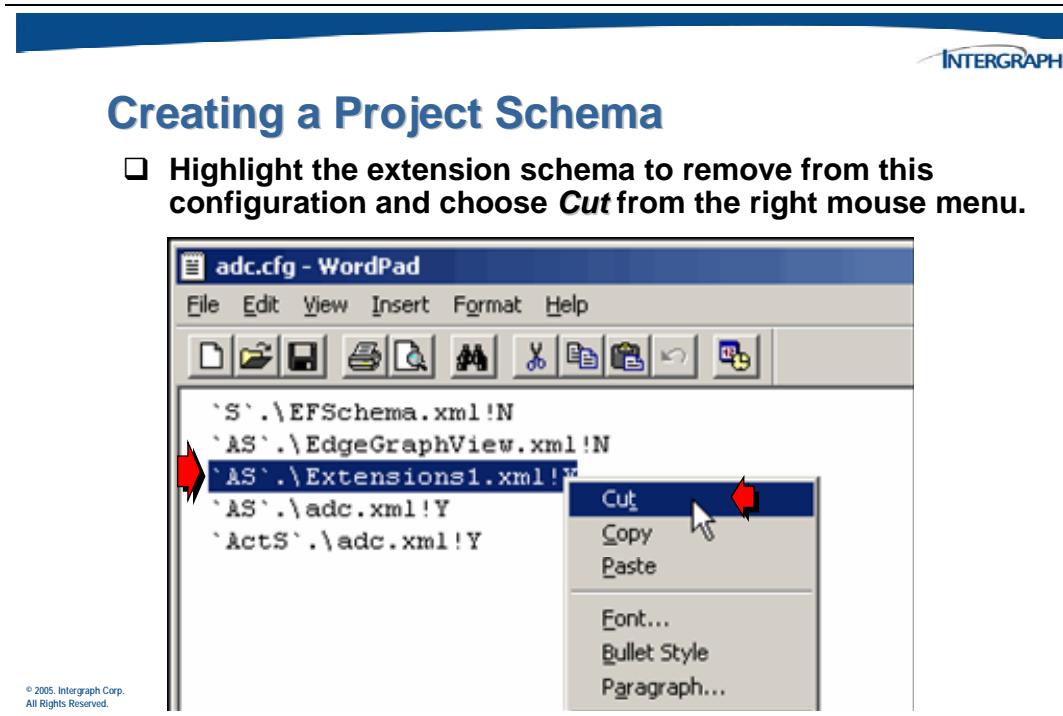
You will be prompted whether or not to save any changes that has been made to the schema.



Edit the new configuration file and remove the default supplied extension schema.



The default supplied extension schema is called **Extensions1.xml** but it won't be used in this custom configuration.

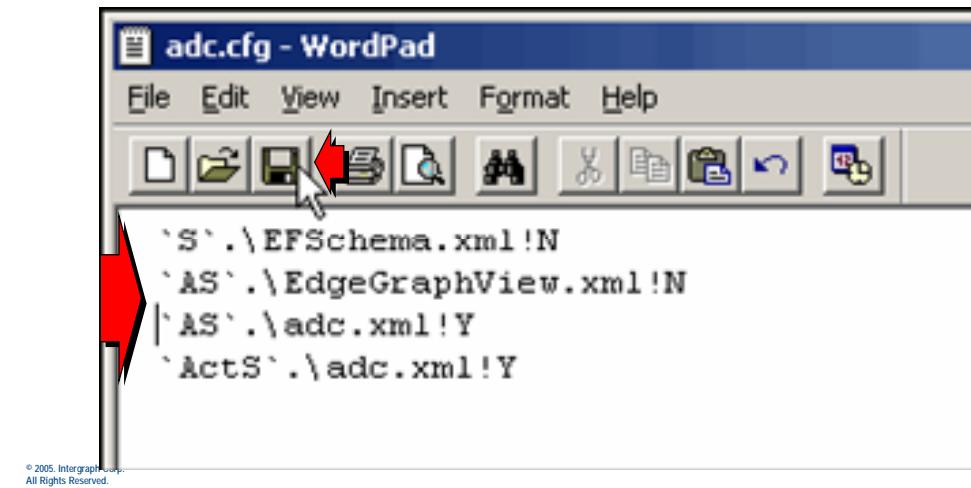


Once the default extension schema has been removed, save the contents of the config file.



Creating a Project Schema

- Use the Save button to save this configuration file and exit from WordPad.

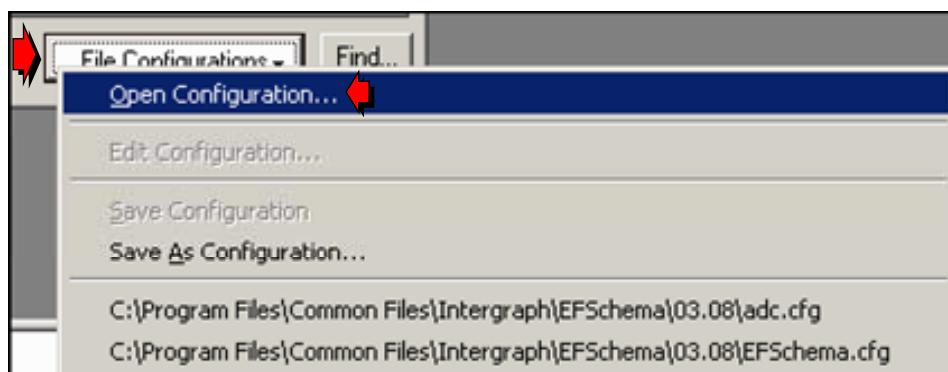


In the future, whenever you edit your schema model, you can use the **File Configuration** button to open the project schema using the cfg file that was created.

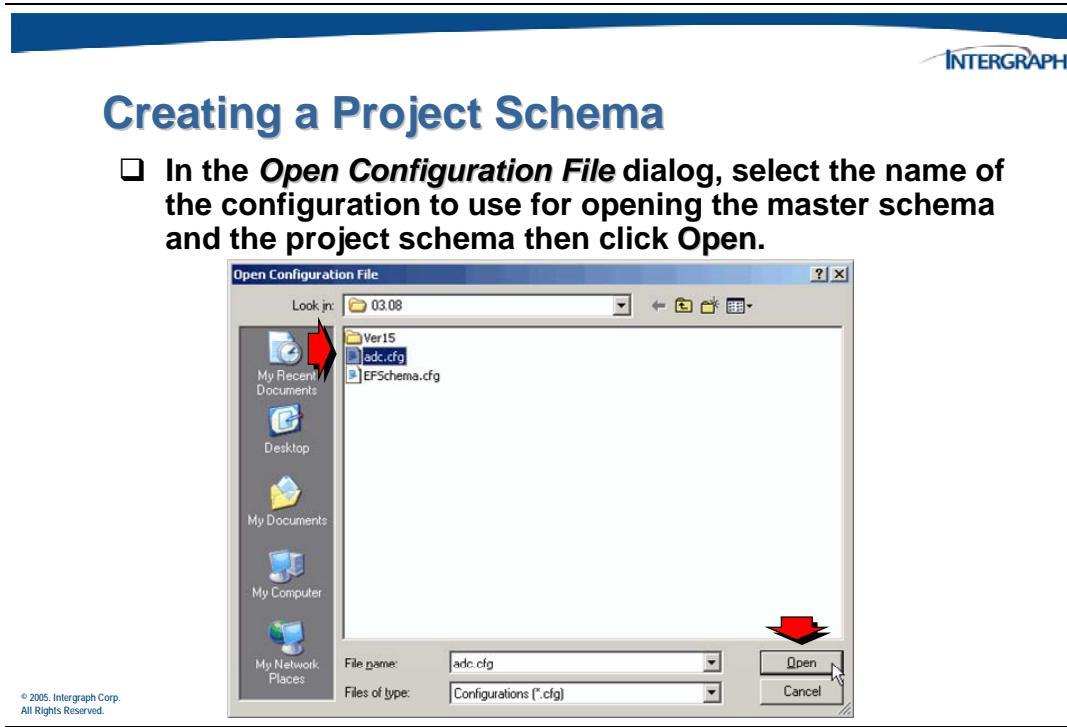


Creating a Project Schema

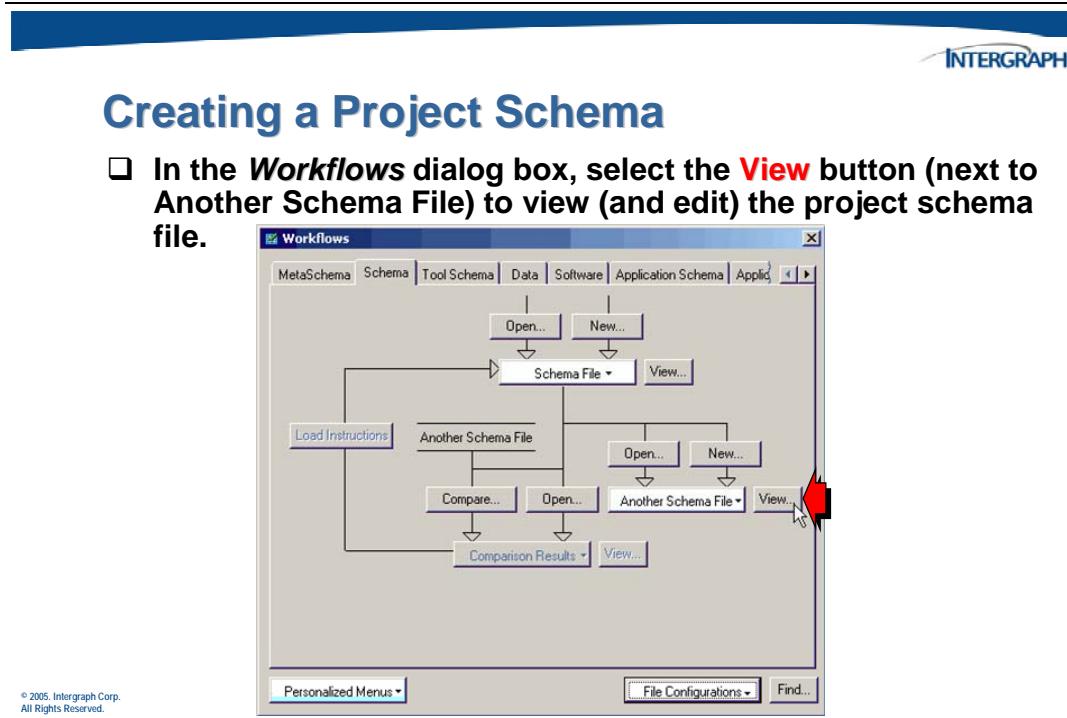
- From the Workflows dialog box, choose the File Configuration button and select *Open Configuration..*



Next select **adc.cfg** to open the saved configuration (EFSchema.xml and adc.xml).

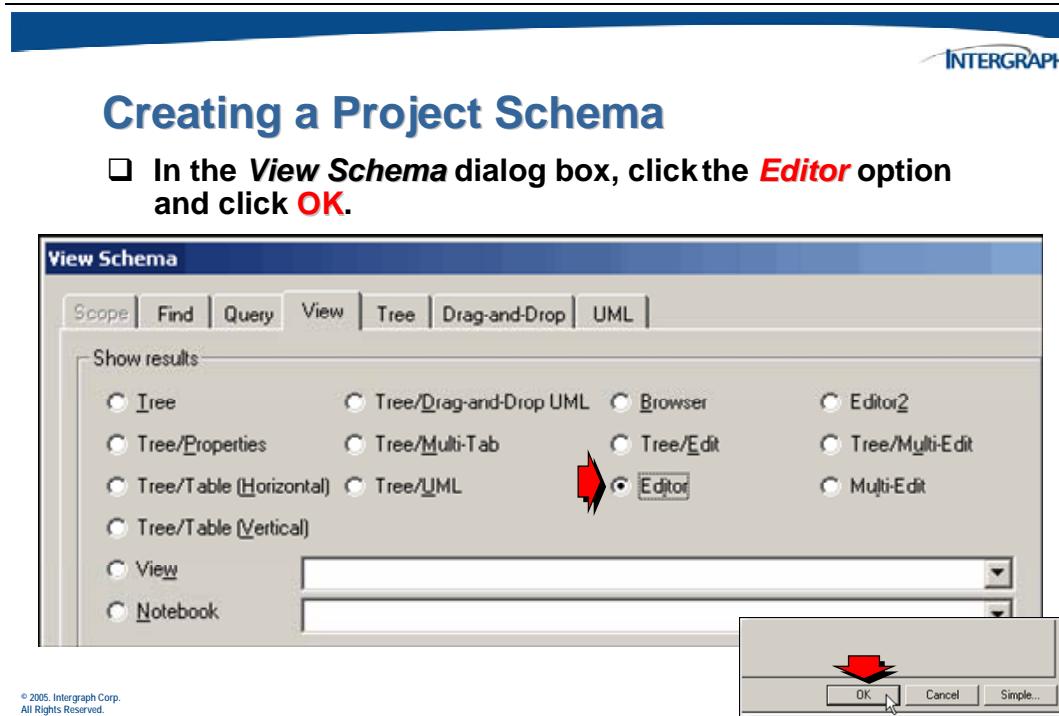


When opened with the configuration file, the **adc.xml** should be the primary edit file. All changes will be made in this file.

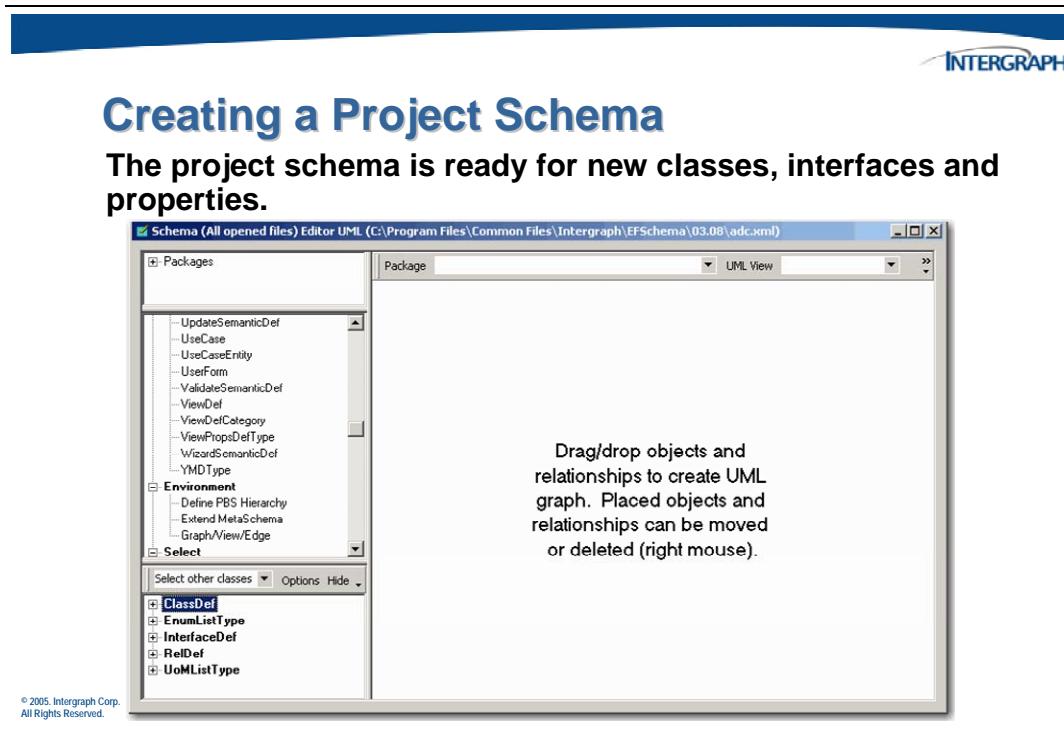


If you select the **View** button next to the Schema File button, you will just see the EFSchema.xml file information. To see both **all schema** data you need to use the **View** button next to the **Another Schema File** button. But understand edits will be written to the adc.xml **not** the EFSchema.xml file because of the configuration that has been set up.

Click the **View** tab in the *View Schema* dialog box.



You are now ready to start adding custom modeling objects to the schema structure.



2.14 Activity 2 – Creating a Project Schema

The goal of this activity is to familiarize you with using the Schema Editor to create a project schema to be used with the activities in chapters 3, 4 and 5. You will start the editor and open the master schema. You will create a project schema and set the configuration so that the project schema will use the master schema as a dependency.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click *Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor* to start the *Schema Editor*.
3. Open the default schema file configuration (EFSchema.cfg).
 - Click the **File Configurations** button in the lower right corner of the application window.
 - When the *Open Configuration File* dialog box displays, select the name **EFSchema.cfg** and click **Open**.

Creating a Project Schema

4. Create a new project schema template file that will contain all of the custom modeling statements.
 - In the *Workflows* dialog box, click the *New* button above the **Another Schema File** button.
 - When the *New Another Schema File* dialog box appears, enter the name **adc.xml** and click **OK**.
 - When the *Set Container Dependencies* dialog box displays, select the master **EFSchema.xml** file, the **EdgeGraphView.xml** extension file and click **OK**.
5. Set the active schema file to be modified during the modeling edit sessions.
 - Click the **Another Schema File** button in the *Workflows* dialog box.
 - Select the **Set Active Schema File...** command
 - When the *Set Active File* dialog box displays, select the **adc.xml** file
 - Click **OK** to dismiss the *Set Active File* dialog.

6. Edit the configuration to be used for both the master schema and the project schema during the modeling edit sessions.
 - Click the **Another Schema File** button in the *Workflows* dialog box.
 - Select the **Edit Configuration...** command
 - When the *Edit Configuration* dialog box displays, make the **EFSchema.xml** file not editable by clicking in the **Editable?** column and toggling the setting to **No**
 - Make the **EdgeGraphView.xml** file not editable by clicking in the **Editable?** column and toggling the setting to **No**
 - Make the **adc.xml** file editable by clicking in the **Editable?** column and toggling the setting to **Yes**
 - Verify that the *Active schema file* is **adc.xml**
 - Close the *Edit Configuration* window (click **X**).
7. Save the configuration so that it can be used in future editing sessions.
 - In the *Workflows* dialog box, click the **File Configurations** button in the lower right corner of the application window.
 - When the *Save As Configuration File* dialog box displays, enter the name **adc.cfg** and click **Save**.
8. Exit the Schema Editor and save your work.
 - Select the **File>Exit** command from the menu
 - When the *Schema Editor* dialog displays, verify that the **adc.xml** file is specified and click **Yes** to save the project schema.
9. Open Windows Explorer and locate the **adc.cfg** configuration file in the C:\Program Files\Common Files\Intergraph\EFSchema\03.08 folder.
10. Edit this config file using WordPad and remove the reference to **Extensions1.xml**.
 - Highlight this line in the file and use the right mouse button **Cut** command to remove it from the file.
 - Save this config file and exit from WordPad.

Testing the Configuration

11. Click **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor** to start the *Schema Editor* once again.

12. Open the configuration that was saved in the steps above in order to open both the master schema (EFSchema.xml) and the project schema (adc.xml).
 - Click the **File Configurations** button in the lower right corner of the application window.
 - When the *Open Configuration File* dialog box displays, select the name **adc.cfg** and click **Open**.

13. View the project schema using the *Editor View*.
 - In the *Workflows* dialog box, click the **View** button beside the **Another Schema File** button.
 - When the *View Schema* dialog box appears, select the **View** tab, choose the **Editor** option.
 - Click **OK**
 - When the view window displays, verify that the project schema (adc.xml) is listed in the title bar.

14. Exit the Schema Editor and save your work.

Once you have closed the Schema Editor, you may take a short break until the other students have finished this activity.

C H A P T E R

3

Schema Overview and Modeling New Classes

3. Meta Schema Concepts

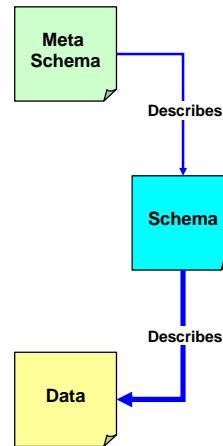
The **Meta Schema** describes the SmartPlant schema and provides the building blocks upon which the SmartPlant schema is built. Because it would be impractical and unwise to write code behind every defined object in the SmartPlant schema, there is code behind every object defined in the meta schema that drives their behavior.



Meta Schema Concepts

The **meta schema** is a set of schema objects that describe the objects in the SmartPlant schema. The meta schema defines the language in which the SmartPlant schema is written.

The SmartPlant schema is a set of schema objects that describe the data that is published by the authoring tools.



© 2005, Intergraph Corp.
All Rights Reserved.

The meta schema classes do not directly describe the **data** that is part of the schema (the SmartPlant schema describes the data). Instead, these classes describe the **classes** in the schema. The meta schema defines the rules of the schema and is the code behind the schema.

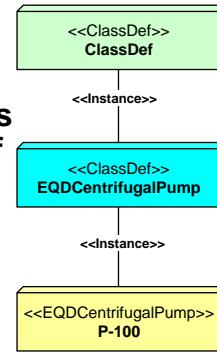
For example, all classes in the schema are instances of the *ClassDef* class in the meta schema. Similarly, all relationship definitions in the schema are of class *RelDef*, which is part of the meta schema.

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

Meta Schema Concepts

In this example, **ClassDef** is the meta schema object that describes the **EQDCentrifugalPump** object in the schema. The EQDCentrifugalPump class in the schema is an instance of ClassDef in the meta schema.

The EQDCentrifugalPump class in the schema describes **P-100**, which is the data. P-100 is an instance of the EQDCentrifugalPump class.



More information on the meta schema classes will be discussed in the following sections of this chapter.

3.1 Model Definitions

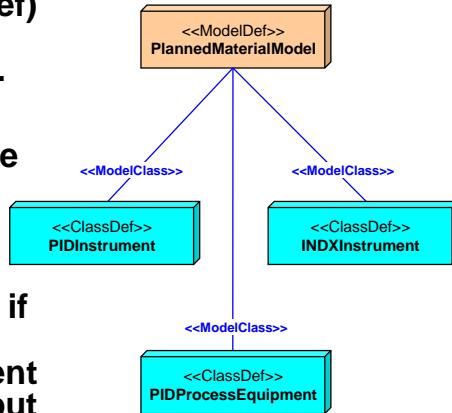
A model is a non-redundant description of an object. For example, a 3D plant model is a non-redundant 3D description of a plant (non-redundant = non-excessive). Model definitions in the SmartPlant schema are roughly based on the POSC-CAESAR model.

Model Definitions

A **model definition** (ModelDef) is the schema information used to describe the model.

The primary function of a ModelDef is to identify those objects that can be shared through a ModelClass relationship.

Objects can only be shared if they exist within the same ModelDef. Objects in different ModelDefs can be related, but cannot be shared.



© 2005, Intergraph Corp.
All Rights Reserved.

The only use for the ModelDef currently is to enforce a rule that ClassDefs must be in the same ModelDef to participate in a *Sharing* relationship.



Model Definitions

ModelDefs define domains of data. There are seven ModelDefs in the SmartPlant schema:

- MetaModel**
- OrganizationModel**
- WorkBreakdownModel**
- FacilityModel**
- PlannedMaterialModel**
- TypicalMaterialModel**
- ActualMaterialModel**

© 2005, Intergraph Corp.
All Rights Reserved.

MetaModel - Used only by the ClassDefs in the meta-schema. You should never use this model for Schema.

OrganizationModel - Like the name implies, this is intended for ClassDefs that are organizational concepts. Examples would be User, Company, Department, Manufacturer, Vendor, etc.

WorkBreakdownModel - For ClassDefs that provide organizations of work. This includes the ClassDefs that you see in the WBS documents such as Project and Contract as well as all the document ClassDefs defined in the schema.

FacilityModel - For ClassDefs that represent abstract concepts in a process plant. Examples include Plant, Area, Unit, System, Discipline (although you might put Discipline in the OrganizationModel instead.)

PlannedMaterialModel - For ClassDefs that represent real objects that you plan to purchase to construct your design. This is the “design” arena in which engineering tools like Zygad, SPPID, and SmartPlant Instrumentation work. In a traditional SPF Model, this would be the “tag”.

TypicalMaterialModel - For ClassDefs that represent real objects offered by manufacturers or vendors. When an engineer goes looking for something to fulfill the requirements of his design, he might look through a manufacturer catalog. This is the domain of this model. In a traditional SPF Model, this would be the “model”, or “catalog”, or “standard equipment”.

ActualMaterial - For ClassDefs that represent what is purchased and assembled to construct the plant. In a traditional SPF 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 ModelDefs are part of the meta schema and are therefore predefined.

3.2 Component Schemas

A **Component Schema** is a subdivision of the complete SmartPlant schema and is a set of class definitions that are used within a specified domain or application area. Class definitions tie a primary interface definition to a component schema. Class definitions and interface definitions will be discussed in more detail later.

The component schema breaks up the SmartPlant schema into manageable chunks that correspond almost one to one to a document type being published. At a minimum, there is a component schema defined for each tool that publishes into SmartPlant. You can extract component schemas from the SmartPlant schema using the Schema Editor.

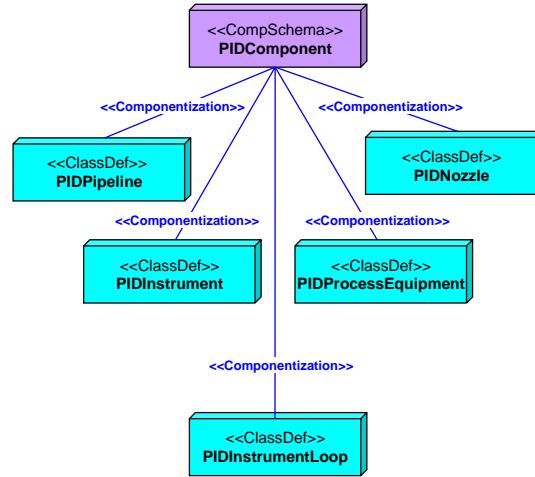
One of the primary purposes of the component schema is performance. When a tool indicates it is going to publish a certain document type, the SmartPlant Client loads only the necessary component schema instead of the entire SmartPlant schema.



Component Schemas

A component schema (CompSchema) is a set of class definitions that are used within a specified domain or application area.

For example, the PIDComponent schema contains all the class definitions that are relevant to P&IDs.





Component Schemas

Component schemas contain the following:

- The set of class definitions that have a componentization relationship with the component schema**
- The interface definitions realized by those class definitions**
- The property definitions exposed by those interface definitions**
- The property types scoped by those property definitions**

© 2005, Intergraph Corp.
All Rights Reserved.

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. Shared object definitions will be discussed later in this chapter.

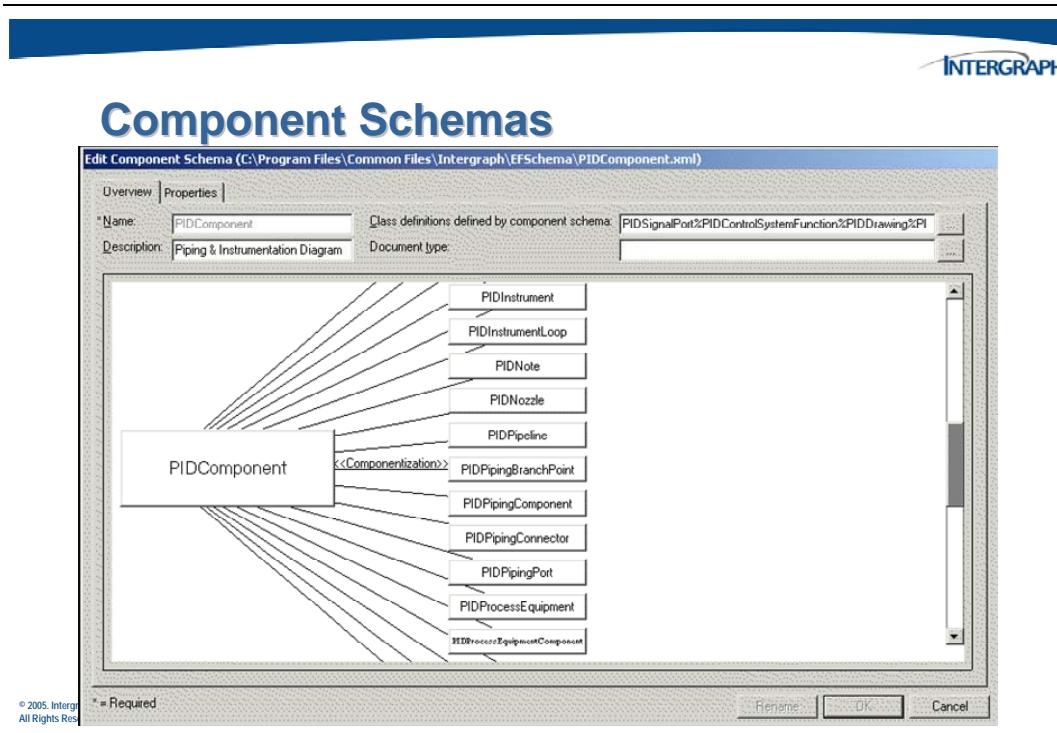
For example, if an object in one component schema is classified as class AA1 and the same object with the same UID is classified as class AA2 by another component schema, then the shared object in the SmartPlant Foundation database contains the information associated with the union of classes AA1 and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

Editing of the component schema should always be done against the SmartPlant schema files from which the component schema files were generated rather than the component schema files themselves (because they can be regenerated).

3.2.1 Properties of a Component Schema

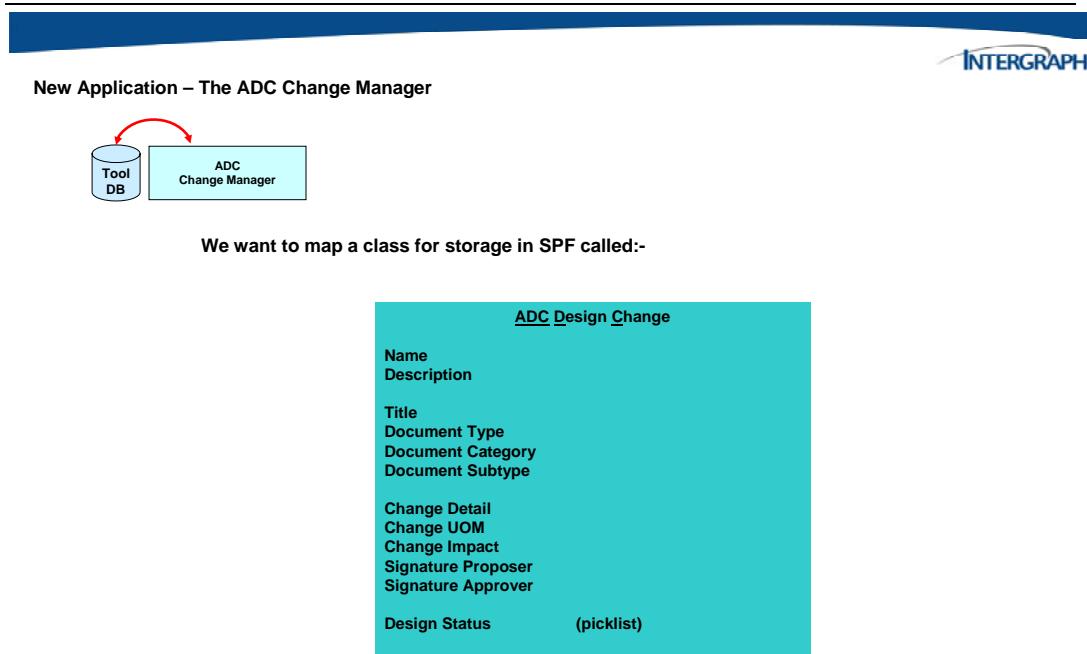
Component schemas have the following properties:

- ❑ **Name** – Specifies the name of the component schema.
- ❑ **Description** – Specifies the description of the component schema.
- ❑ **Class definitions defined by component schema** – List all class definitions that have a componentization relationship with the component schema. Because validation of published XML is done using the component schema instead of the SmartPlant schema, the component schema should contain all class definitions required to publish a particular document type.
- ❑ **Document type** – Specifies which document types the component schema scopes. This setting is currently optional because the SmartPlant Client queries the adapter for this relationship during a publish.



3.3 Interactive Activity – Creating a Component Schema

For the interactive activities in this chapter, we will be creating a new component schema as well as all the necessary definitions to add some custom attributes to the schema. The following slides will explain the process that you will use throughout this chapter to accomplish this task.



This is an example of **Design Change Document** that needs to be implemented. What you see depicted above is a form that will be used to gather the necessary information to specify the change.

First, we will need a new **Component Schema** to contain the type of document to be published.

New Application – The ADC Change Manager

Step 1 – Create a Component Schema to govern the document types to be published and assign it to a suitable Model Definition.

Component Schema :: ADCCComponent

Want to map a class ADC Design Change for Storage in SPF:-

<Comp Schema> ADCCComponent

ADC Design Change

- Name
- Description
- Title
- Document Type
- Document Category
- Document Subtype
- Change Detail
- Change UOM
- Change Impact
- Signature Proposer
- Signature Approver
- Design Status (picklist)

© 2005, Intergraph Corp.
All Rights Reserved.

Since a new type of document will need to be published, a new class of document must be defined which is called a Class Definition.

New Application – The ADC Change Manager

Step 2 – Create a class definition for the ADC Design Change

Want to map a class ADC Design Change for Storage in SPF:-

<Comp Schema> ADCCComponent

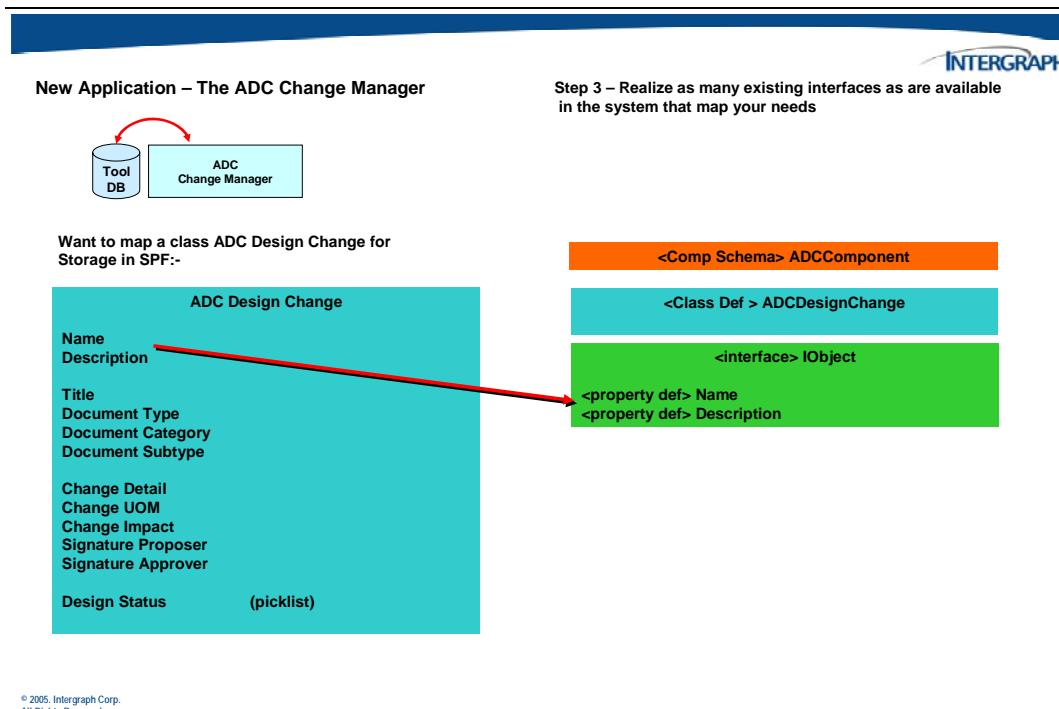
<class def > ADCDesignChange

ADC Design Change

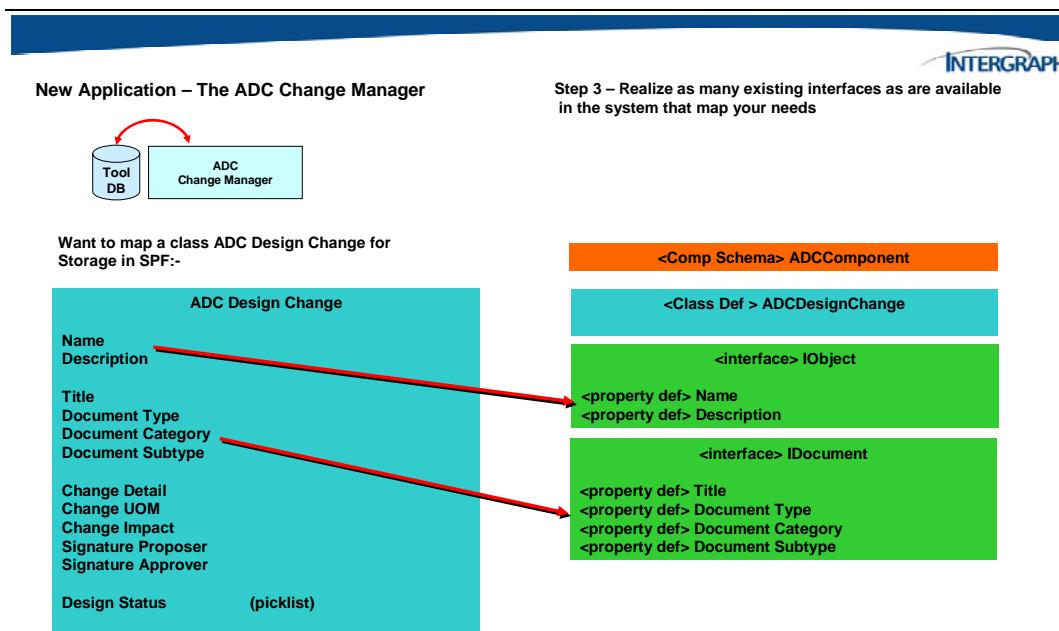
- Name
- Description
- Title
- Document Type
- Document Category
- Document Subtype
- Change Detail
- Change UOM
- Change Impact
- Signature Proposer
- Signature Approver
- Design Status (picklist)

© 2005, Intergraph Corp.
All Rights Reserved.

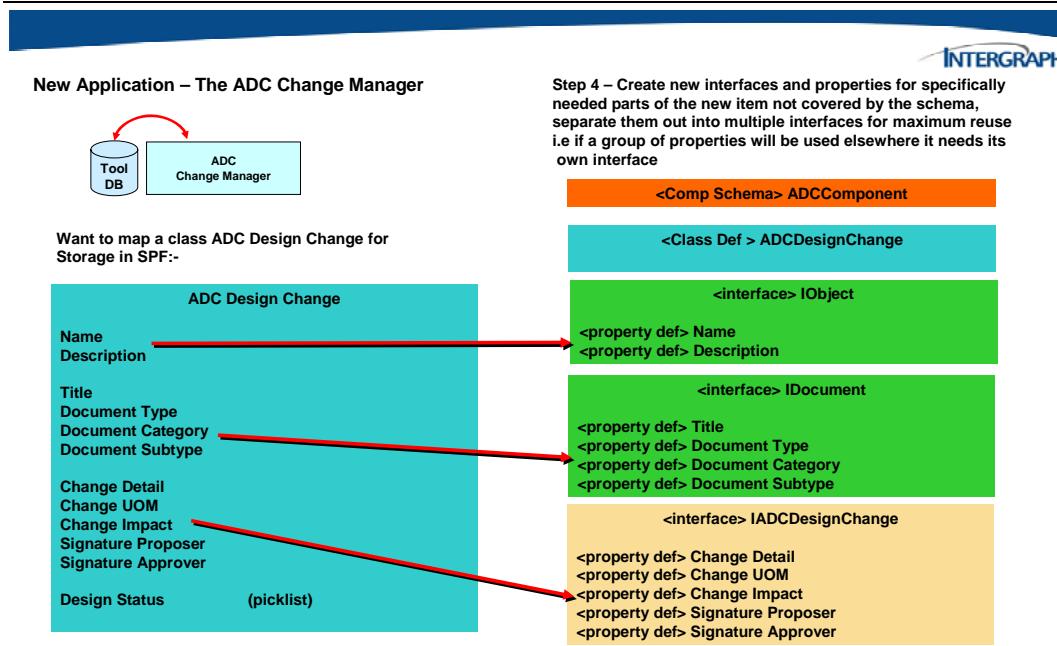
In a couple of instances, the necessary attributes or **properties** needed to provide the necessary information have already been defined in the system and associated with an **interface**.



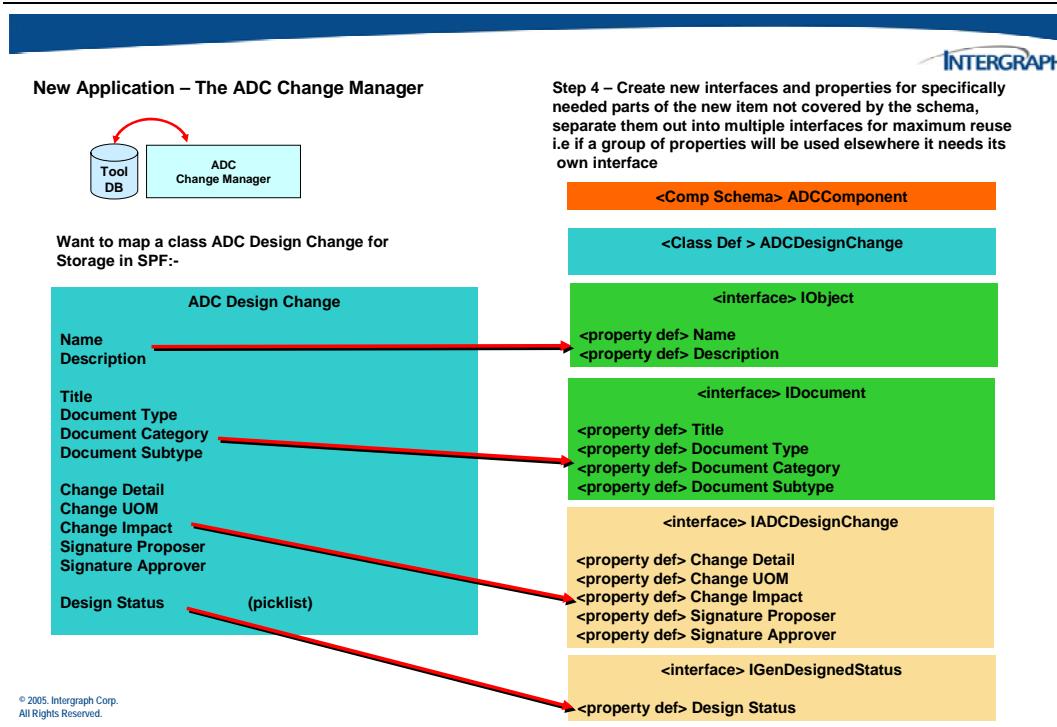
Rather than re-define existing objects, you can reference those interfaces and utilize their properties.



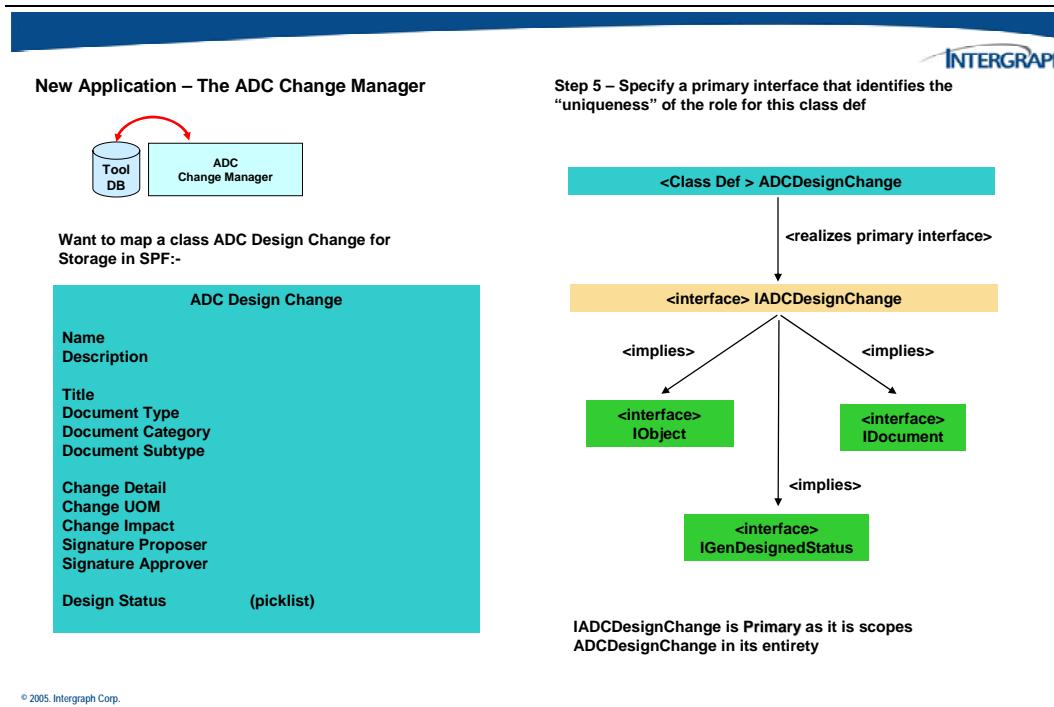
You will also need to define **new** interfaces and properties to complete the mechanism to capture all of the needed information for this new document type.



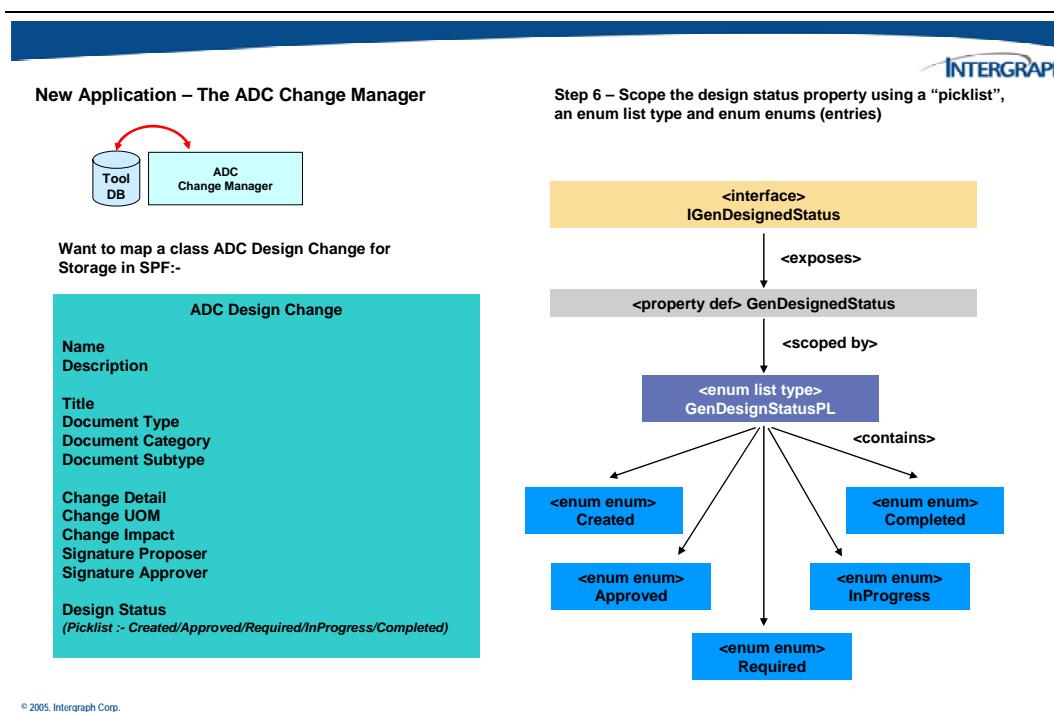
A portion of the needed data can utilize a picklist or **enumerated list**.



As part of defining a new interface for a new class (ClassDef), a **primary interface** for the class should be specified.



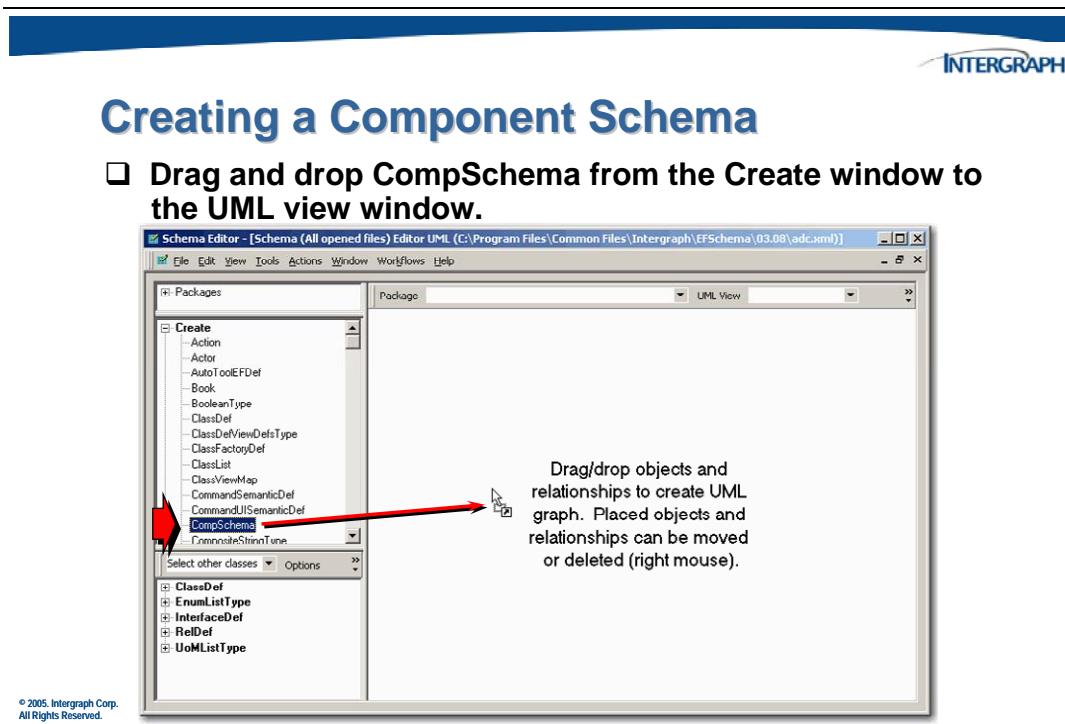
One of the last steps will be to specify the values used for the picklist.

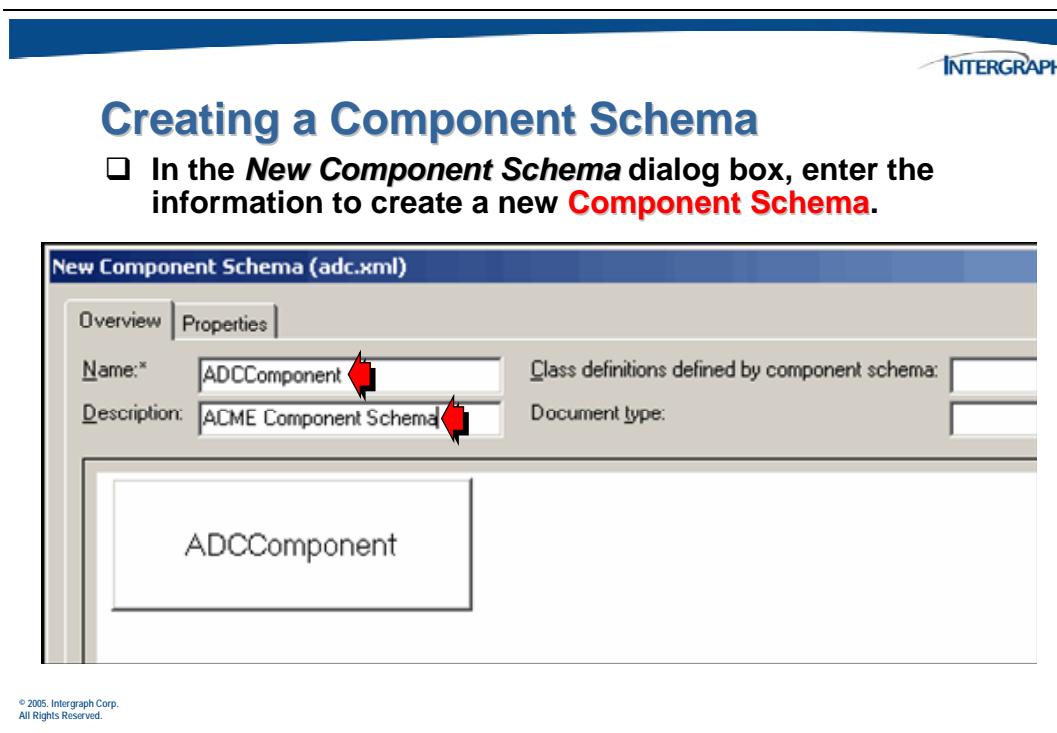
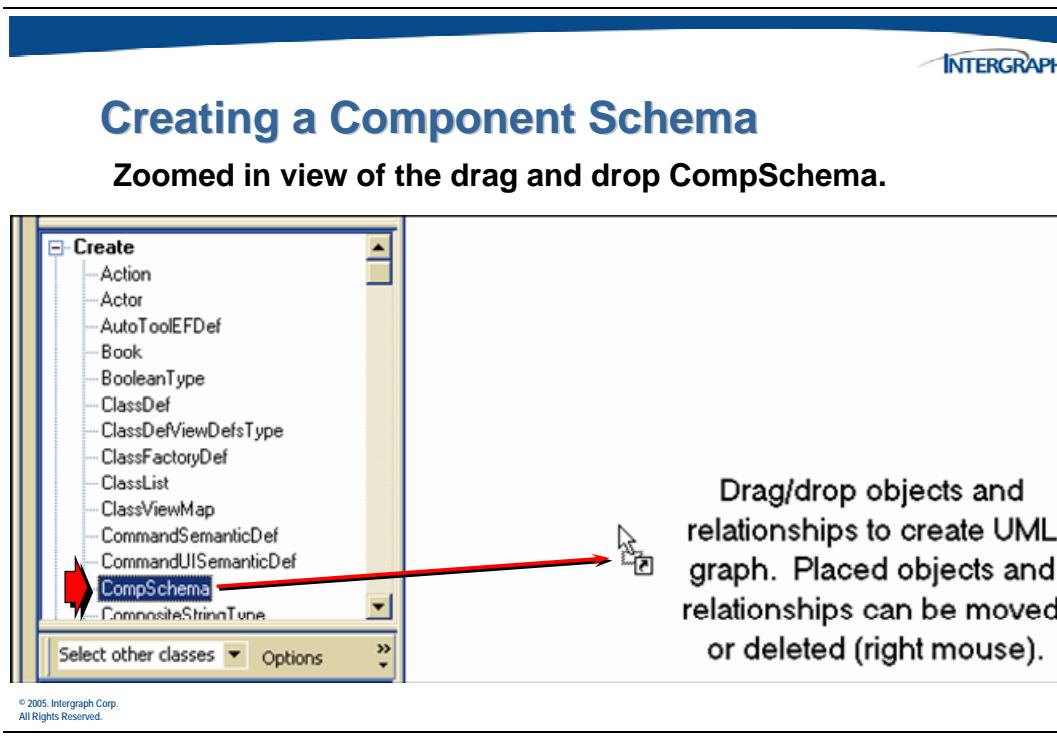


Hands on:

Please follow along with the instructor in creating a new component schema. **Do not close** your Schema Editor window as this activity will continue with additional components.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
3. Open the project schema XML file.
 - Click the **File Configurations** button and select **Open Configuration...** from the *Schema Editor* Workflows dialog
 - Select the **adc.cfg** file and click **Open** (Path is C:\Program Files\Common Files\Intergraph\EF Schema\03.08)
 - Use the **Editor** view to perform your modeling tasks.





3.4 Class Definitions

Class definitions are typically real-world objects like instruments, equipment, or pipe runs. Class definitions have the following characteristics:

- ❑ Every instance of a class definition is instantiated by a class factory.
- ❑ 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. Interface definitions will be discussed in the next section.



Class Definitions

A **class definition (ClassDef)** is a named description of a set of objects that specifies the attributes (data elements) and methods (member functions) for those objects in the schema.



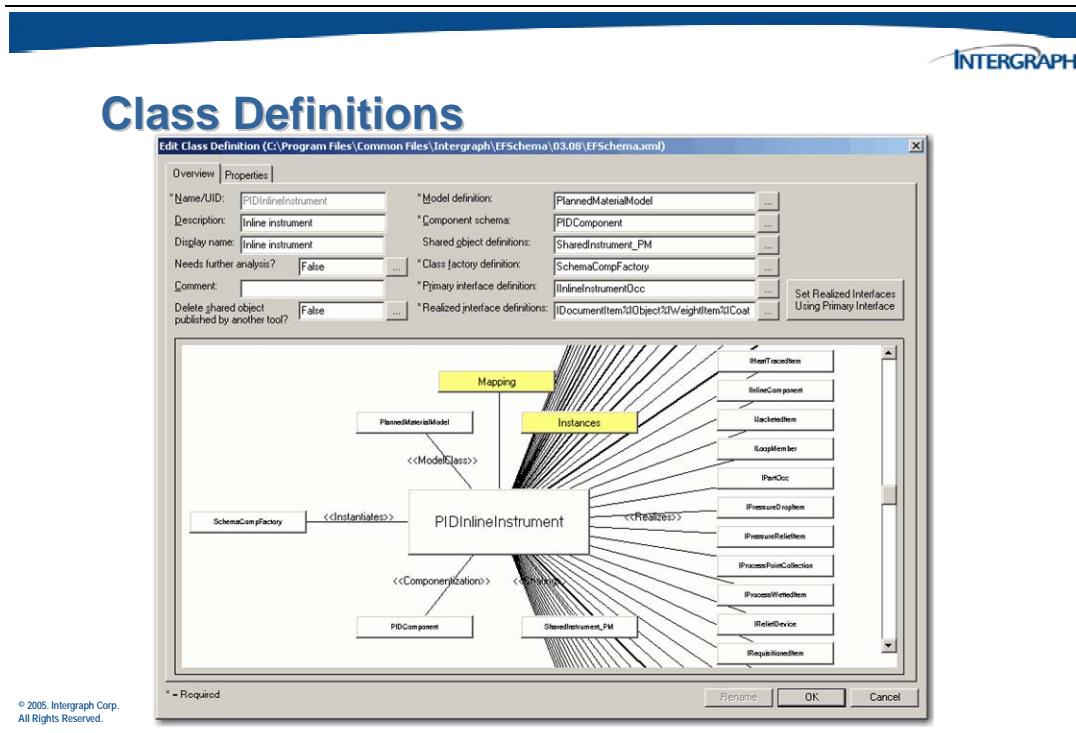
In the schema, ClassDefs can represent physical things, such as pumps and instruments, or conceptual things, such as projects.

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

3.4.1 Properties of a Class Definition

The properties of a class definition include the following:

- Name** – Specifies the name of the class definition.
- Description** – Specifies a description of the class definition.
- Display name** – Specifies the name that you want the user interface to use when displaying the class definition.



- Delete shared object published by another tool** – Specifies what happens when SPF receives a delete instruction for a shared object. SPF either deletes all ClassDefs/UIDs for the shared object or only the ClassDef/UID referenced by the delete instruction. For example, a shared object CV-2001 is published by SmartPlant P&ID as PIDInlineInstrument and by SmartPlant Instrumentation as INDXInstrument. If SmartPlant P&ID publishes a delete instruction and this option is false on the PIDInlineInstrument ClassDef, then the UID for the PIDInlineInstrument object will be terminated in SPF and the INDXInstrument object will remain in the SPF database. If the option is set to true, both UIDs will be terminated in SPF.
- Model definition** – Specifies the model definition to which this class definition belongs. Class definitions can only participate in a sharing relationship if they exist in the same model.

- Component schema** – Specifies the component schema to which the class definition belongs. A component schema is a subdivision of the SmartPlant schema. There is typically one component schema per published document type.
- Shared object definitions** – Specifies the shared object definitions with which this class definition has a sharing relationship. Shared object definitions define the same object in different authoring tools.
- Class factory definition** – Specifies the class factory used to create the class definition. There is currently only one class factory in the SmartPlant schema: SchemaCompFactory.
- Primary interface definition** – Identifies the primary interface definition for the class definition. The primary interface definition defines the set of possible roles for a ClassDef and should imply everything that is known about the ClassDef.
- Realized interface definitions** – Identifies all interface definitions realized by the class definition.
- Set Realized Interfaces Using Primary Interfaces** – Displays the **Identify Realized Interfaces Using Primary Interface** dialog box. This dialog box allows you to select which optional interfaces implied by the class definition's primary interface that you want the class definition to realize.

3.5 Interactive Activity – Creating a Class Definition

New Application – The ADC Change Manager

Step 2 – Create a class definition for the ADC Design Change

Want to map a class ADC Design Change for Storage in SPF:-

The screenshot shows the ADC Change Manager interface. At the top, there's a diagram with two boxes: 'Tool DB' and 'ADC Change Manager', connected by a curved arrow pointing from 'Tool DB' to 'ADC Change Manager'. Below this, a message says 'Want to map a class ADC Design Change for Storage in SPF:-'. To the right, it says 'Step 2 – Create a class definition for the ADC Design Change'. A large central window displays the 'ADC Design Change' class definition. It includes fields for Name, Description, Title, Document Type, Document Category, Document Subtype, Change Detail, Change UOM, Change Impact, Signature Proposer, Signature Approver, and Design Status (picklist). To the right, a sidebar shows the component schema mapping: '<Comp Schema> ADCComponent' and '<class def > ADCDesignChange'.

© 2005. Intergraph Corp.
All Rights Reserved.

Creating a Class Definition

- Choose the browse (...) button to display the dialog to create a new class definition for this component schema.

The screenshot shows a dialog box titled 'Creating a Class Definition'. It has two main input fields: 'Class definitions defined by component schema:' and 'Document type:'. Both fields have dropdown menus with ellipsis buttons (...). A red arrow points to the ellipsis button in the 'Document type:' field, indicating where to click to open the creation dialog.

© 2005. Intergraph Corp.
All Rights Reserved.

Creating a Class Definition

- ❑ Select the **New** button from the **Possible ComponentClassDefs** for **ADCComponent** dialog box.

The screenshot shows a dialog box titled "Possible ComponentClassDefs for ADCComponent". At the top are two input fields: "Value:" and "Starts with:". Below them is a scrollable list of class names under the heading "All". A red arrow points to the "New..." button at the bottom of the list.

© 2005, Intergraph Corp.
All Rights Reserved.

Creating a Class Definition

- ❑ Enter the information to create the new ClassDef

The screenshot shows a dialog box titled "New Class Definition (adc.xml)". It has tabs for "Overview" and "Properties". The "Overview" tab is selected. It contains the following fields:

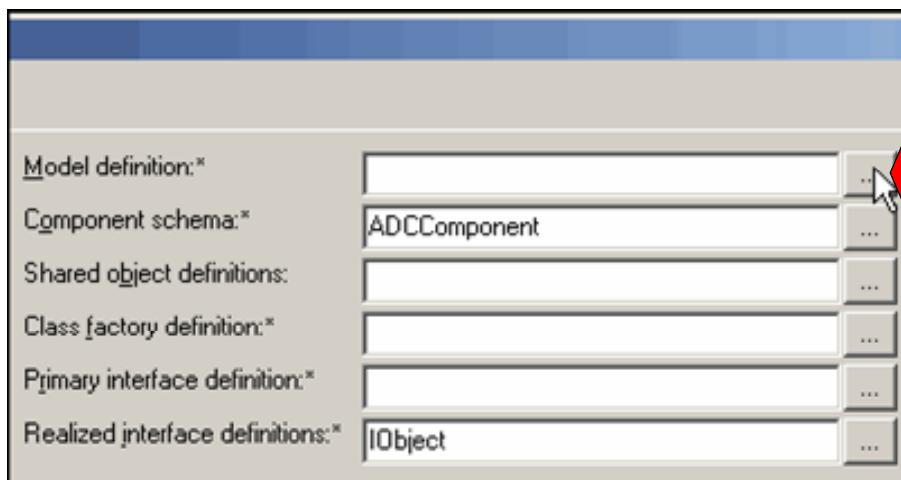
Name/UID:*	ADCDesignChange
Description:	ACME Design Change Note
Display name:	ADC Design Change
Needs further analysis:	False
Comment:	(empty)
Delete shared object published by another tool:	False

© 2005, Intergraph Corp.
All Rights Reserved.



Creating a Class Definition

- choose the **Model definition** browse (...) button

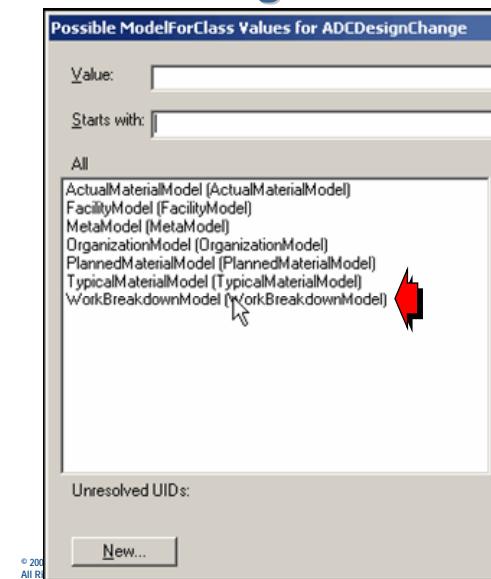


© 2005, Intergraph Corp.
All Rights Reserved.



Creating a Class Definition

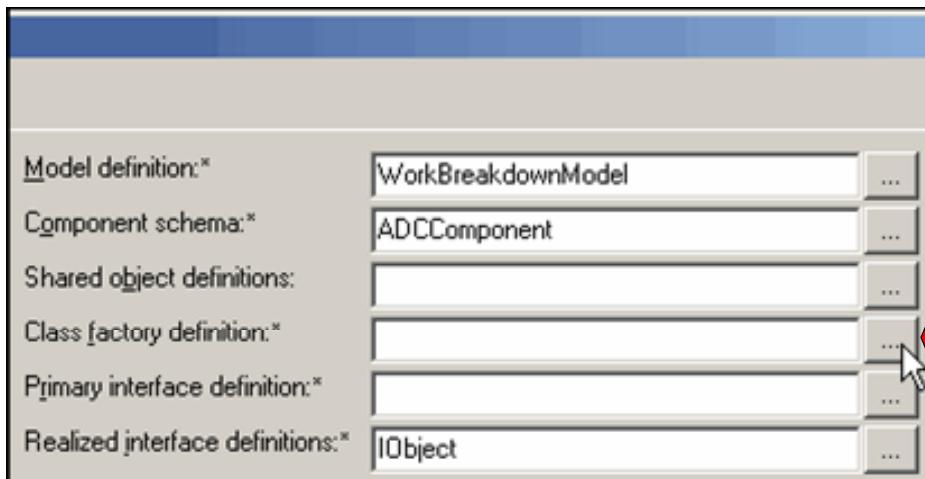
- From the **Possible ModelForClass Values for ADCDesignChange** choose **WorkBreakdownModel**.





Creating a Class Definition

- Choose the *Class factory definition* browse (...) button.

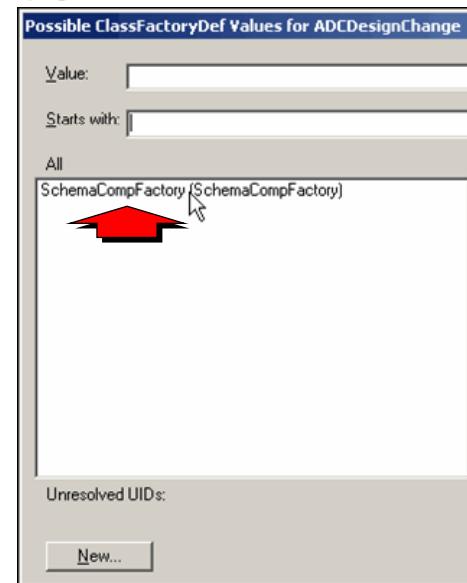


© 2005, Intergraph Corp.
All Rights Reserved.



Creating a Class Definition

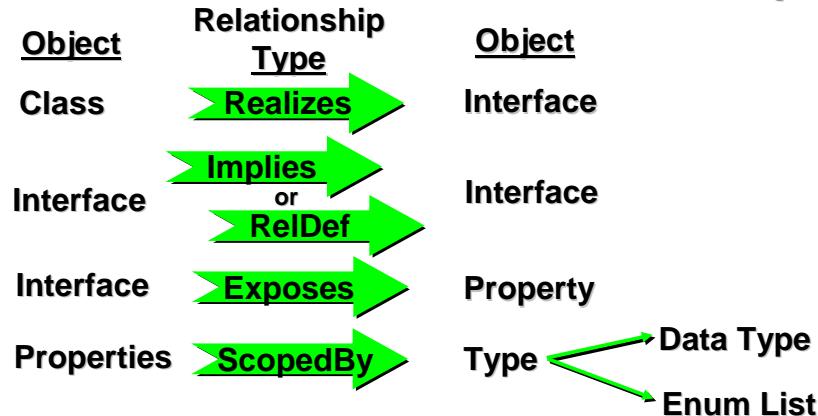
- From the *Possible ClassFactoryDef Values for ADCDesignChange* choose **SchemaCompFactory**.



© 2005, Intergraph Corp.
All Rights Reserved.



SmartPlant Associations and Relationships



Navigation Relationships

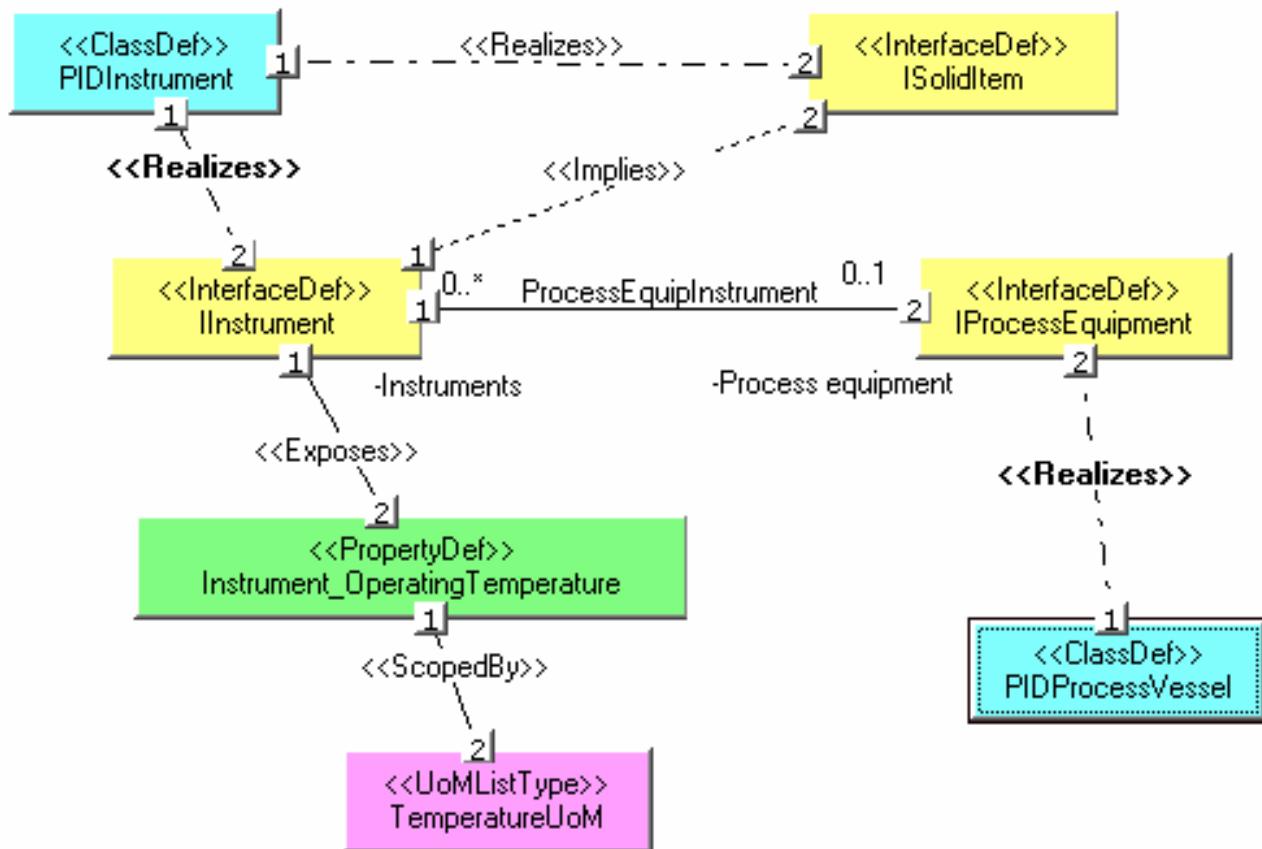
Interface from/to Interface

© 2005, Intergraph Corp.
All Rights Reserved.

A reminder from the previous chapter that the two types of relationships used from interface to interface are:

- RelDefs – these are user defined relationships defined by the data modeler and are used to associate interfaces that are realized by different class definitions (ClassDef's).
- Implies – this type of relationship is thought of as a system default relationship. It is used primarily for inheritance from one interface to another interface where both are realized by the **same** ClassDef.

The following example shows how these different types of relationships are used.



3.6 Interface Definitions

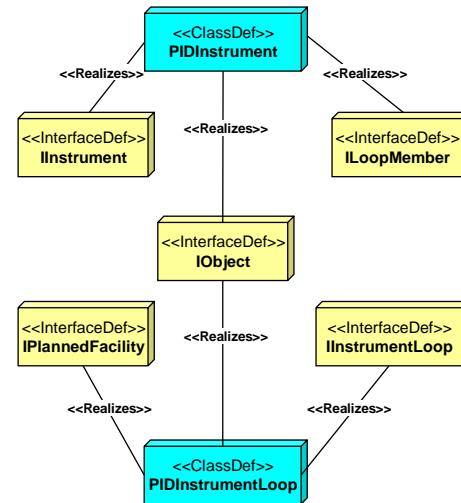
An **Interface Definition** (InterfaceDef) is the most important concept in the meta schema. An interface definition is a named collection of property definitions. Interface definitions expose the property definitions for class definitions. Every interface definition is realized by one or more class definitions. By sharing specific interface definitions, class definitions can also share property definitions, but not the data associated with the properties.

Interface Definitions

An interface definition (InterfaceDef) represents a "role" for a class definition.

Different class definitions can share the same InterfaceDefs, and therefore, the same roles.

For example, all ClassDefs in the schema realize the IObject interface.



© 2005. Intergraph Corp.
All Rights Reserved.

Just as in the real world, the role defines both the properties and relationships of an object. Some interface definitions are defined to carry properties. Some are defined to carry relationships. Others may be defined merely to indicate a role.

Note: SPF uses InterfaceDefs to define what methods are allowed on an object and what workflows are appropriate. These special relationships are not available in the Schema Editor, only within SPF SmartPlant Administration.

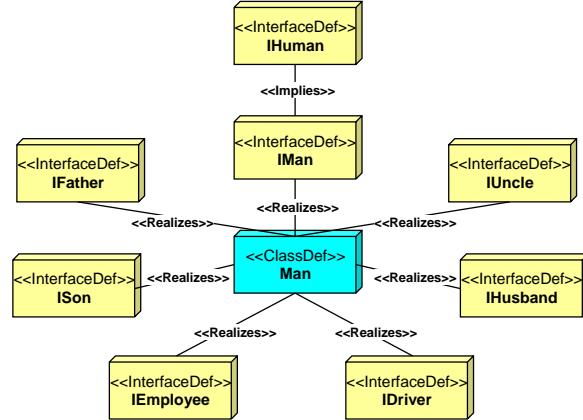
3.6.1 Role Example



Interface Definitions - Role Example

A class definition can offer up many **roles** to the world through its interface definitions.

For example, a "Man" can expose many roles, such as "Father", "Son", "Uncle", "Husband", "Employee", "Driver", and so on.

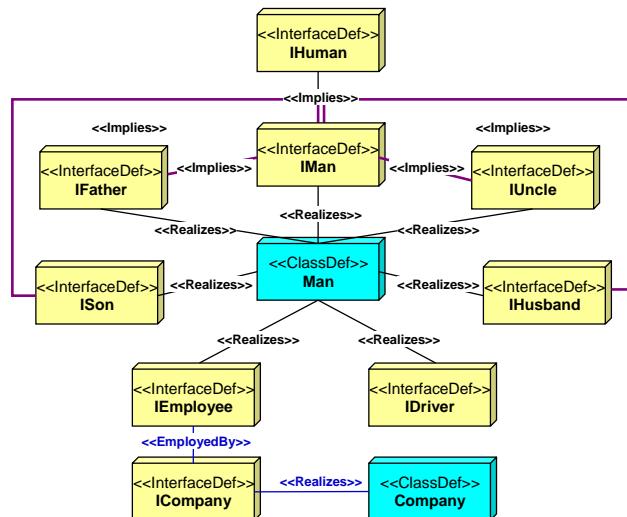


© 2005, Intergraph Corp.
All Rights Reserved.



Interface Definitions - Role Example

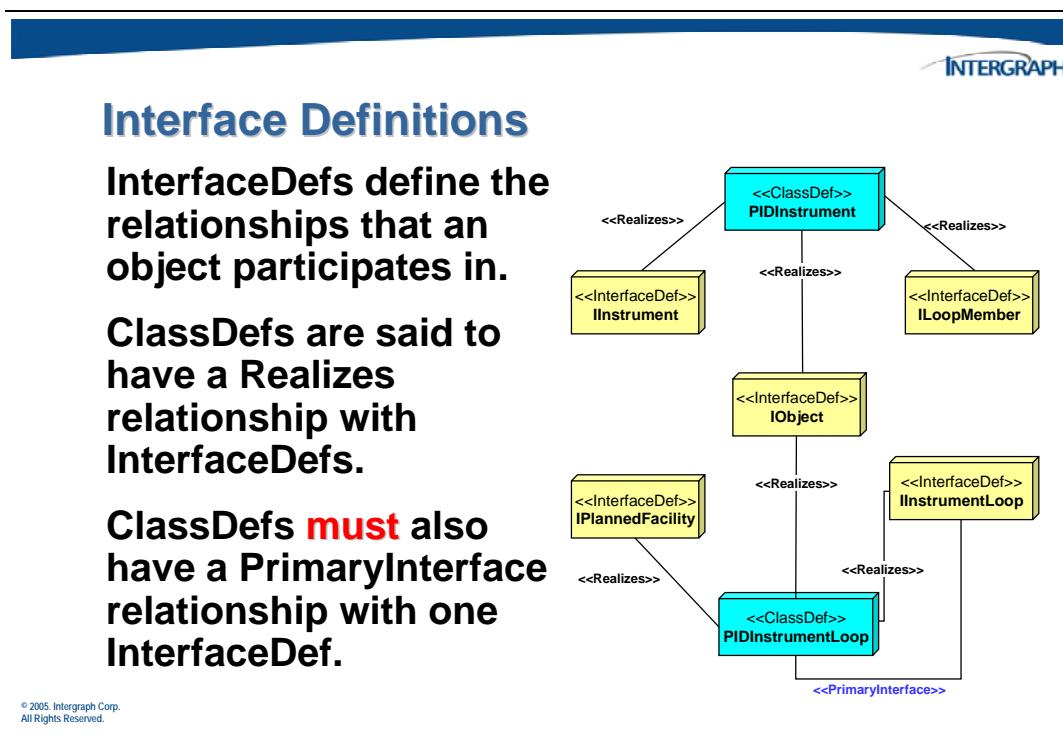
When a company wants to interact with a Man, the company does not care that he is a son or an uncle. The company only cares that the "Man" class exposes the "IEmployee" role for the company to interact with.



© 2005, Intergraph Corp.
All Rights Reserved.

3.6.2 Interface Definitions and Relationships

There are two relationships defined between ClassDef and InterfaceDef. The first (and most common) is the “**Realizes**” relationship. We say that a ClassDef *Realizes* an InterfaceDef. The Realize relationship may be required or optional. If it is required, the Schema Component will insist that any instance of that ClassDef must realize an instance of the required InterfaceDef.



The second relationship defined between ClassDef and InterfaceDef 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 relevant purpose for the **Primary InterfaceDef** is to define the full and complete “roleness” of an object. It defines what an object “is” and everything that is known about it. The *IEquipmentOcc* defines the what a piece of equipment is in the schema and the *IInstrumentOcc* defines what an instrument is. This definition is completed through the use of the **Implies** relationships. When you consider the complete hierarchy of Implies under a Primary InterfaceDef, you know what that object “is”.

This definition of “roleness” is used in several ways in SmartPlant:

1. **Validation** – ClassDefs may not realize any InterfaceDef outside of the implies hierarchy of the primary InterfaceDef.
2. **Sharing** – ClassDefs defined as shared must have the same primary InterfaceDef or must imply the same. PFDProcessEquipment and PIDProcessEquipment have the same primary InterfaceDef and are defined as shared. EQDReactorVessel does not have the same primary InterfaceDef, but it does imply the same; i.e. **IReactorVesselOcc implies IEquipmentOcc**. Therefore, EQDReactorVessel is defined as shared with PFDProcessEquipment and PIDProcessEquipment.
3. **Mapping** – It is possible to map the SmartPlant Schema into the tool Schema by ClassDef. But to do so, 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 ClassDef, you would end up mapping to PFDProceeeEquipment, PIDProcessEquipment, and dozens of EQD* ClassDefs. If you map to IEquipmentOcc, you only do so once and can retrieve any object which realizes that InterfaceDef.

So to summarize primary interface definitions:

1. A primary interface definition is a concept of “roleness” so for example, *IProcessEquipementOcc* outlines all process equipment but *PFDProcessEquipment* does NOT realize all of its hierarchy only the interfaces it needs for the class.
2. ALL realize relationships between a classdef and interfaces are scoped by the primary interface def AND ITS CHILD IMPLIES relations downwards.
3. An implies or primary interface concept is for modelling and mapping “a schema concept” and bears no relation to instantiation in SPF (only realizes does this).
4. An interface must be realized to be used on a form.

One of the interface definitions that identify a role is **IObject**. This is the most common interface in the system and has the properties of *unique identifier*, *a name*, and a *description*. **Every class definition in SmartPlant must realize IObject**.

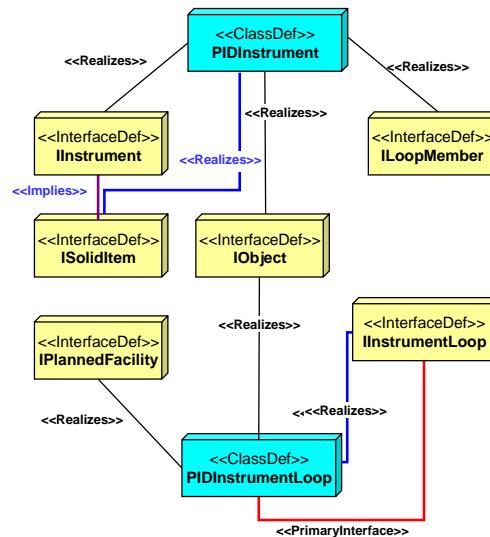
There are no relationships between class definitions, only between interface definitions. That is because *only the interface definitions are exposed to the outside world*. The underlying reason for modeling class definitions and interface definitions instead of class definitions and relationships is 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.

Interface Definitions

InterfaceDefs can imply other InterfaceDefs. If an InterfaceDef implies another InterfaceDef, then any ClassDef that realizes the first InterfaceDef can also realize the implied InterfaceDef.

For example, IInstrument implies the ISolidItem. Therefore, any ClassDef, such as PIDInstrument, that realizes IInstrument must also realize ISolidItem.



© 2005, Intergraph Corp.
All Rights Reserved.

There is a relationship defined between InterfaceDefs called “**Implies**”. If an InterfaceDef such as *IFather* implies another interface such as *IMale*, then any object that has the *IFather* interface must also have the *IMale* interface. The Schema Component enforces these rules.

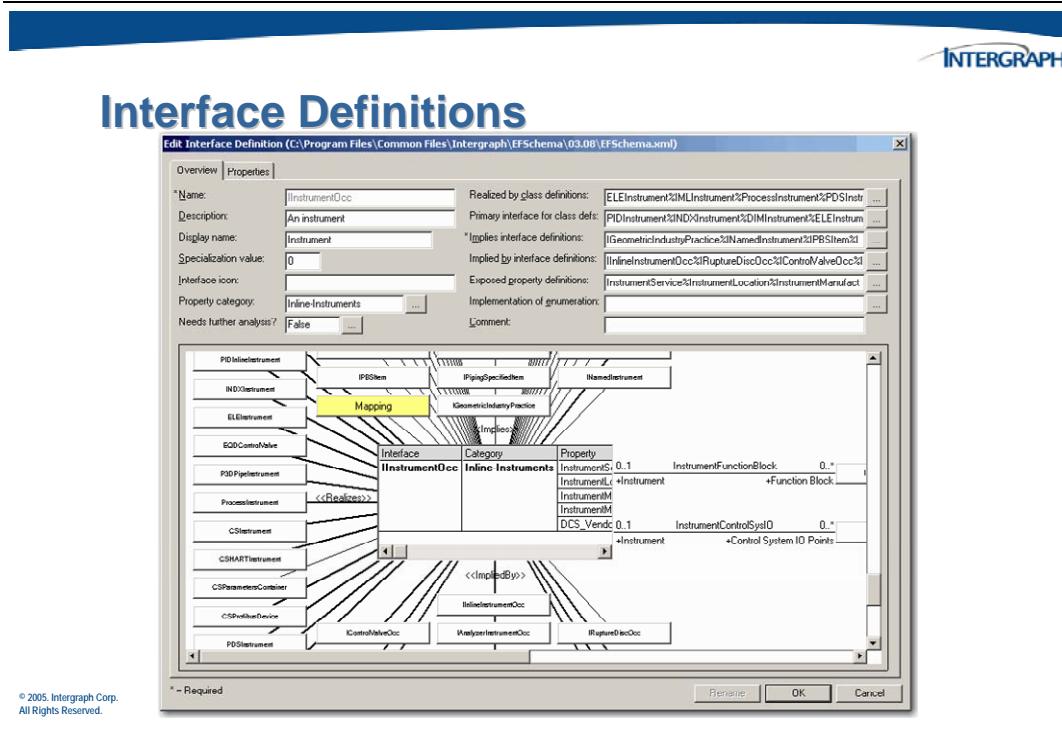
The set of interface definitions directly or indirectly implied by an interface definition defines the “role” for that interface definition. The Implies relationships between InterfaceDefs, also referred to as implied interfaces, also defines the schema hierarchy, which allows you to navigate through the schema almost endlessly.

In the schema, every interface definition, except for *IObject*, should imply the *IObject* interface definition.

3.6.3 Properties of an Interface Definition

Interface definitions have the following properties:

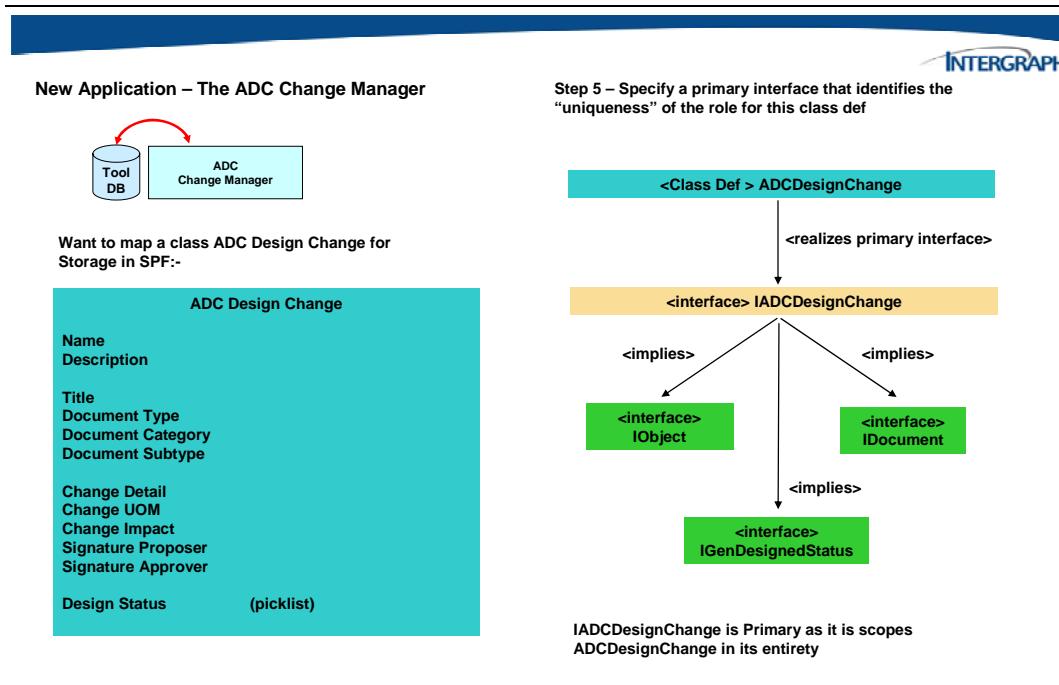
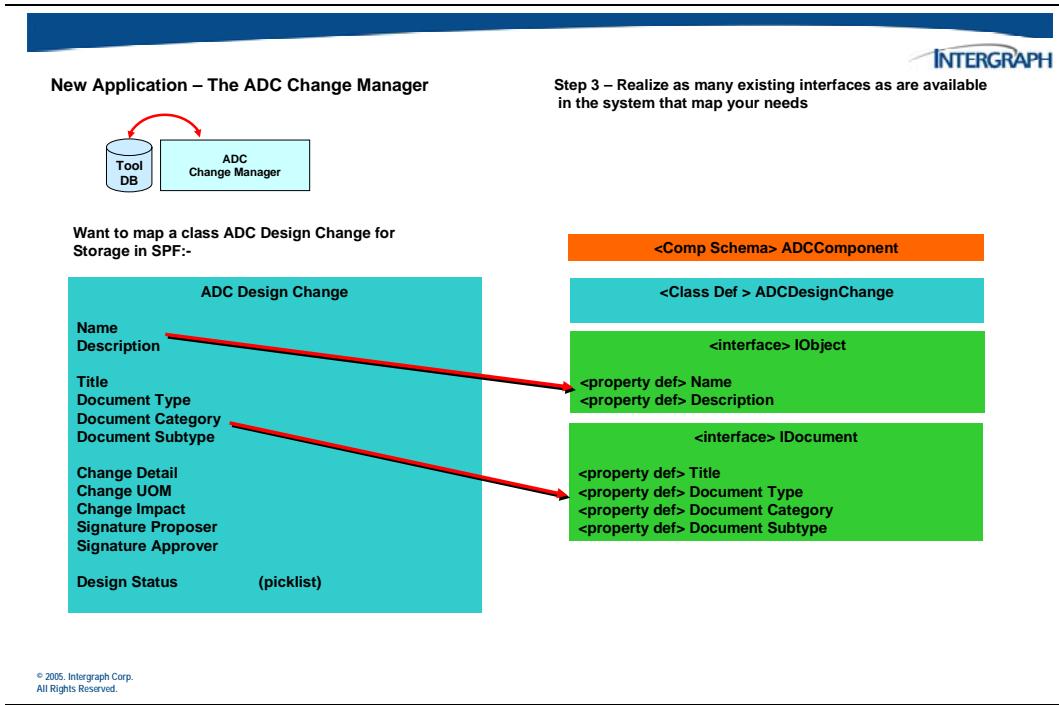
- Name** – Specifies the name of the interface definition.
- Description** – Specifies a description for the interface definition.
- Display name** – Specifies the name that you want the user interface to use when displaying the InterfaceDef.



- Specialization value** – Determines the extent to which this interface identifies the object. The higher the number, the more specific the interface is for defining the object. The primary interface for a class definition should have the highest specialization value. An interface definition should always have a higher specialization value than any of the interface definitions that it implies. However, this is not currently checked during validation.
- Interface icon** – Specifies the icon that you want to use to represent this InterfaceDef in the Schema Editor user interface.
- Property category** – Defines the property category for properties exposed by this interface. The property category helps organize properties in the **Properties** window in SmartPlant Foundation and SmartPlant P&ID.
- Realized by class definitions** – Lists the class definitions that realize this InterfaceDef.

- ❑ **Primary interface for class defs** – Lists the class definitions for which this InterfaceDef is the primary interface. Primary interfaces should imply everything that is known about a particular class definition.
- ❑ **Implies interface definitions** – Lists all the InterfaceDefs that this InterfaceDef implies. All class definitions that realize this InterfaceDef must also realize all the InterfaceDefs this InterfaceDef implies.
- ❑ **Implied by interface definitions** – Lists all other InterfaceDefs that imply this one.
- ❑ **Exposed property definitions** – Lists all property definitions that this InterfaceDef exposes.
- ❑ **Implementation of enumeration** - Defines a relationship between an enumerated entry that represents a classification hierarchy and the primary interface of the objects the enumerated entry classifies. For example, a primary interface, such as IPIDDrawing, should have a relationship with the P&ID enumerated entry from the DocCategories enumerated list.

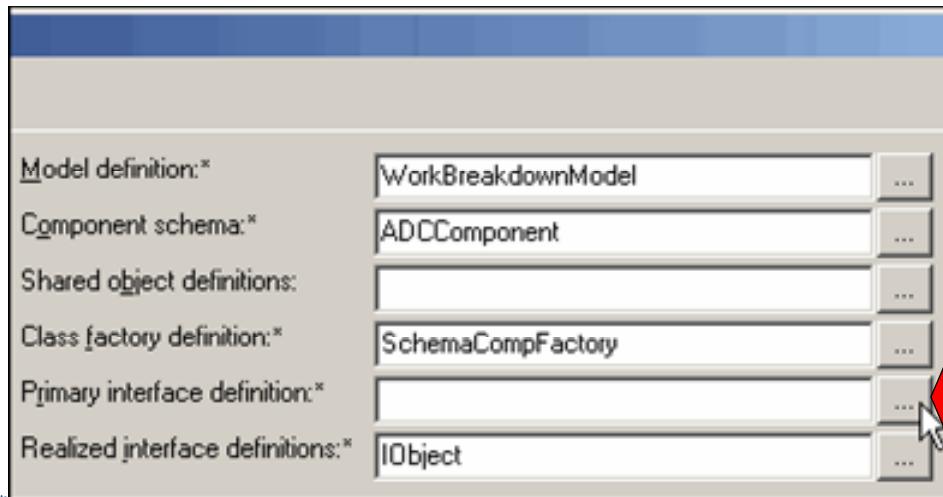
3.7 Interactive Activity – Creating Interface Definitions





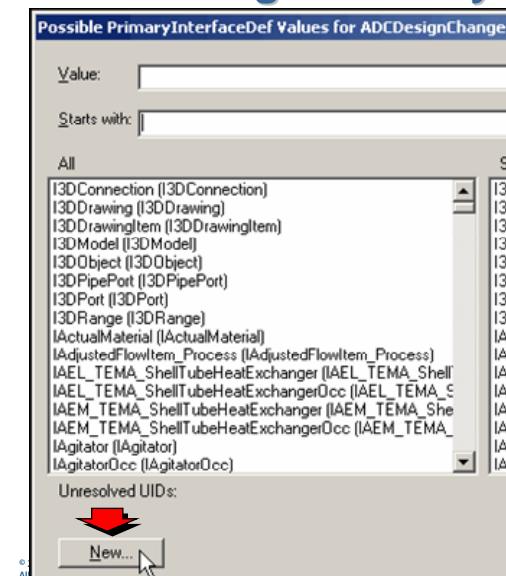
Creating a Primary Interface Definition

- Choose the *Primary interface definition* browse (...) button.



Creating a Primary Interface Definition

- Select the **New** button from the *Possible PrimaryInterfaceDef Values for ADCDesignChange* dialog box.





Creating a Primary Interface Definition

- ❑ Enter the information to create the new InterfaceDef

New Interface Definition (adc.xml)

Overview	Properties
Name: [*]	IADCDesignChange
Description:	Primary Interface for ADCDesignChange
Display name:	
Specialization value: [*]	0
Interface icon:	
Property category:	
Needs further analysis?	False

© 2005, Intergraph Corp.
All Rights Reserved.

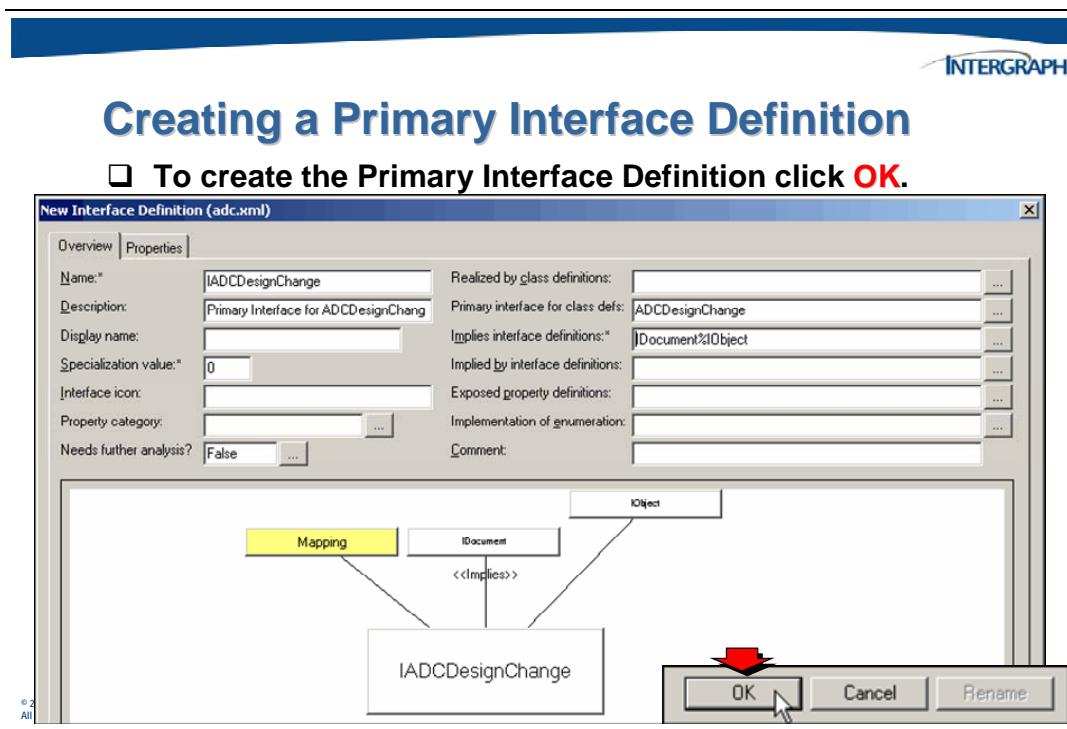
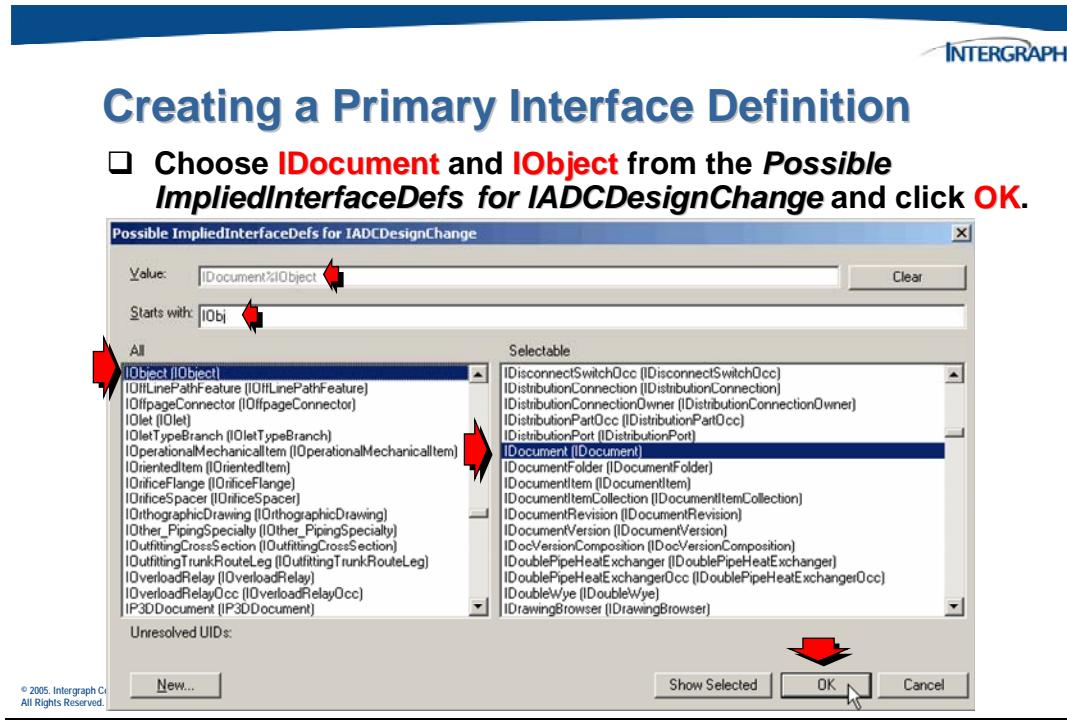


Creating a Primary Interface Definition

- ❑ Choose the *Implies interface definitions* browse (...) button

Realized by class definitions:	
Primary interface for class defs:	ADCDesignChange
Implies interface definitions: [*]	
Implied by interface definitions:	
Exposed property definitions:	
Implementation of enumeration:	
Comment:	

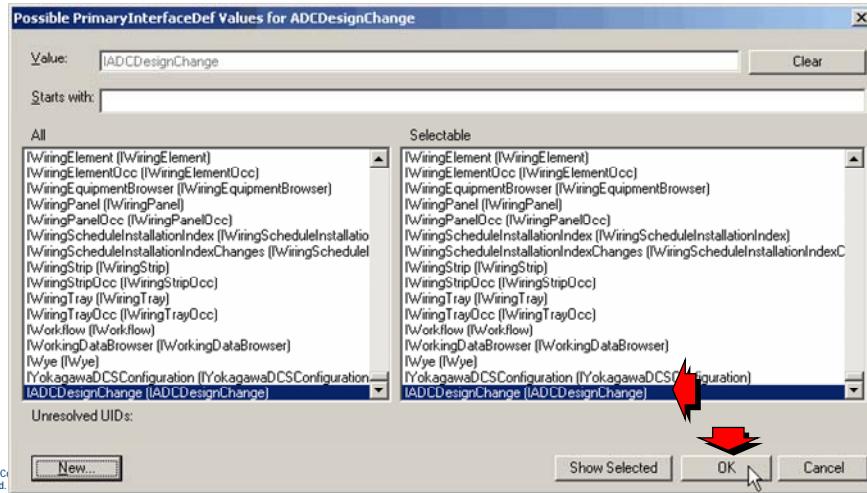
© 2005, Intergraph Corp.
All Rights Reserved.





Creating a Primary Interface Definition

- From the **Possible PrimaryInterfaceDef Values for ADCDesignChange** dialog box click **OK**.

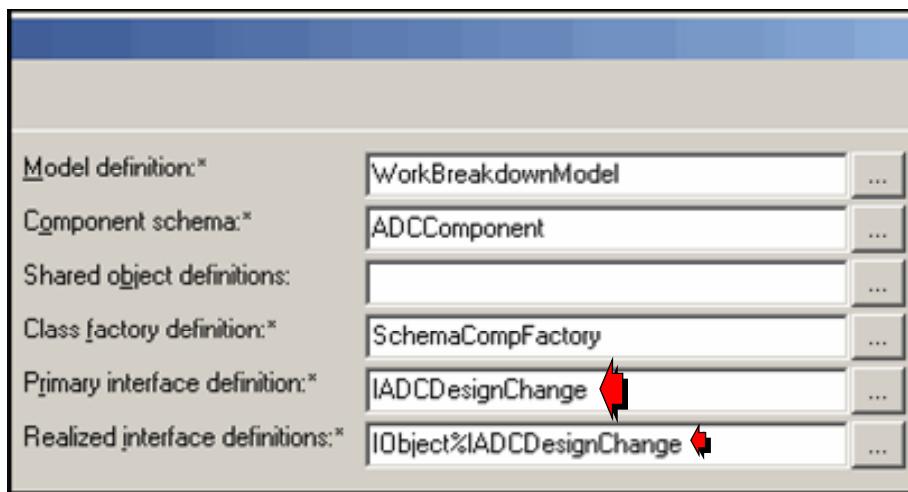


Note that the new ClassDef also **Realizes** the new *Primary Interface*.



Creating a Primary Interface Definition

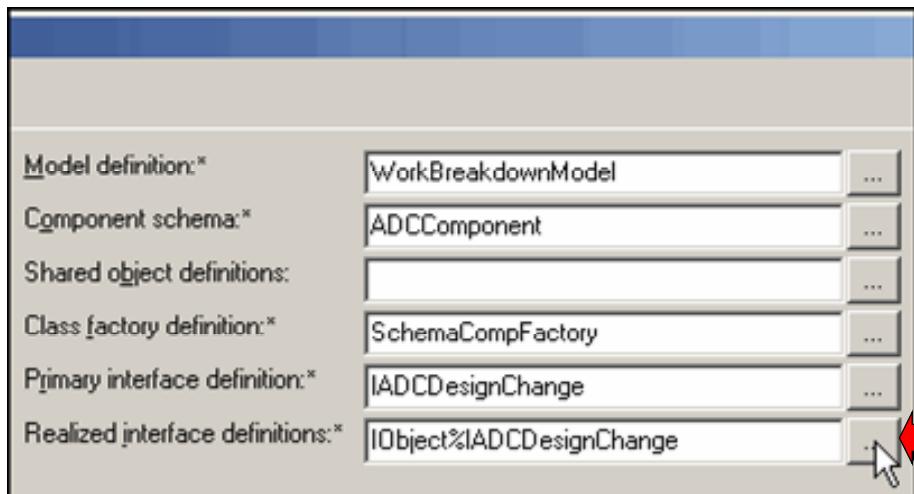
The Primary Interface Definition has been set.





Creating a Class Definition

- Choose the **Realized interface definitions** browse (...) button.

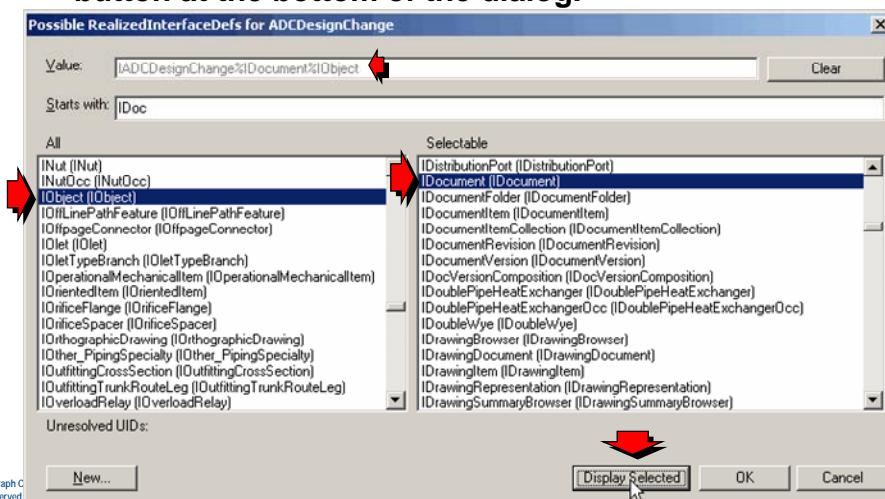


© 2005, Intergraph Corp.
All Rights Reserved.

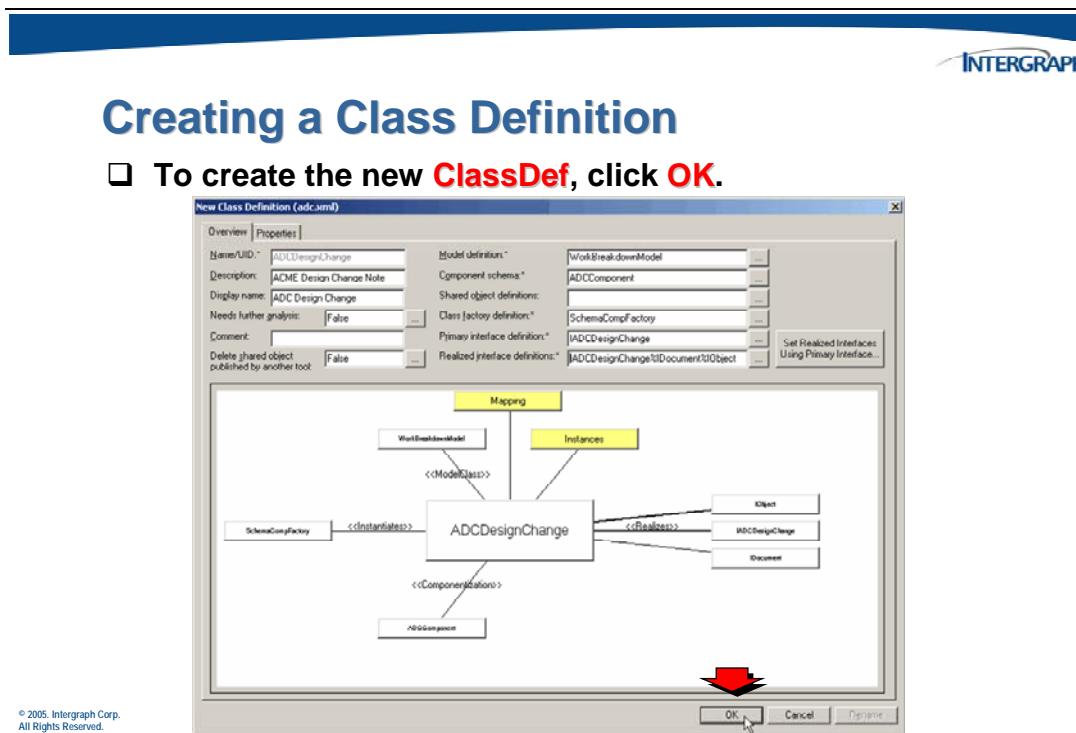
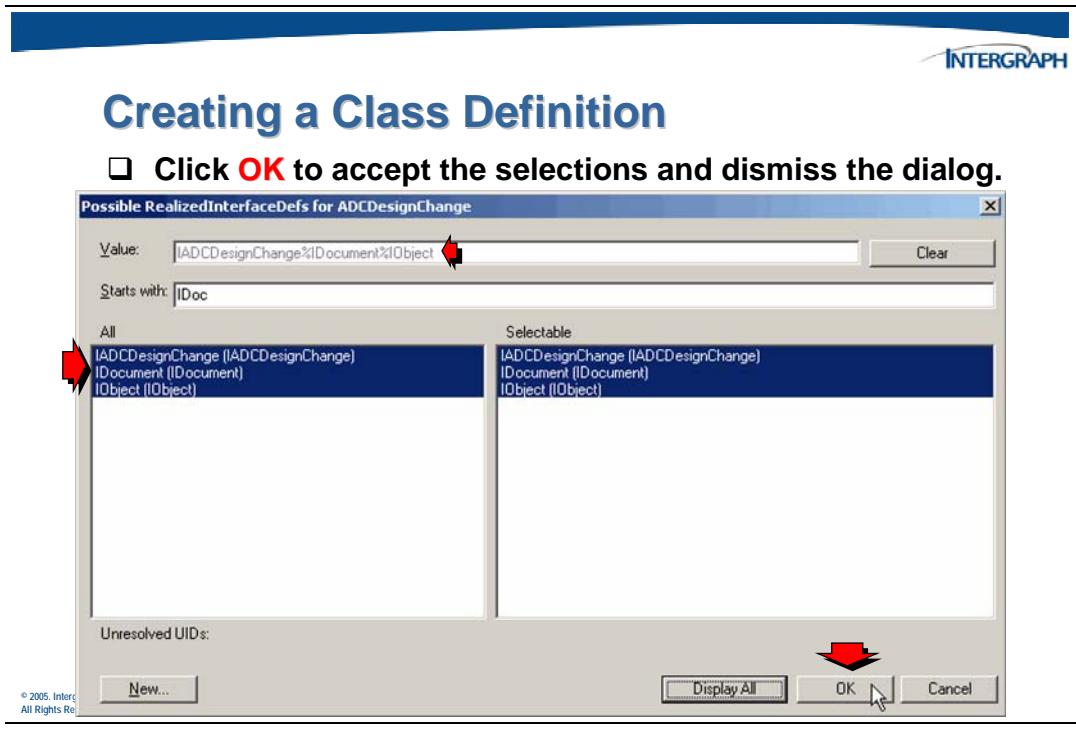


Creating a Class Definition

- To see the selected entries, click the **Display Selected** button at the bottom of the dialog.



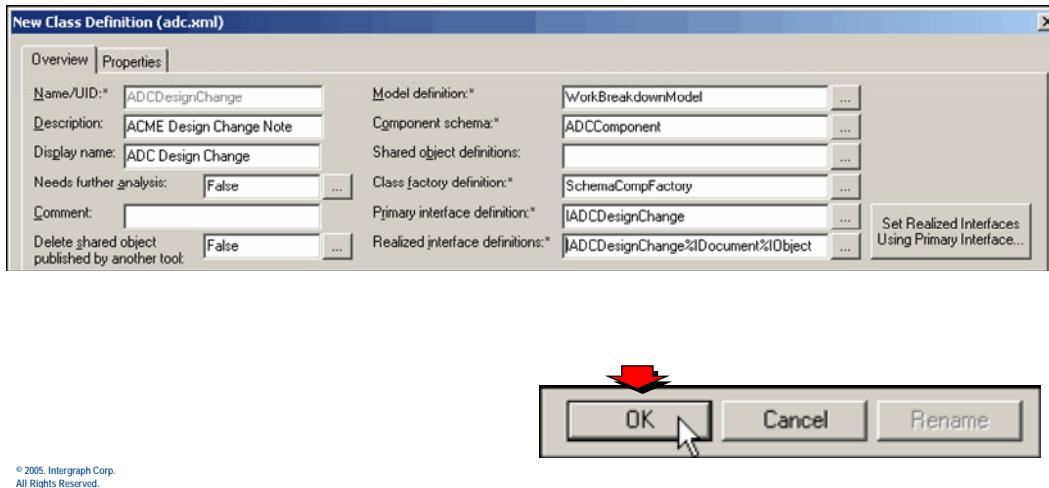
© 2005, Intergraph C
All Rights Reserved





Creating a Class Definition

Zoomed in view of the **New Class Definition** dialog box.

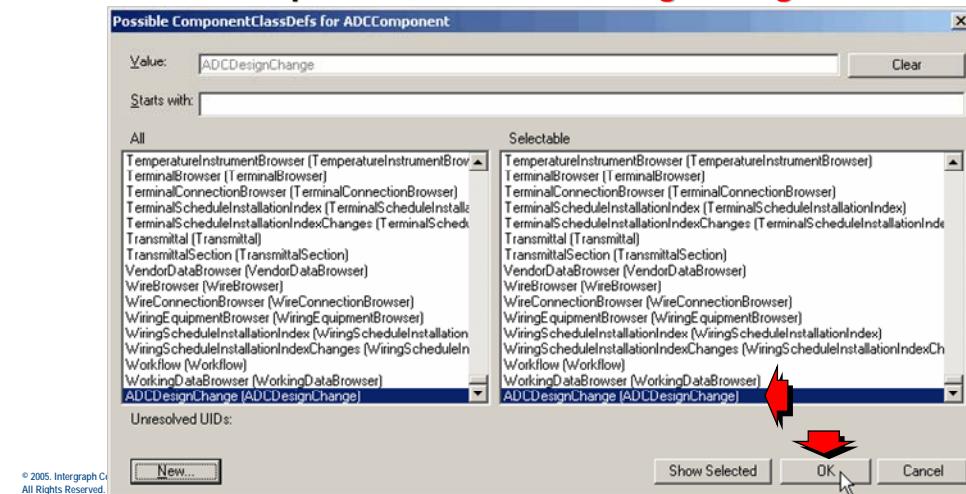


© 2005, Intergraph Corp.
All Rights Reserved.



Creating a Component Schema

- From the **Possible ComponentClassDefs for ADCComponent** choose **ADCDesignChange**.

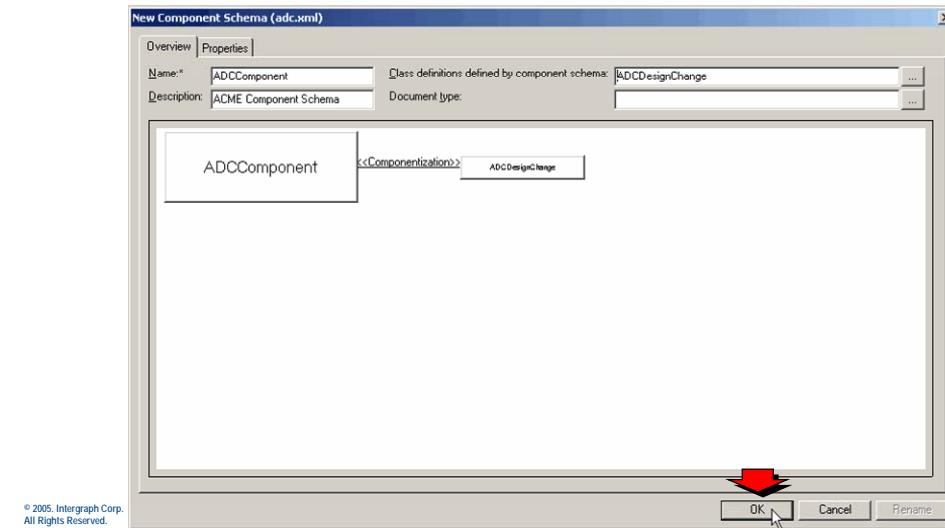


© 2005, Intergraph C
All Rights Reserved.



Creating a Component Schema

- To create the new **Component Schema**, click **OK**.



Creating a Component Schema

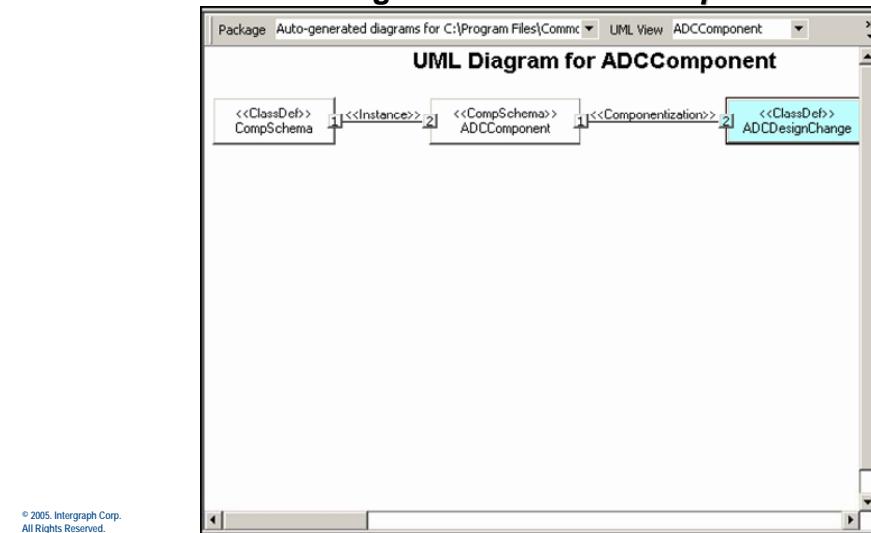
- To create a default UML view for the **ADCComponent**, click **Yes**.





Creating a Component Schema

A default UML diagram for the **ADCComponent** is created.



ClassDef/InterfaceDef Properties

- Click the **View > Schema (Active)...** menu command to view the InterfaceDef properties for a ClassDef.

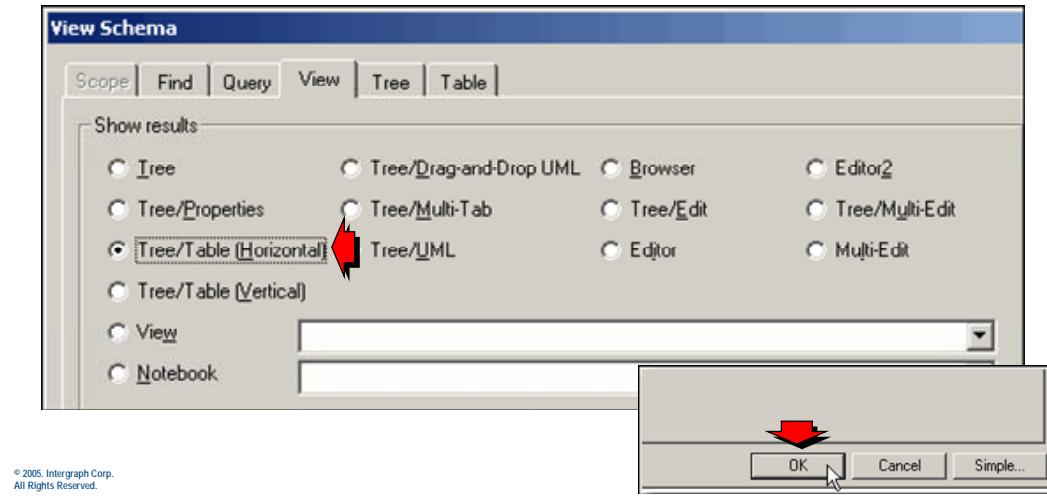
The screenshot shows the "Schema Editor - [Schema (All opened files) Editor UML (adc.xml)]" window. The "View" menu is open, and the "Schema (Active) ..." option is highlighted with a red arrow. To the right of the menu, a UML diagram titled "UML Diagram for ADCComponent" is visible, showing the same four nodes and connections as the previous diagram.

© 2005, Intergraph
All Rights Reserved.



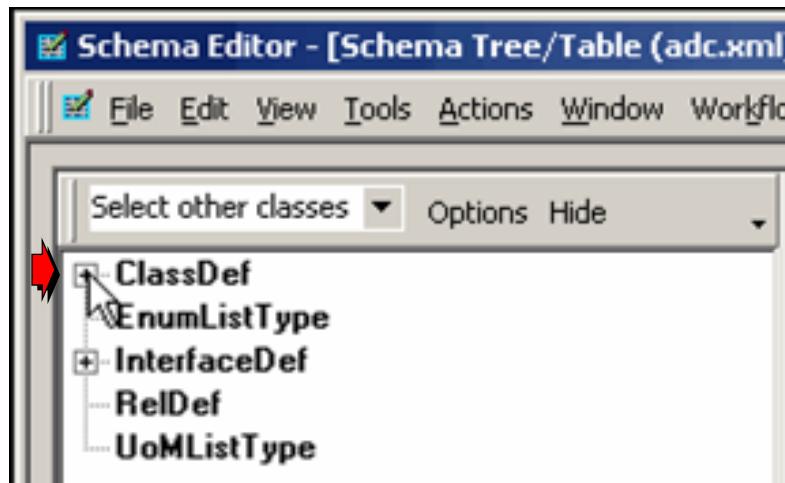
ClassDef/InterfaceDef Properties

- ❑ In the **View Schema** dialog box, select the **Tree/Table (Horizontal)** option and click **OK**.



ClassDef/InterfaceDef Properties

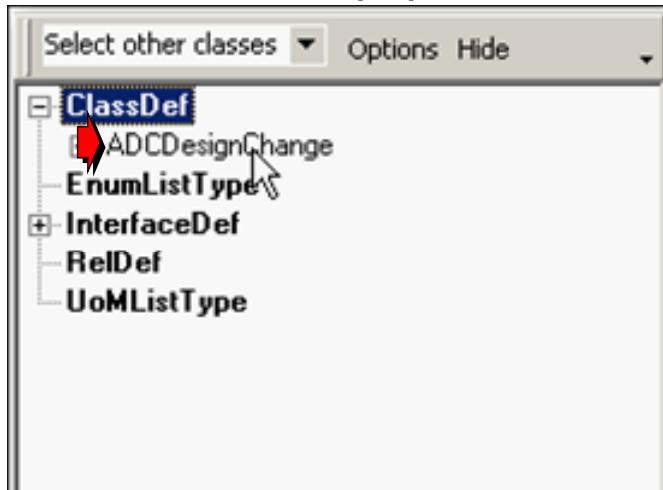
- ❑ Expand the **ClassDef** objects to see a listing of all ClassDefs.





ClassDef/InterfaceDef Properties

- Click on the **ADCDesignChange** object to see a listing of the realized interfaces and properties.



© 2005, Intergraph Corp.
All Rights Reserved.



ClassDef/InterfaceDef Properties

The **ADCDesignChange** properties are displayed in the right pane in a horizontal table.

The screenshot shows the 'Schema Editor - [Schema Tree/Table (adc.xml)]' window. The left pane displays the schema tree with the 'ADCDesignChange' node selected. The right pane contains a table showing the properties of the 'ADCDesignChange' class. The table has columns for Class, Interface, Property, and Type. There are two rows for the 'ADCDesignChange' class:

Class	Interface	Property	Type
ADCDesignChange	IObject	UID	string128
		Name	
		Description	string
	IADCDesignChange	DocCategory	DocCategories
		DocSubtype	DocSubtypes
		DocTitle	string
		DocType	DocTypes

© 2005, Intergraph Corp.
All Rights Reserved.



ClassDef/InterfaceDef Properties

- Click the X to **Close** the Schema Tree/Table window.

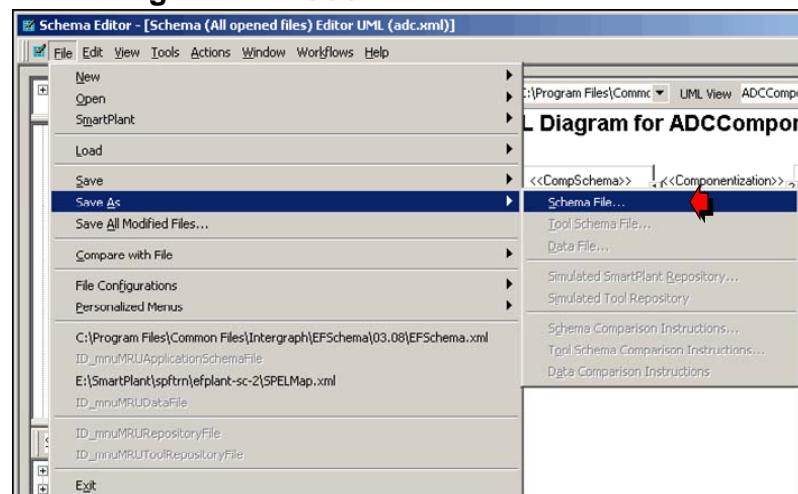
Class	Interface	Property	Type
ADCDesignChange	IObject	UID	string128
		Name	
		Description	string
	IADCDesignChange	DocCategory	DocCategories
		DocSubtype	DocSubtypes
		DocTitle	string
		DocType	DocTypes

© 2005, Intergraph Corp.
All Rights Reserved.



Saving Schema Changes

- On the menu, click **File > Save As > Schema File...** to save the changes to the **adc.xml**.

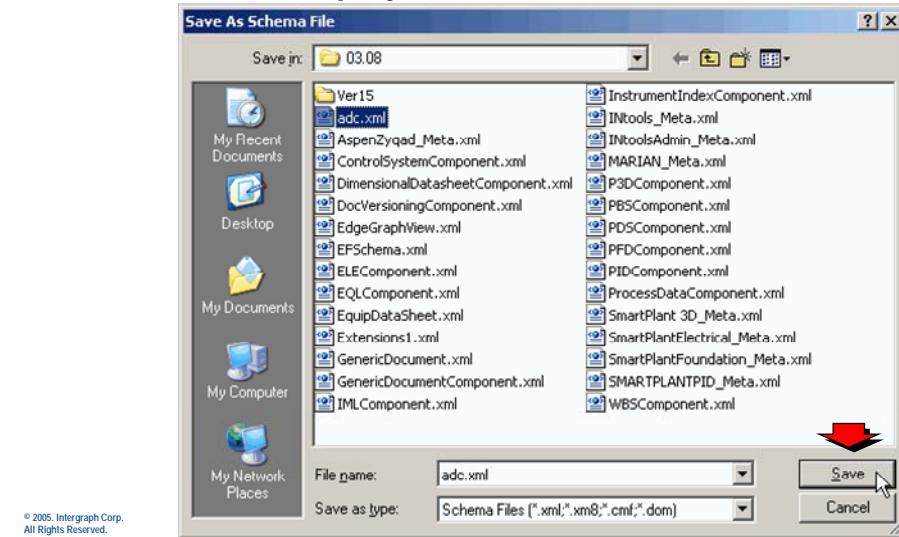


© 2005, Intergraph Corp.
All Rights Reserved.



Saving Schema Changes

- Choose the project schema, adc.xml, and select **Save**.

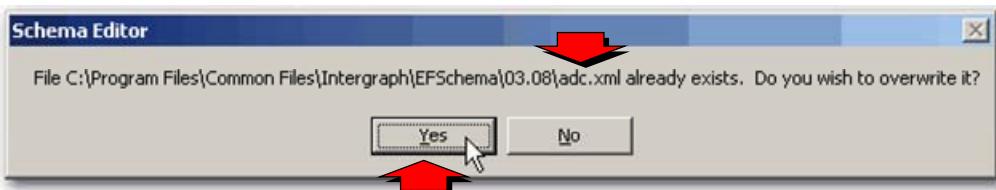


© 2005, Intergraph Corp.
All Rights Reserved.



Saving Schema Changes

- Click **Yes** to save the changes to the project schema.



© 2005, Intergraph Corp.
All Rights Reserved.

3.8 Activity 1 – Reviewing Schema Concepts

The goal of this activity is to review what you have learned about the schema.

1. Fill in the blank:

In the schema, each _____ represents a role.

2. Relationships exist between

- a. Class Definitions
- b. Interface Definitions
- c. Property Definitions
- d. Property Types

3. Fill in the blank:

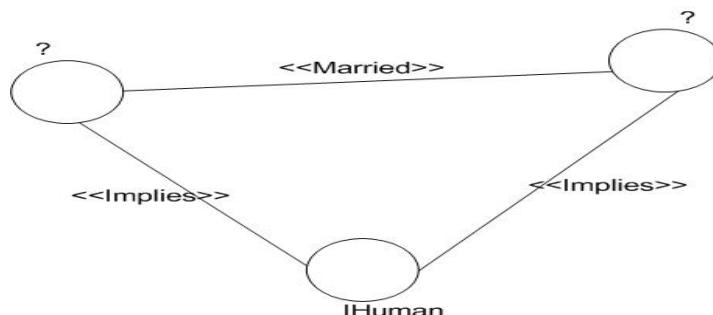
A _____ is a named description of a set of objects, which can represent physical things such as pumps, or conceptual things, such as projects.

4. Fill in the blank:

Cardinalities are found at each end of a _____.

5. Suggest Interface names for the two Interfaces represent by “?” below and apply direction to the “implies” relationships:
-

Note: Interfaces are represented as circles and Relationships are represented with <<Relationship Name>>.



6. Suggest a name for the relationship represented by the “?” below:



Fill in the blanks:

7. _____ Realize _____.
8. _____ Expose _____.
9. _____ Imply _____.
10. _____ are Scoped by _____.
11. Every Class Definition must Realize the _____ Interface.
12. Every Interface Definition must Imply the _____ Interface.

3.9 Activity 1 – Answer Key

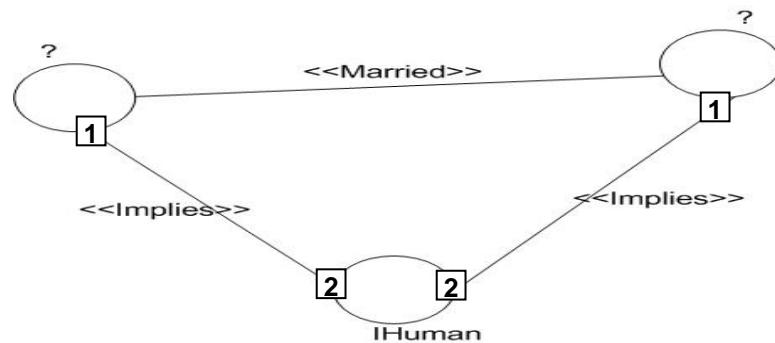
1. In the schema, each *interface definition* represents a role.

2. Relationships exist between
 - a. Class Definitions
 - b. *Interface Definitions*
 - c. Property Definitions
 - d. Property Types

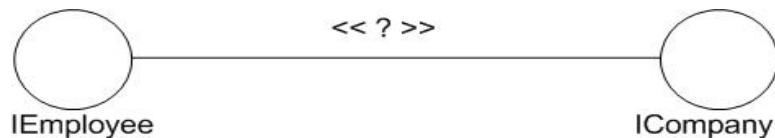
3. A *class definition* is a named description of a set of objects, which can represent physical things such as pumps, or conceptual things, such as projects.

4. Cardinalities are found at each end of a *relationship definition*.

5. Suggest Interface names for the two Interfaces represent by “?” below:
 - *IHusband*
 - *IWife*



6. Suggest a name for the relationship represented by the “?” below: *EmployedBy*



Fill in the blanks:

7. Class definitions Realize interface definitions.
8. Interface definitions Expose property definitions.
9. Interface definitions Imply other interface definitions.
10. Property definitions are Scoped by property types.
11. Every Class Definition must Realize the IObject Interface.
12. Every Interface Definition must Imply the IObject Interface.

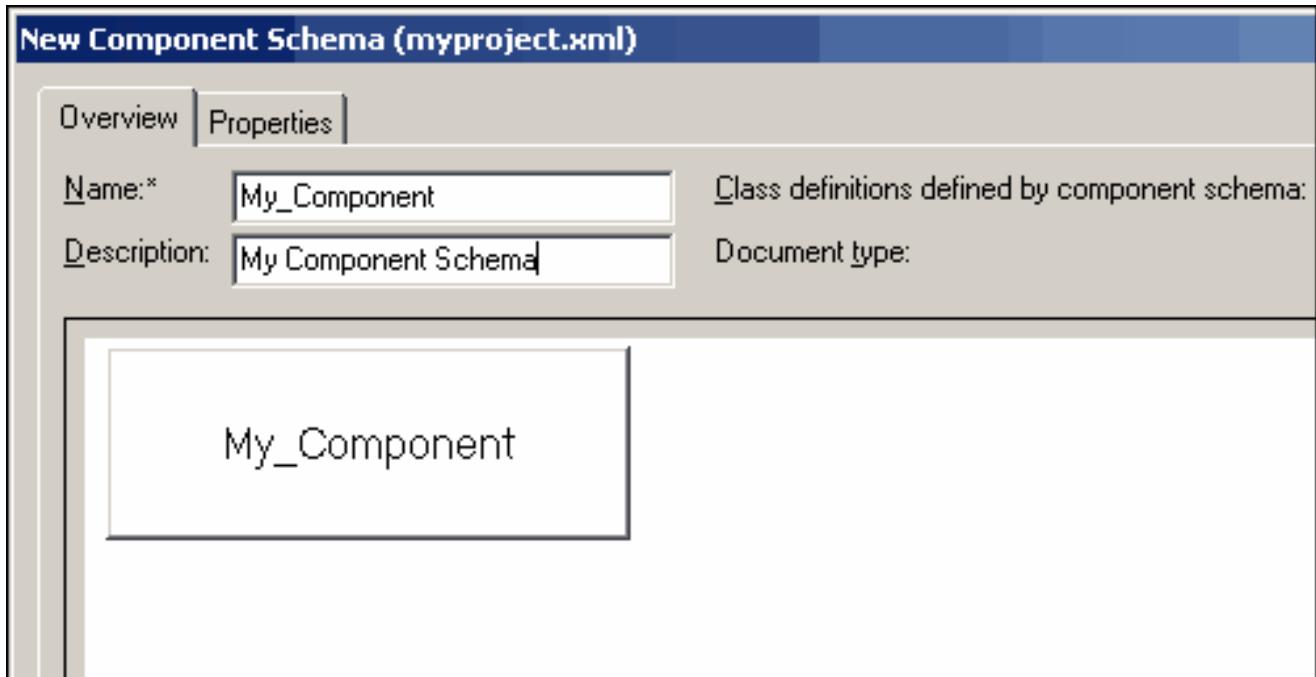
3.10 Activity 2 – Creating Objects and Relationships with Schema Editor

In the Schema Editor, you can modify the schema and tool schemas by creating new objects, interfaces, relationships, enumerated lists, and so on. You can also modify data files in the Schema Editor. However, you cannot make changes to the meta schema using the Schema Editor.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click *Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor* to start the *Schema Editor*.
3. Create a new project schema template file that will contain all of the custom modeling statements for this lab exercise.
 - Open the default configuration file, **EFSchema.cfg**.
 - In the *Workflows* dialog box, click the *New* button above the *Another Schema File* button.
 - When the *New Another Schema File* dialog box appears, enter the name **myproject.xml** and click **Save**.
 - When the *Set Container Dependencies* dialog box displays, select the master **EFSchema.xml** file, **EdgeGraphView.xml** file and click **OK**
4. Set the active schema file to be modified during the modeling edit sessions.
 - Click the *Another Schema File* button in the *Workflows* dialog box.
 - Select the *Set Active Schema File...* command
 - When the *Set Active File* dialog box displays, select the **myproject.xml** file
 - Click **OK** to dismiss the *Set Active File* dialog.
5. Edit the configuration to be used for both the master schema and the project schema during the modeling edit sessions.
 - Click the *Another Schema File* button in the *Workflows* dialog box.
 - Select the *Edit Configuration...* command

- When the *Edit Configuration* dialog box displays, make the **EFSchema.xml** file not editable by clicking in the **Editable?** column and toggling the setting to **No**
 - Make the **EdgeGraphView.xml** file not editable by clicking in the **Editable?** column and toggling the setting to **No**
 - Make the **myproject.xml** file editable by clicking in the **Editable?** column and toggling the setting to **Yes**
 - Verify that the *Active schema file* is **myproject.xml**
 - Close the *Edit Configuration* window (click **X**).
6. Save the configuration so that it can be used in future editing sessions.
- In the *Workflows* dialog box, click the **File Configurations** button in the lower right corner of the application window.
 - When the *Save As Configuration File* dialog box displays, enter the name **myproject.cfg** and click **Save**.
7. Exit the Schema Editor and save your work.
- Select the **File>Exit** command from the menu
 - When the *Schema Editor* dialog displays, verify that the **myproject.xml** file is specified and click **Yes** to save the project schema.
8. Open Windows Explorer and locate the **myproject.cfg** configuration file in the C:\Program Files\Common Files\Intergraph\EFSchema\03.08 folder.
9. Edit this config file using WordPad and remove the reference to **Extensions1.xml**.
- Highlight this line in the file and use the right mouse button **Cut** command to remove it from the file.
 - Save this config file and exit from WordPad.
- To create a new class definition in the schema, do the following:
10. Start the *Schema Editor* once again and open the configuration that was saved in the steps above in order to open both the master schema (EFSchema.xml) and the project schema (myproject.xml).

- Click the **File Configurations** button in the lower right corner of the application window.
 - When the *Open Configuration File* dialog box displays, select the name **myproject.cfg** and click **Open**
11. Click the **View** button beside the *Another Schema File* button.
12. On the *View Schema* dialog box select the **View** tab and click **Editor2**.
13. Click **OK**.
14. Drag **CompSchema** from the **Create** tree to the UML view.



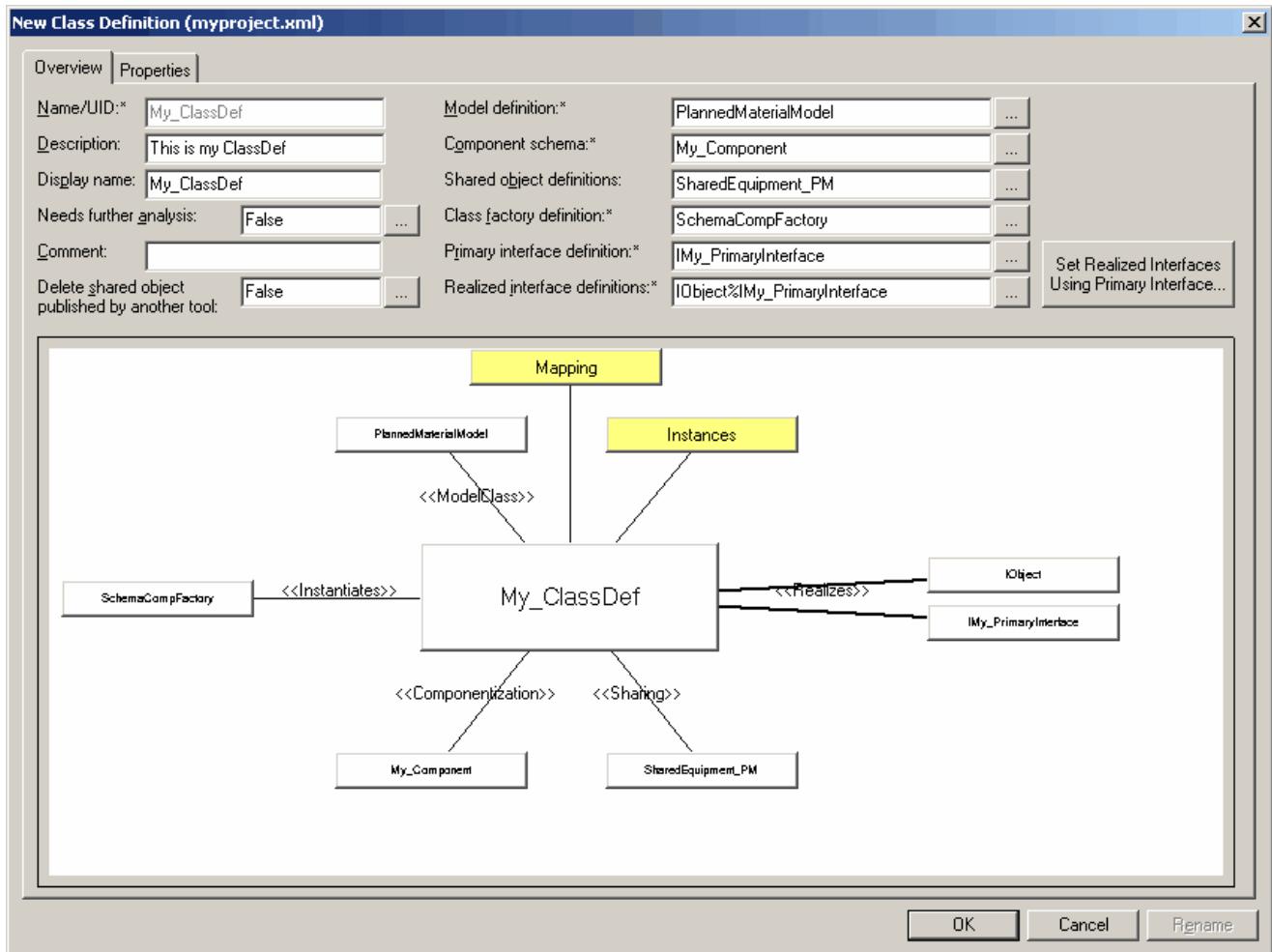
Note: You can also double-click the **CompSchema** node in the **Create** tree to create a new class definition.

11. In the *New Component Schema* dialog box, type **My_Component** in the **Name** box.
12. Type **My Component Schema** in the **Description** box.

13. Beside the **Class definitions defined by component schema** box, click  and select the **New** button from the *Possible ComponentClassDefs for My_Component* dialog box.

14. In the *New Class Definition* dialog box, type **My_ClassDef** in the **Name** box.

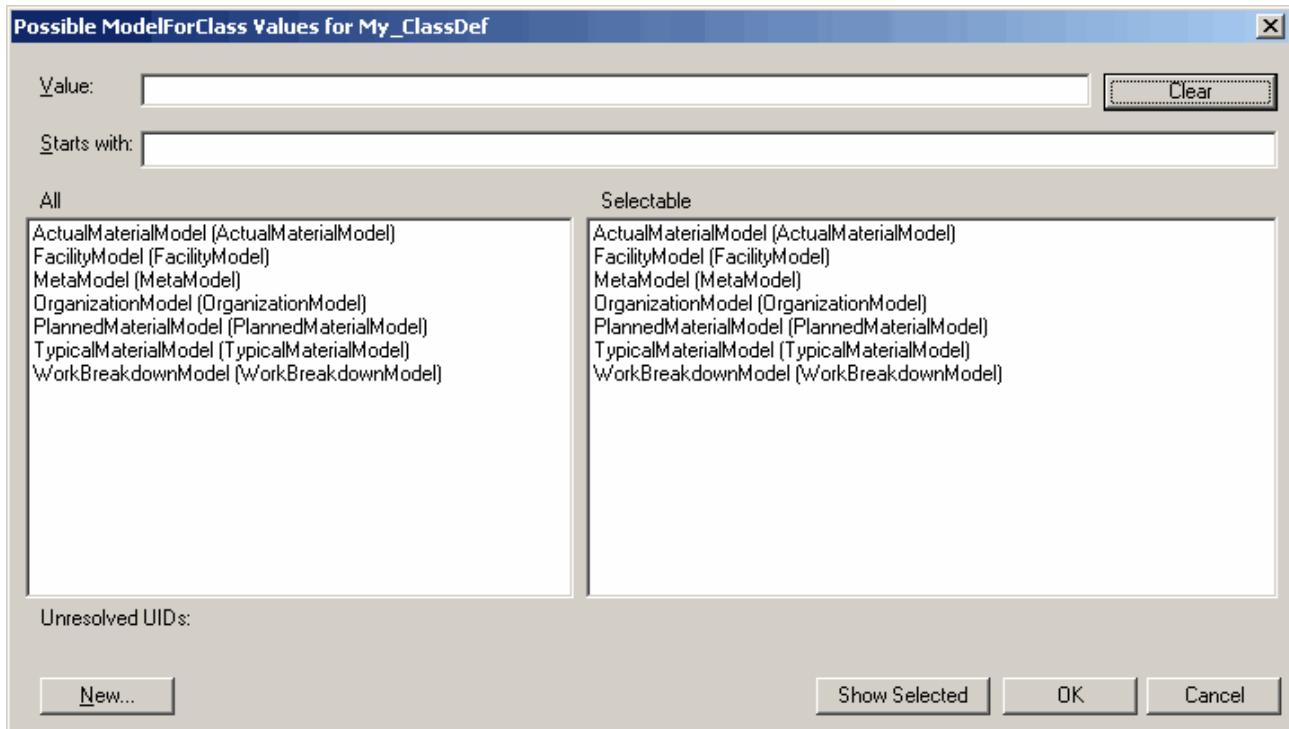
15. Type **This is my ClassDef** in the **Description** box.



16. Type **My_ClassDef** in the **Display Name** box.

17. Beside the **Model definition** box, click  and select **PlannedMaterialModel** from the *Possible ModelForClass Values for My_ClassDef* dialog box:

Note: You can type the object names in the text boxes on the right side of the **New Class Definition** dialog box instead of using the  button to display the relationship help dialog box.



18. Beside the **Shared object definitions** box, click , and select the **SharedEquipment_PM**. You can select one or multiple shared objects.

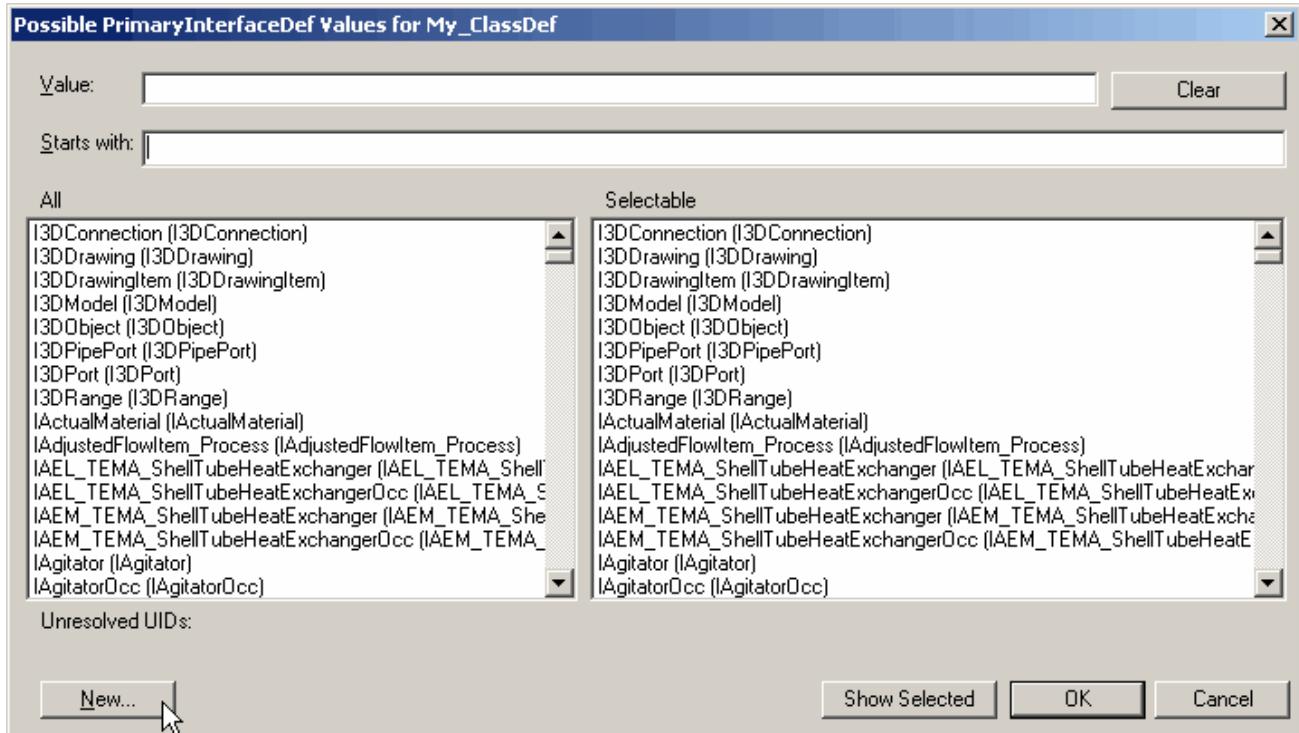
Note: Shared objects must exist in the same model. For example, since you selected **PlannedMaterialModel** as the shared object definition for the new class, all shared objects for this class must also exist in the **PlannedMaterialModel**.

19. Click **Ok** to dismiss the *Possible SharedObjDefs for My_ClassDef* dialog box.

20. Beside the **Class factory definition** box, click , and select **SchemaCompFactory**.

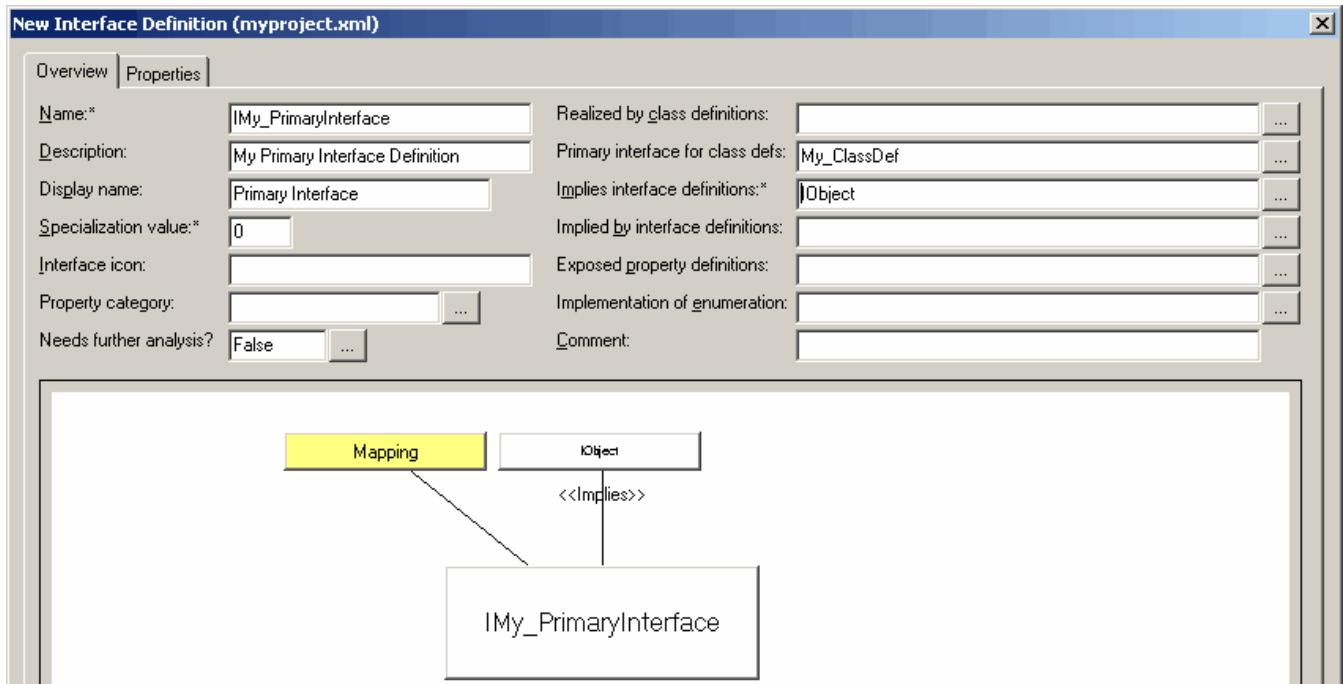
Creating Interfaces and Property Definitions

21. Beside the *Primary interface definition* box, click to display the *Possible Primary interface definition* dialog box for My_ClassDef dialog box.



22. Click **New** (lower left) of the *Possible PrimaryInterfaceDef Values for My_ClassDef* dialog box to create a new interface.

23. Fill in the data as shown below:



24. Click **OK** in the *New Interface Definition* dialog box.

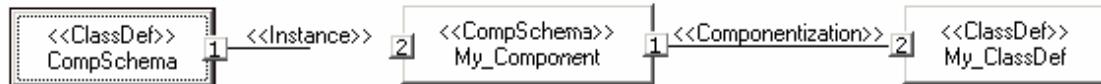
25. Click **OK** in the *Possible PrimaryInterfaceDef Values* dialog box for **My_ClassDef** dialog box to save the new Primary Interface Definition.

Note: As you create relationships to the new class definition, the software creates a default UML diagram for the class definition in the UML view at the bottom of the *New* dialog box. You can modify this UML view the same way you modify any drag-and-drop.

26. Click **OK** in the *New Class Definition* dialog box to save the new Class Definition and the *Possible ComponentClassDefs for My_Component* dialog to finish the operation. Click **OK** in the *New Component Schema* dialog box to save the new Component Schema.

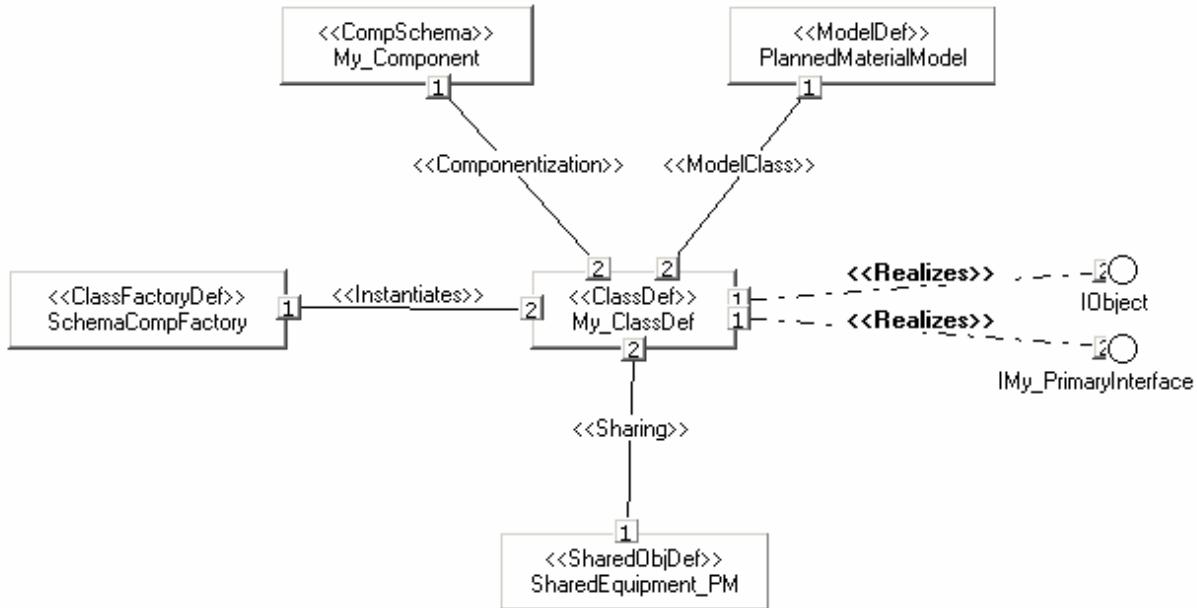
27. A prompt will appear asking you “*Do you want to create default view for My_Component*”. Click **Yes** to create the default view in the UML as shown below:

UML Diagram for My_Component



28. Perform the operation(s) to display the UML view that is shown below:

Class Diagram for My_ClassDef



29. Save your edits to the new schema called “**myproject.xml**” and exit from the schema editor.

30. Once you have closed the Schema Editor, you may take a short break until the other students have finished this activity.

4

C H A P T E R

Creating Properties, Enumerated Lists and Relationships

4. Creating Properties, Enumerated Lists and Relationships

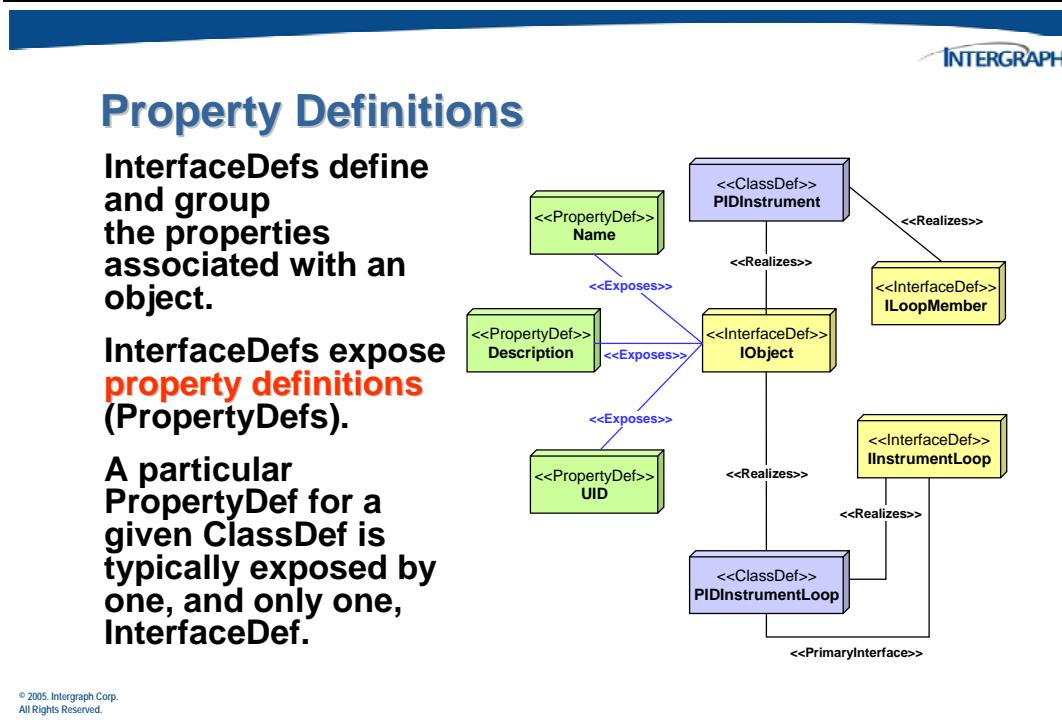
In this chapter, creating attributes or properties will be covered. This will allow characteristics to be added to the interfaces that have been created to define a “role” for the associated class definition.

Also, there may be a requirement to utilize “picklists” from a user interface. This can be accomplished by creating an enumerated list type which contains entries called enum enums.

Finally, the ability to create additional relationships, or relationship definitions, between custom interfaces and existing interfaces will be discussed.

4.1 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.



© 2005, Intergraph Corp.
All Rights Reserved.

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

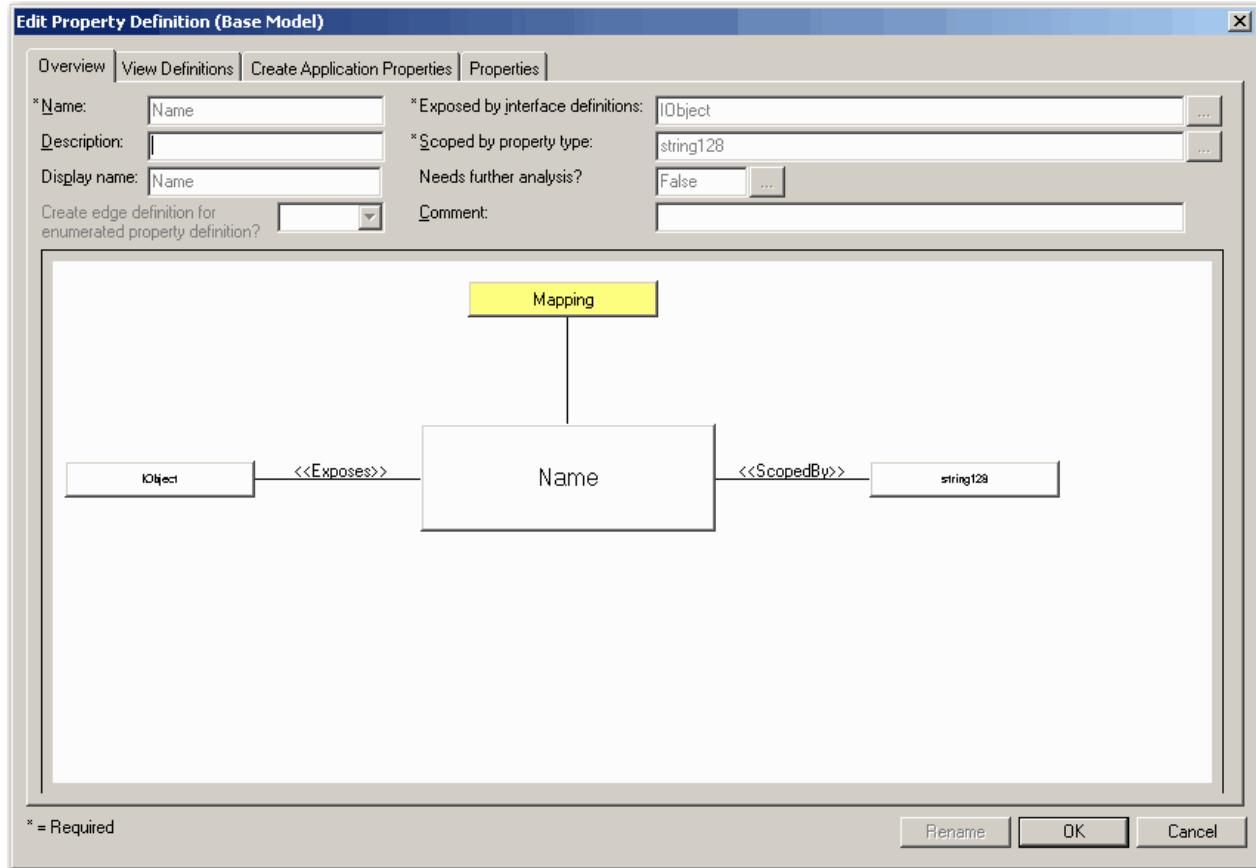
- ❑ **UID** - Unique identifier for the object. This identifier is only required to be unique within the SmartPlant.
- ❑ **Name** - Name of the object
- ❑ **Description** - Description of the object

Some interface definitions exist to hold properties and while other interface definitions exist as merely relationships. Some interface definitions are defined for neither properties nor relationships – they are defined only for a role.

4.1.1 Properties of a Property Definition

Property definitions have the following attributes in the schema:

- Name** – Specifies the name of the PropertyDef.
- Description** – Specifies a description for the PropertyDef.
- Display name** – Specifies the name that you want the user interface to use when displaying the PropertyDef.



- Exposed by interface definitions** – Lists the interface definitions that expose this PropertyDef. PropertyDefs are exposed by one and only one InterfaceDef.
- Scoped by property type** – Specifies the property type that scopes this PropertyDef. 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.

4.2 Property Types

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.

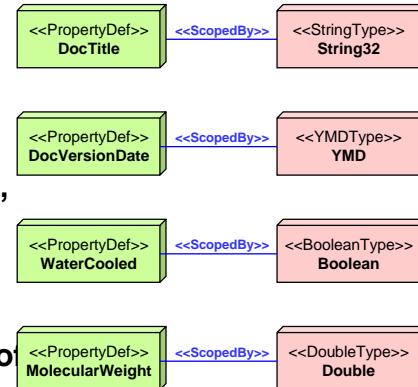


Property Types

Property types define the set of possible values for a **PropertyDef**.

The **ScopedBy** 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.

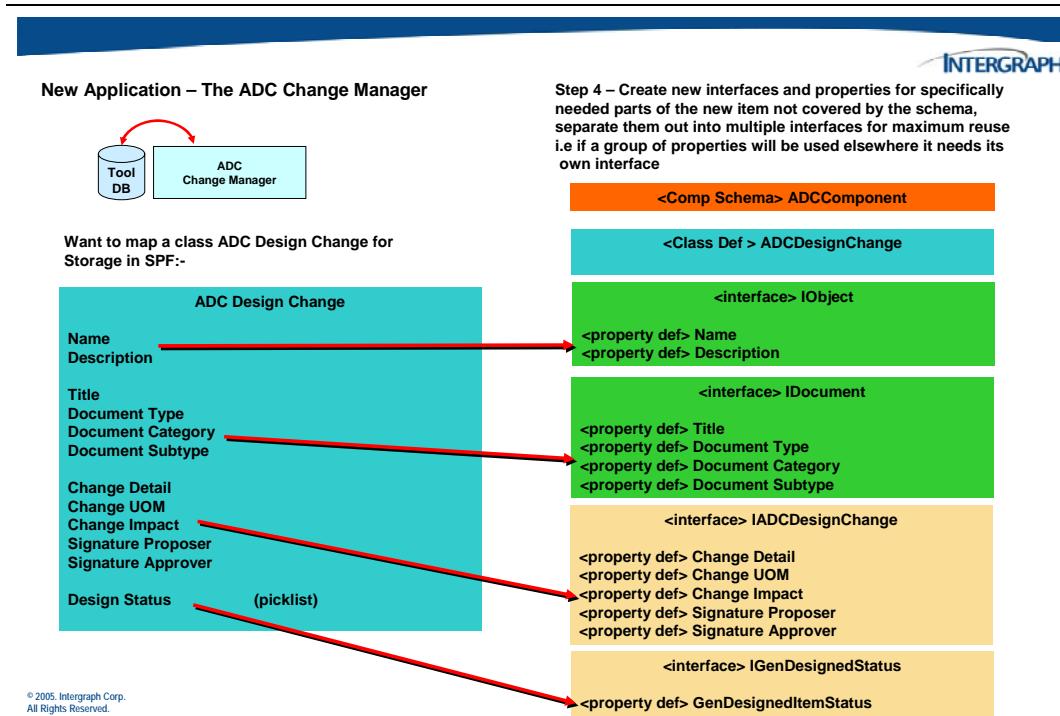
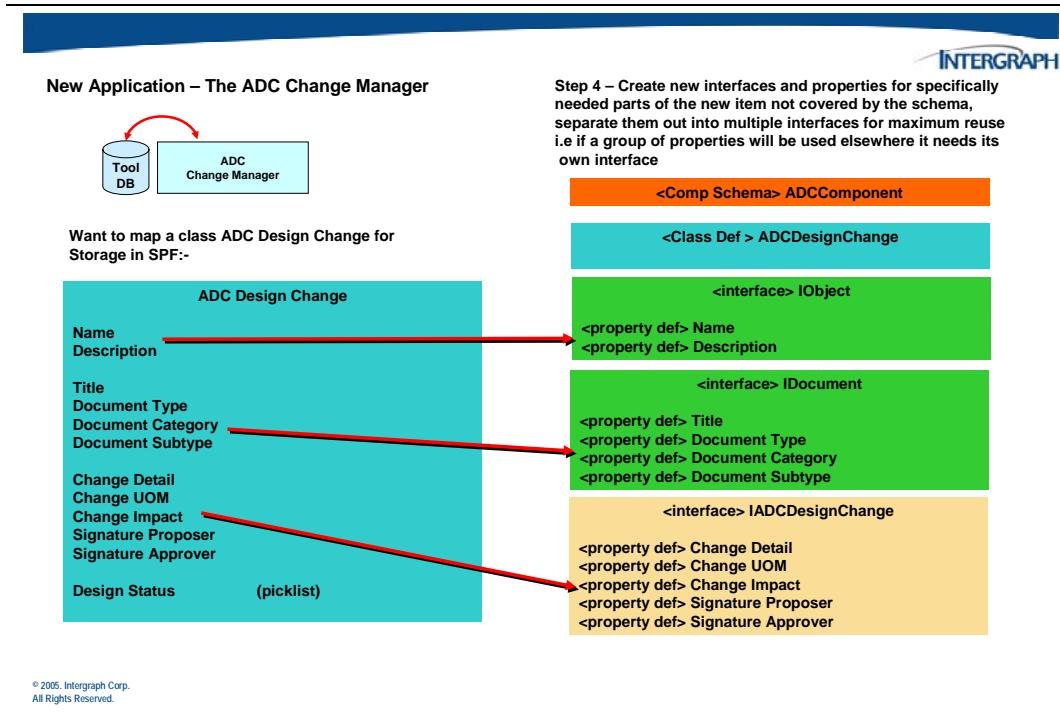


© 2005, Intergraph Corp.
All Rights Reserved.

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.
- ❑ **String** – Specifies that values for the property can contain alphanumeric strings with some punctuation, such as periods (.), underbars (_), question marks (?), and so on.
- ❑ **YMD** – Specifies that the property value is a date (year, month, day).
- ❑ **EnumListType** – Specifies that property values are defined by an enumerated list.
- ❑ **UoMList** – 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.

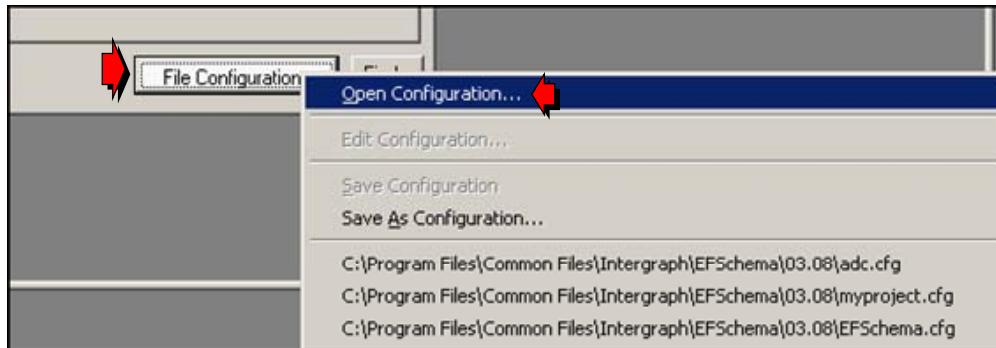
4.3 Interactive Activity – Creating Property Definitions





Opening a Package

- From the Workflows dialog box, choose the File Configurations button and select **Open Configuration..**

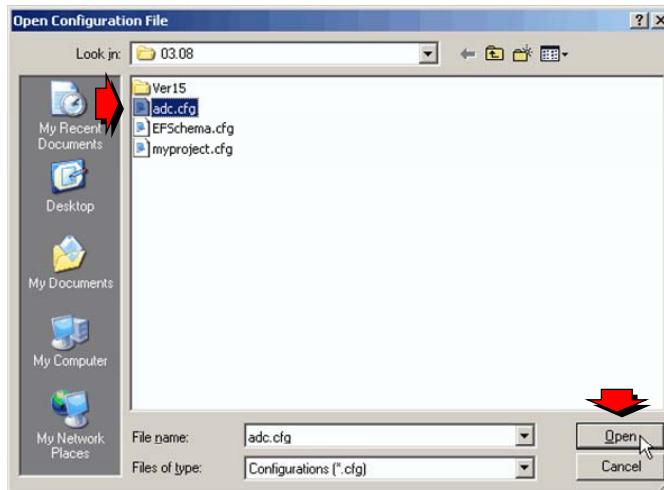


© 2005. Intergraph Corp.
All Rights Reserved.



Opening a Package

- In the **Open Configuration File** dialog, select the adc configuration then click Open.

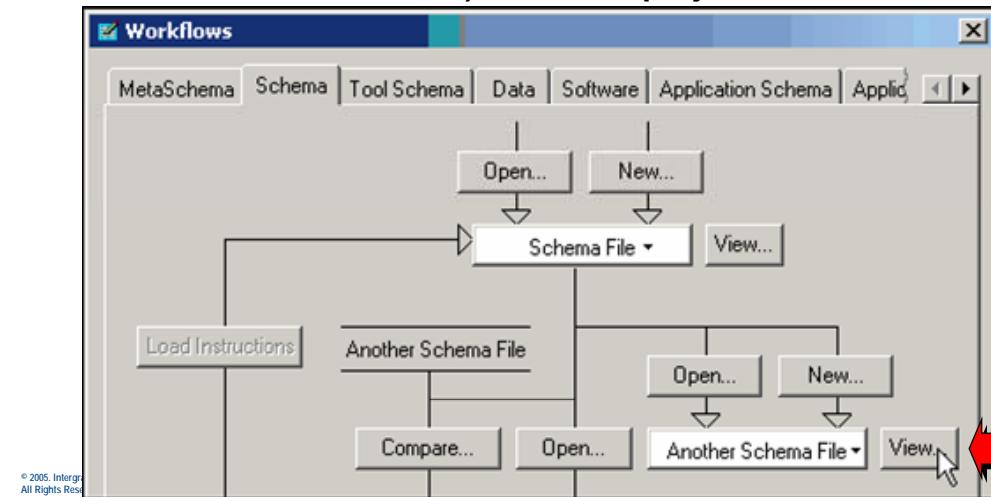


© 2005. Intergraph Corp.
All Rights Reserved.



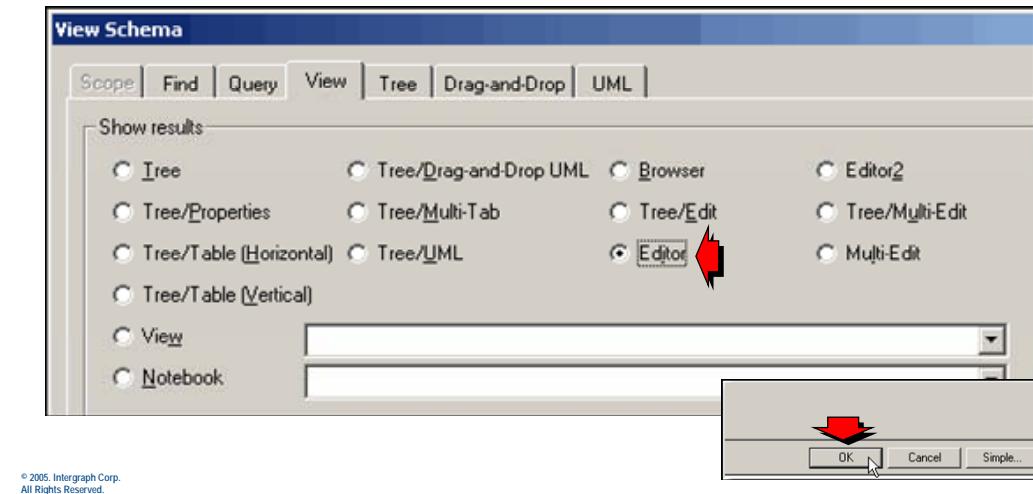
Opening a Package

- ❑ In the **Workflows** dialog box, select the **View** button (next to Another Schema File) to edit the project schema file.



Opening a Package

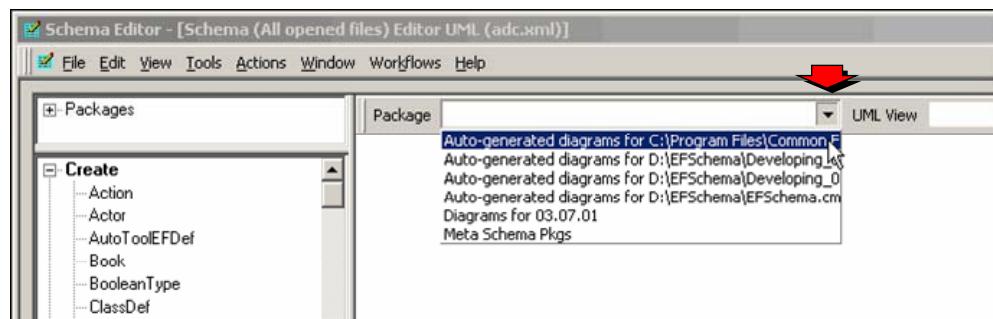
- ❑ In the **View Schema** dialog box, select the **Editor** view and click **OK**.





Opening a Package

- Select the **Package** list button to see the available packages.

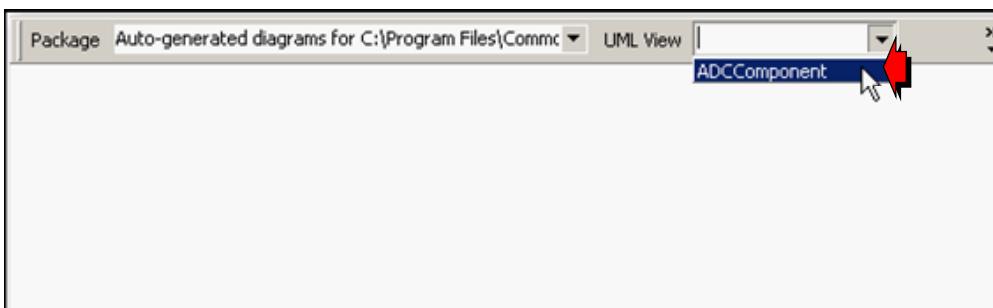


© 2005, Intergraph Corp.
All Rights Reserved.



Opening a Package

- Select **ADCComponent** from list of available UML views.

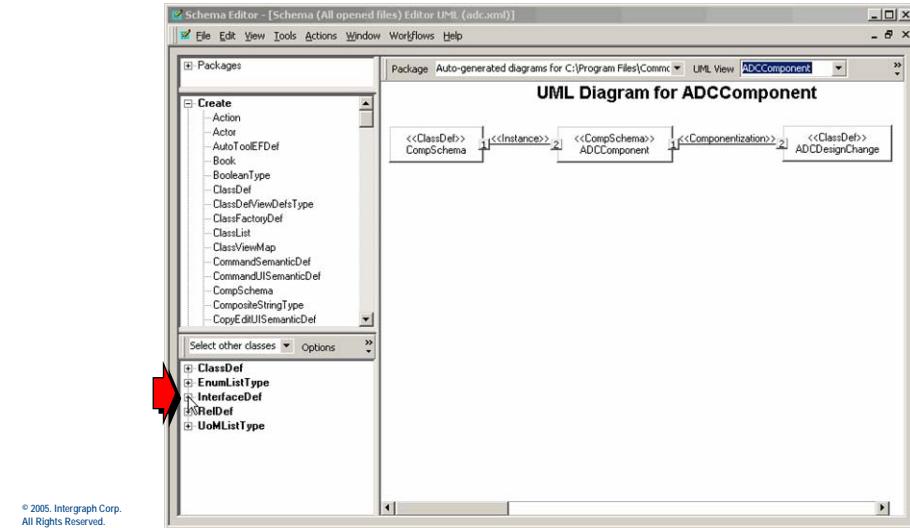


© 2005, Intergraph Corp.
All Rights Reserved.



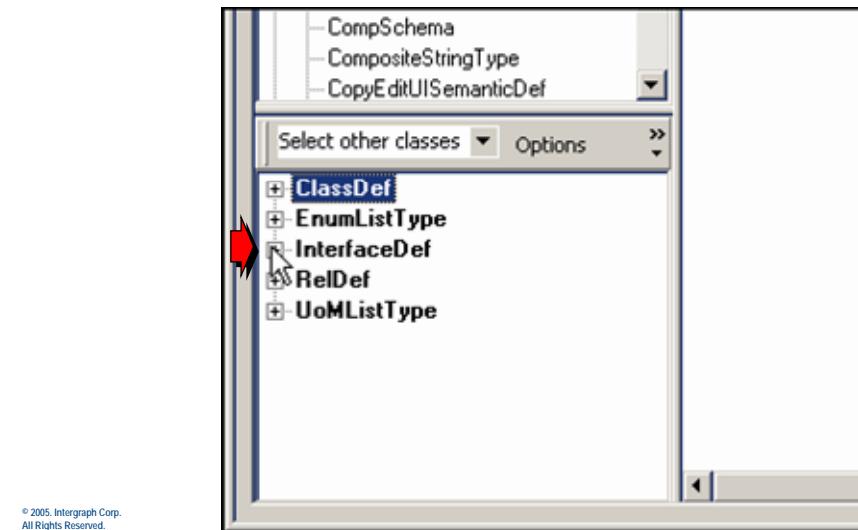
Create a Property Definition

- Click the + to expand the list of interface definitions.



Create a Property Definition

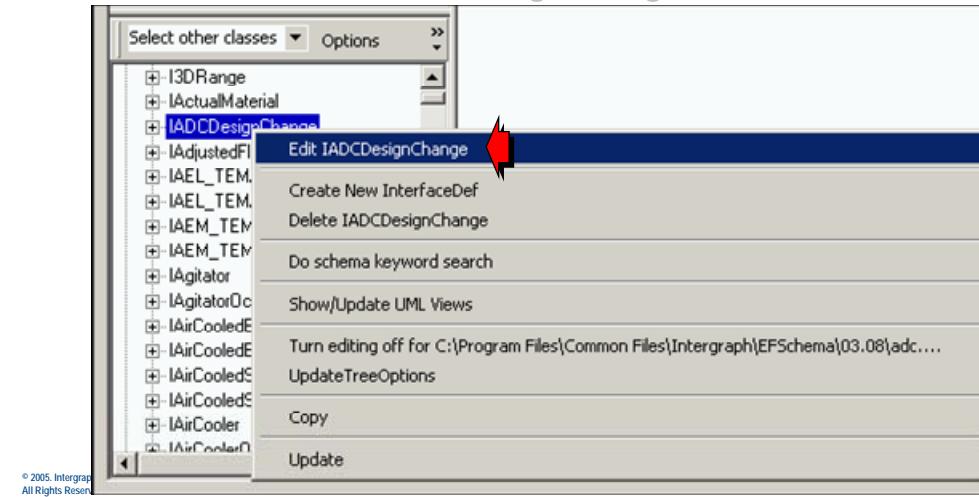
Zoomed in view of the *Schema Editor UML* view.





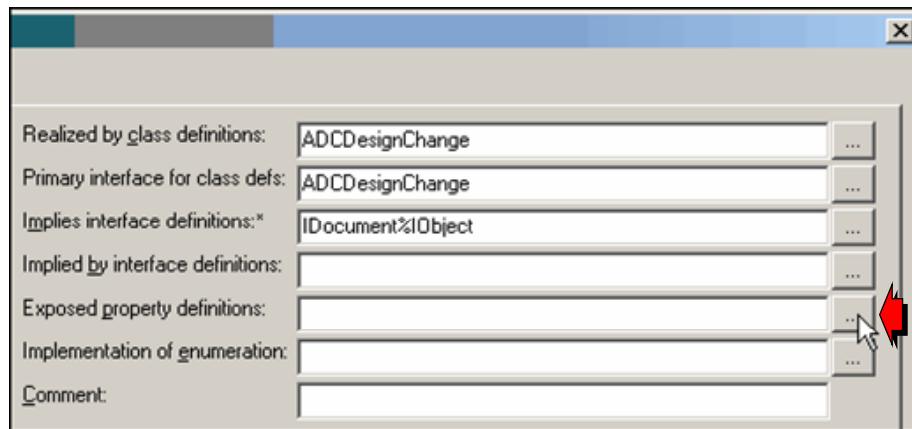
Create a Property Definition

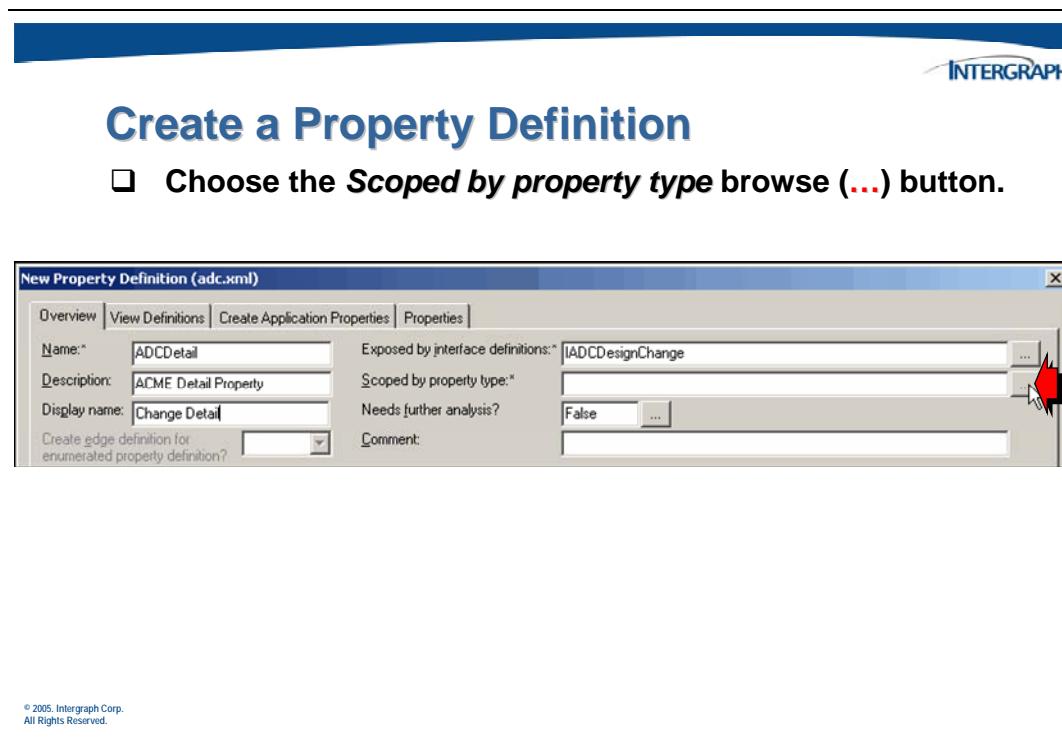
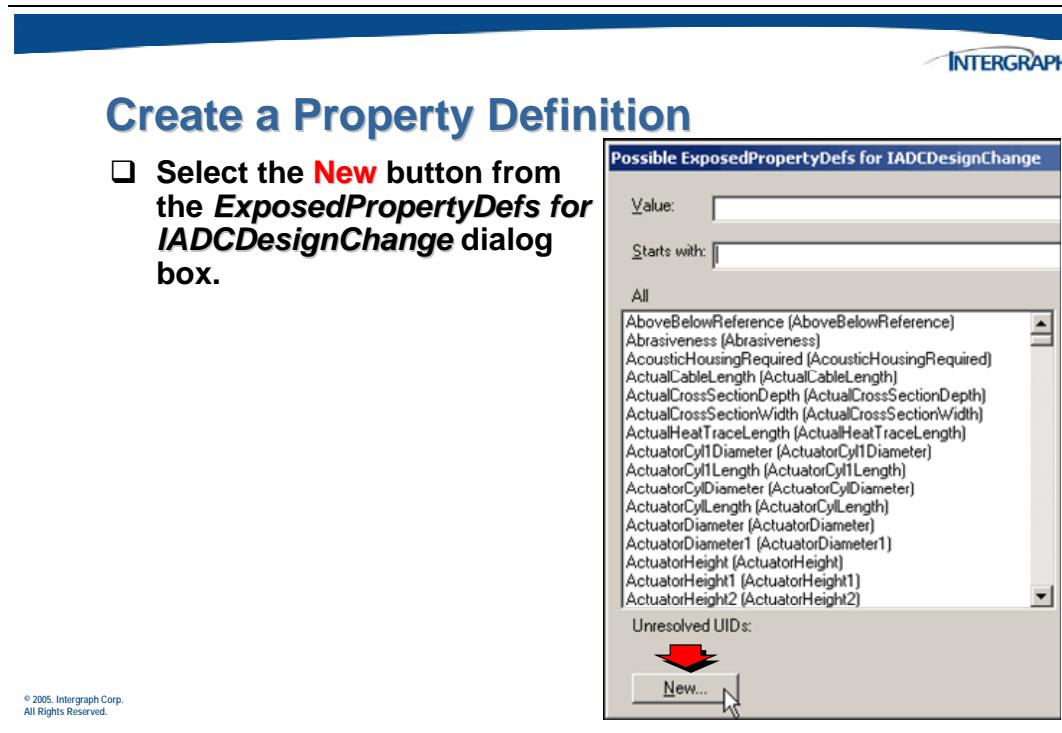
- Right-click on the **IADCDesignChange** interface definition and select *Edit IADCDesignChange*.



Create a Property Definition

- Choose the *Exposed property definitions* browse (...) button.





Create a Property Definition

Possible ScopedByPropertyType Values for ADCDetail

Value:

Starts with: **stl** 

All

Straight Pipe Nozzle ({311F5D55-96F6-4128-AAFB-1F69E2C...} 
 Straight section ({81CD92A2-BC07-11D6-BDBC-00104BCC...}
 Straight Sections ({1AB404C3-4223-4828-898C-3451A6F4C...}
 Strainer ({39EC4BD5-578B-4884-9A3E-47D910DE224C})
 Strainer ({81CD9486-8C07-11D6-BDBC-00104BCC2B69})
 Strand cutter ({238D25CB-A208-4803-841E-CC9B0546A65...}
 StressJouM (StressJouM)
 Stretch-wrapper ({F84C460E-1BF8-4501-B0FA-BD5F6471C...}
 string(string)
 string128(string128)
 string16(string16)
 string256(string256)
 string32(string32)
 string64(string64)
 string8(string8)
 Structural ({8DA446CE-D26B-4EA5-8FFF-15CD3427692C})

Unresolved UIDs:

New...

□ Enter the first several characters of the type, then select the type of data for the new property.

Create a Property Definition

□ Click OK to add the new *ADCDetail* property.

New Property Definition (adc.xml)

Overview | View Definitions | Create Application Properties | Properties

Name: **ADCDetail**

Description: ACME Detail Property

Display name: Change Detail

Expose by interface definitions: **IADCDesignChange**

Scoped by property type: **string**

Needs further analysis? False

Create edge definition for enumerated property definition?

Comment:

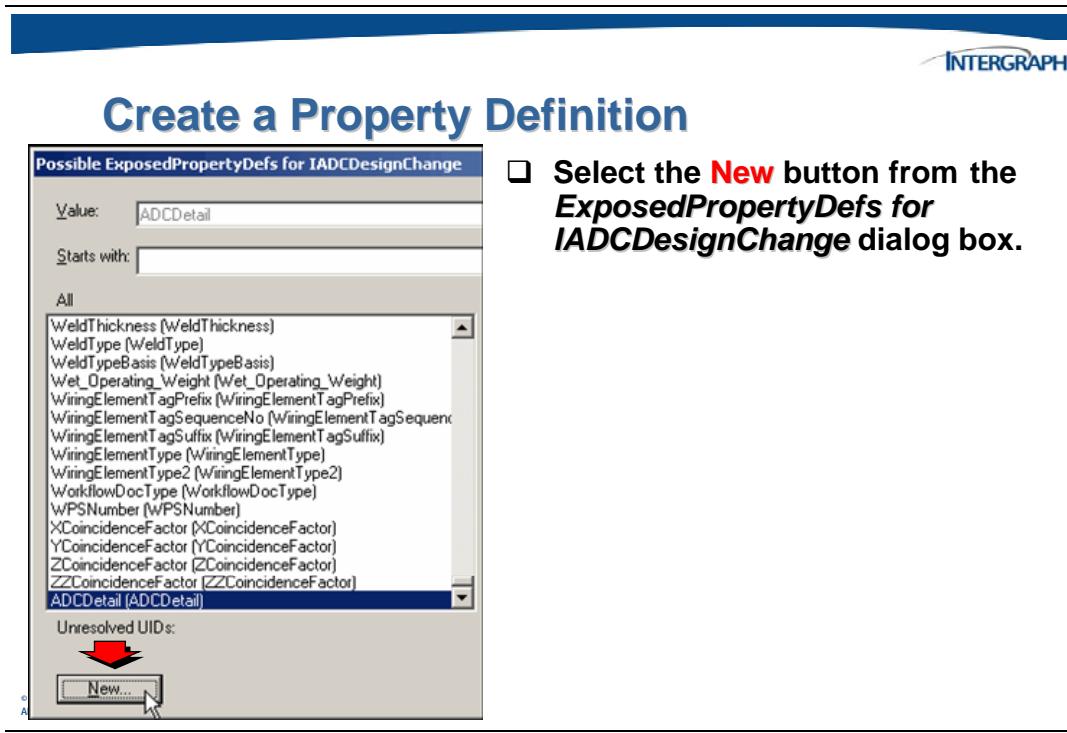
Mapping

```

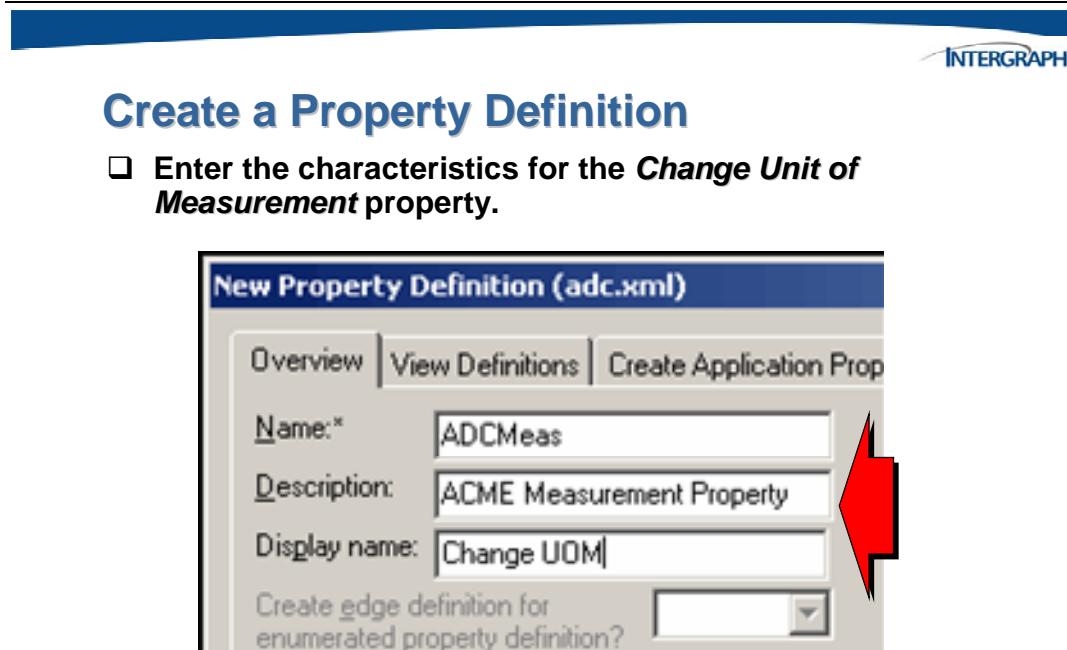
graph TD
    IADCDesignChange -->|<<Exposes>>| ADCDetail
    ADCDetail -->|<<ScopedBy>>| string
  
```

OK  **Cancel** **Rename**

© 2005, Intergraph Corp.
All Rights Reserved.



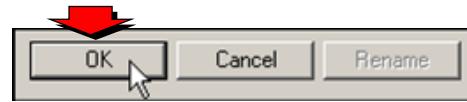
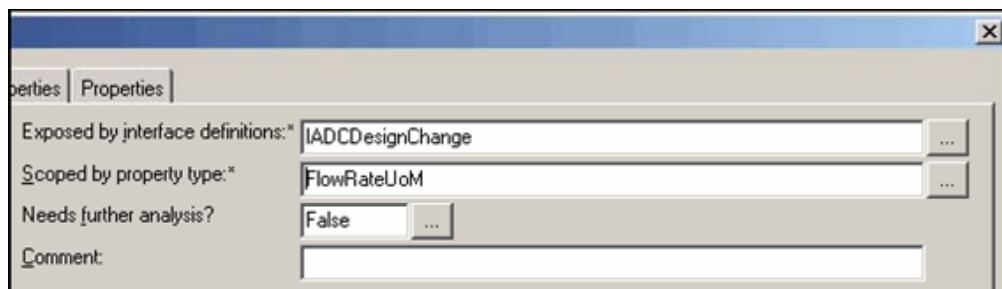
- Select the **New** button from the **ExposedPropertyDefs for IADCDesignChange** dialog box.





Create a Property Definition

- Click **OK** to add the new **ADCMeas** property.

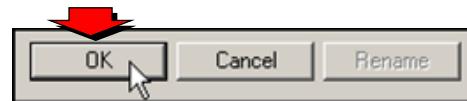
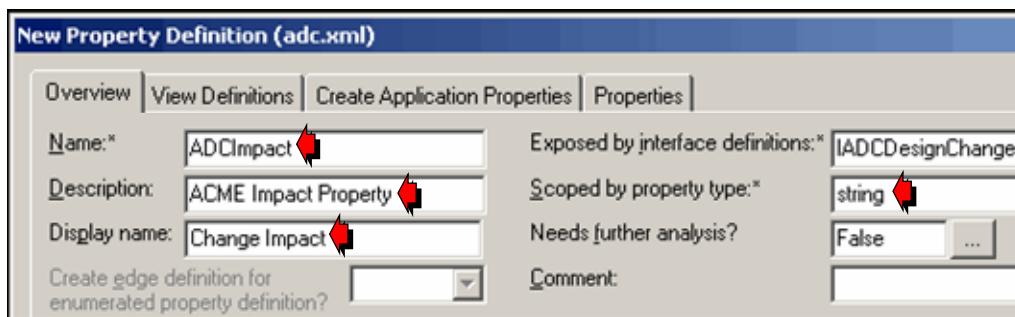


Again, select the **New** button from the *ExposedPropertyDefs* for *IADCDesignChange* dialog box and enter the characteristics for the new property.

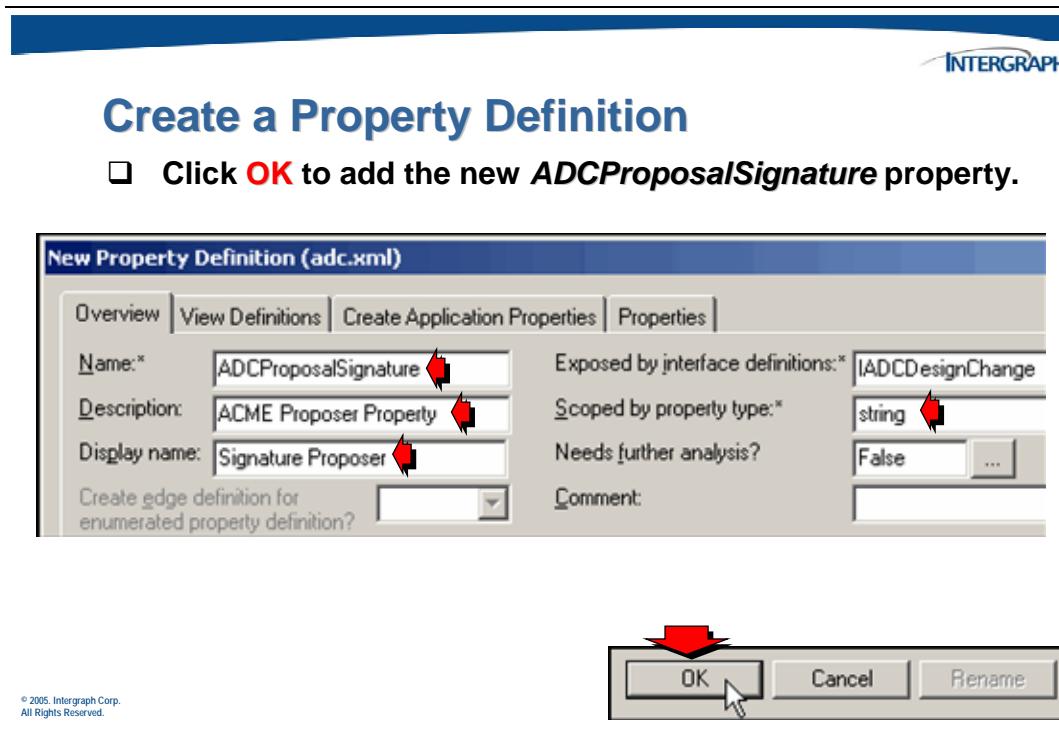


Create a Property Definition

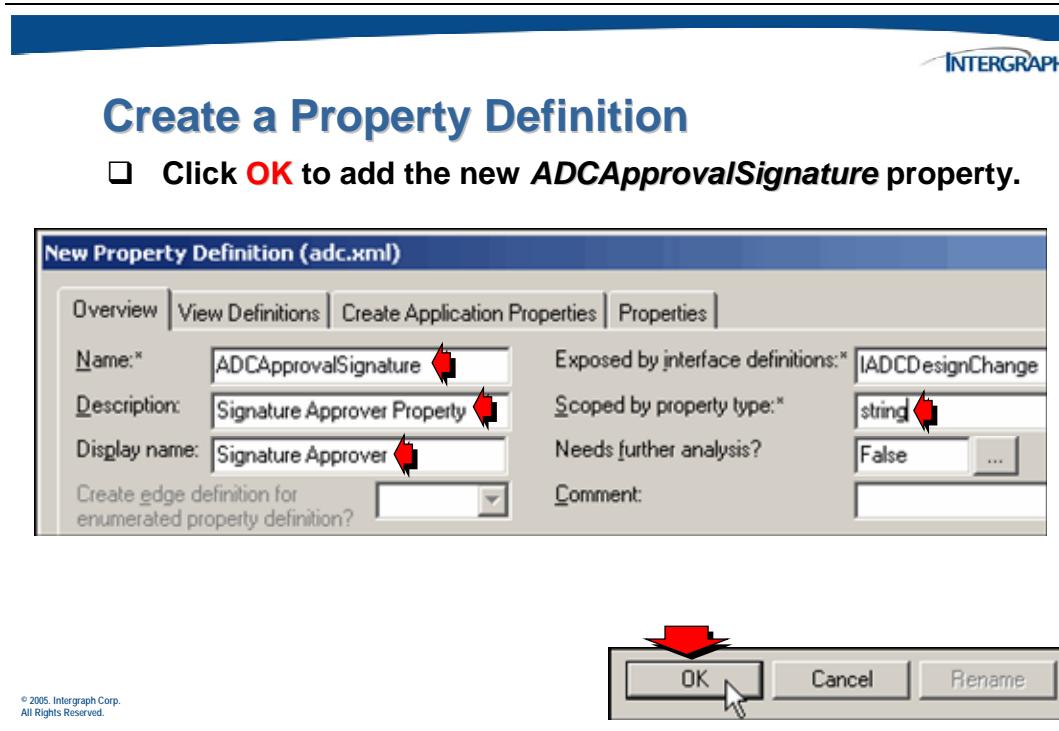
- Click **OK** to add the new **ADCImpact** property.



Repeat the procedure to add the next property.



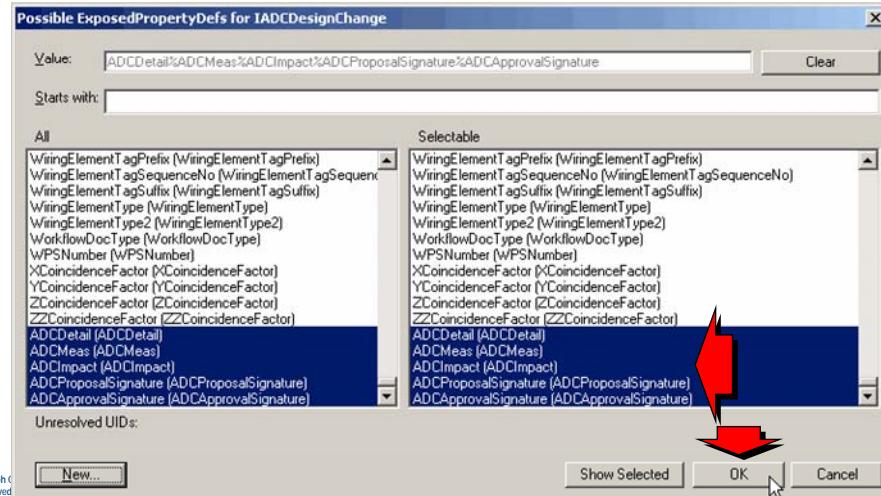
Repeat the procedure again to add the last property.





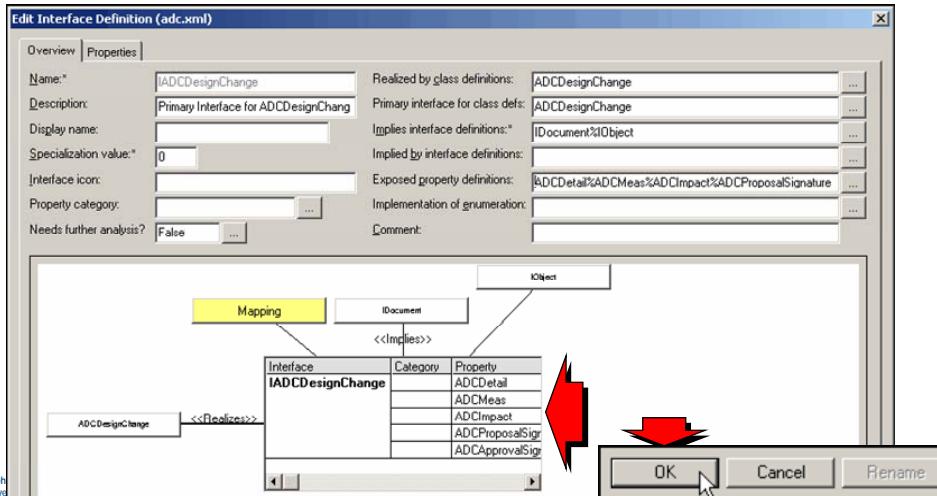
Create a Property Definition

- Verify all the new property types are highlighted, then click **OK**.



Create a Property Definition

- Once all the property definitions have been defined, click **OK**.



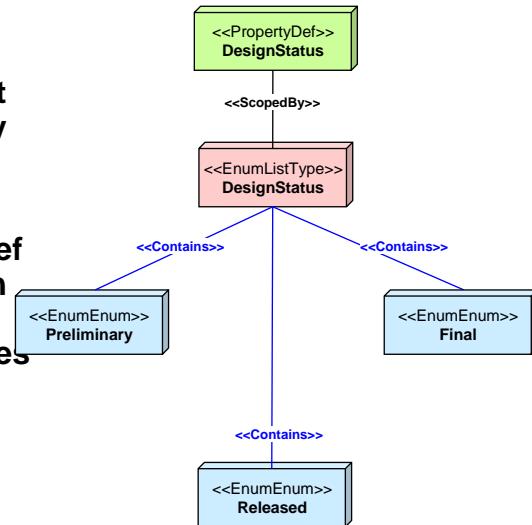
4.4 Enumerated List Type

An enumerated list (EnumListType) is one of the property types that can scope property definitions. Enumerated lists are sometimes called “Picklists”. 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.

Property Types - Enumerated List Type

**Property definitions of the
enumerated list type
(EnumListType) have a list
of possible string property
values defined for them in
an enumerated list.**

**Any value for a PropertyDef
of this type must match an
entry in the list of
enumerated property values
defined for the property
type.**



© 2005, Intergraph Corp.
All Rights Reserved.

Enumerated lists may be combined to form an enumeration hierarchy. When this happens, a special property type (enumerated list level type) is used to describe all of the enumerations at a given level in the enumeration hierarchy.



Property Types - Enumerated List Type

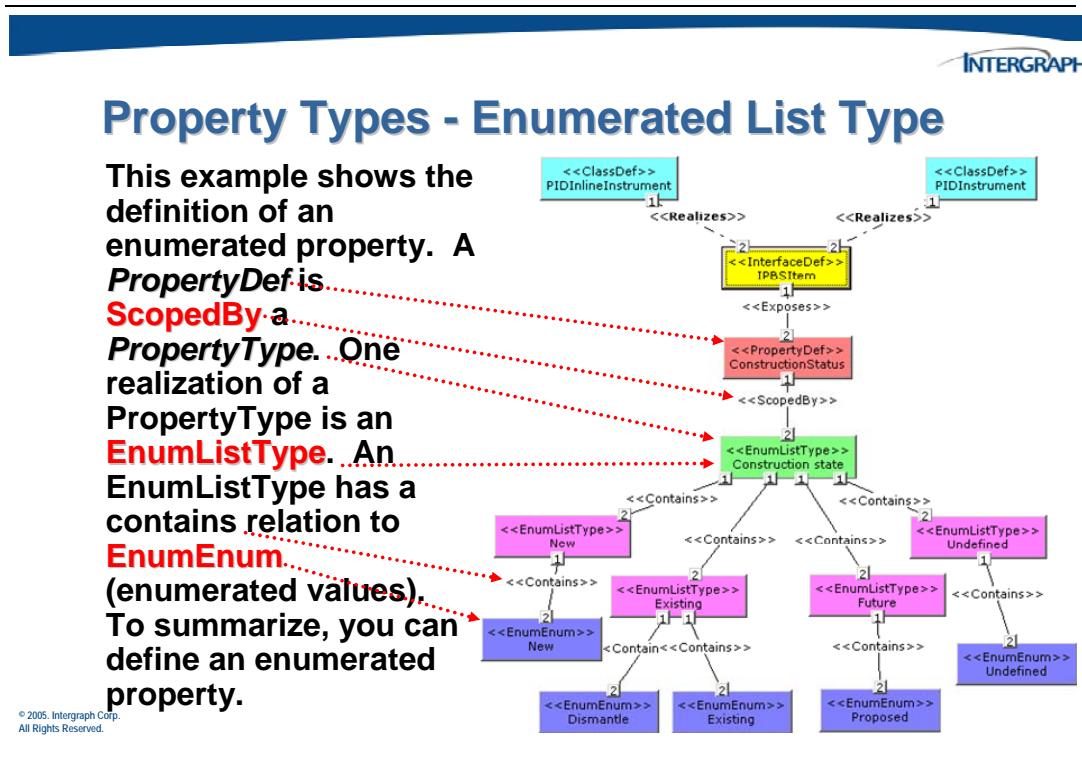
Enumerated lists can be combined to form an enumeration hierarchy.

All nodes in the hierarchy that are `EnumListTypes` are enumerated entries for the `EnumListType` above them.

All `EnumListType` nodes must contain `EnumEnums` (list entries) or other `EnumListTypes` (enumerated lists).

For example, the typing of equipment consists of a multi-level hierarchy where each level down the hierarchy provides a more specific definition of the equipment. Every branch node in a hierarchy is itself an enumerated list as well as an enumerated value. Only the topmost node in the hierarchy is not an enumerated value (just an enumerated list) and only the leaf nodes in the hierarchy are not enumerated lists (just enumerated values).

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



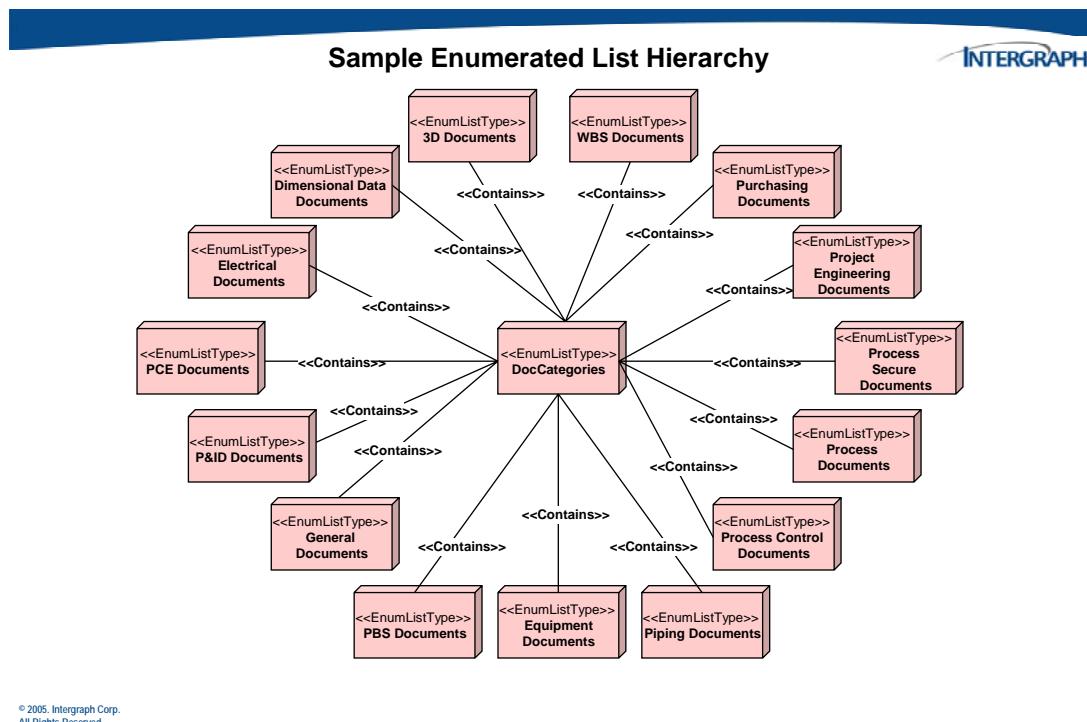
The **EnumListType** class definition is used for enumerated list property types. **EnumListType** realizes **I.PropertyType**. **EnumListType** also realizes **IEnumListType**, the interface definition that supports enumerated list behavior.

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 **IOObject**, 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.

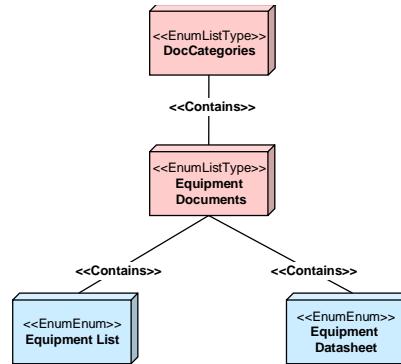
The topmost node in an enumerated list hierarchy is an instance of the `EnumListType` class definition. This instance does not realize `IEnumEnum` since it is not an enumerated entry.



© 2005, Intergraph Corp.
All Rights Reserved.

Each `EnumListType` in the `DocCategories` enumerated list either contains enumerated list entries (`IEnumEnum`) or other nested lists (`EnumListType`). An example of each of these cases appears below.

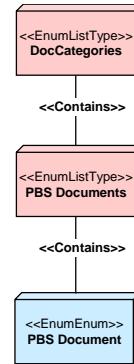
Sample Enumerated List Hierarchy



© 2005. Intergraph Corp.
All Rights Reserved.

The Equipment Documents EnumListType contains two enumerated list values:
Equipment List and Equipment Datasheet.

Sample Enumerated List Hierarchy



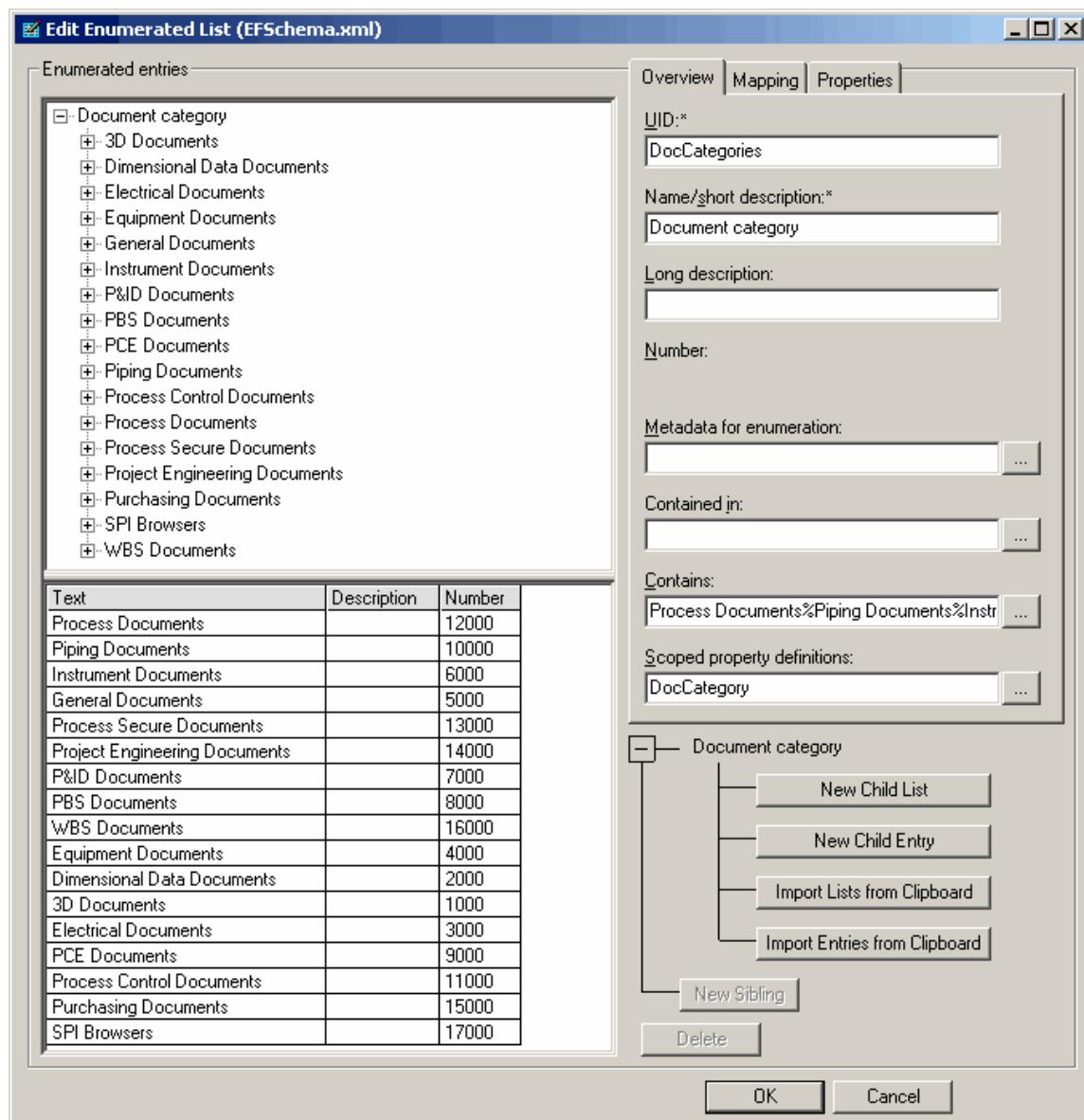
© 2005. Intergraph Corp.
All Rights Reserved.

The PBS Documents EnumListType contains an EnumEnum called PBS Document.

4.4.1 Properties of an Enumerated List Type

The following options are available when you create or edit an enumerated list type:

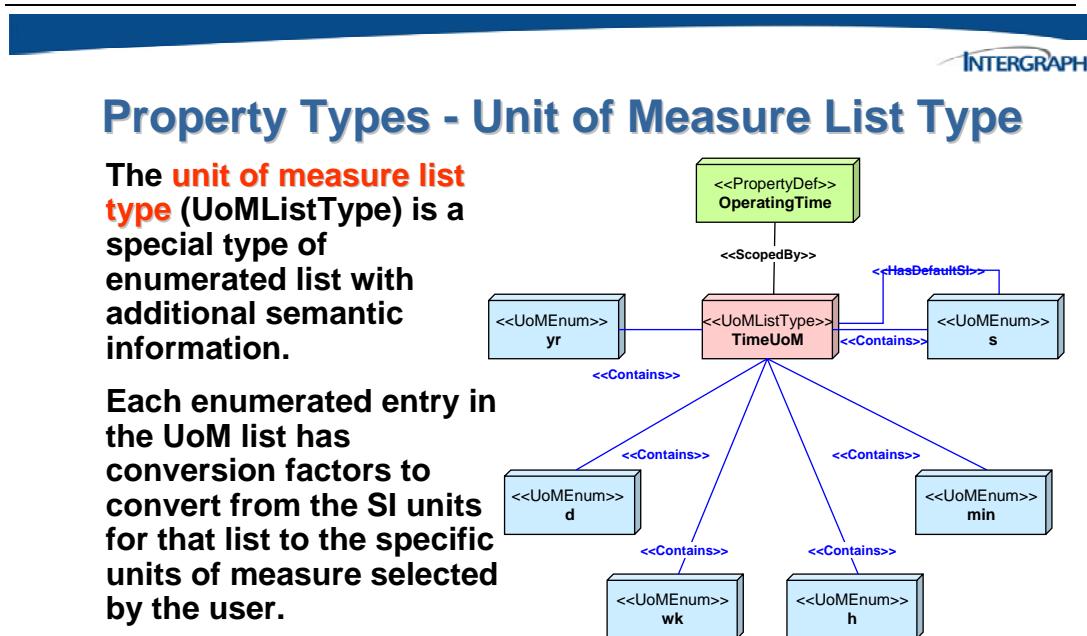
- Enumerated entries** – Displays the enumeration hierarchy as you add child lists and entries to the main enumerated list.
- UID** – Displays the unique identifier for the enumerated list. The UID is automatically assigned by the Schema Editor.
- Short description** – Specifies a short description for the enumerated list. The short description is used as the name for the enumerated list throughout the Schema Editor.
- Long description** – Specifies a long description for the enumerated list.



- Number** – Specifies a number that can be used as a secondary key to identify the enumeration.
- Metadata for enumeration** – Defines a relationship between the enumerated entry and another object or objects in the schema that provide more information about that enumeration. This is frequently used to relate enumerations to interface definitions for type hierarchies.
- Contained in** – Identifies the names of other enumerated lists that contain this enumerated list.
- Contains** – Identifies the names of other enumerated lists and enumerated list entries that this list contains.
- Scoped property definitions** – Identifies the property definitions that have a ScopedBy relationship with this enumerated list type.
- New Child List** – Creates a new child list (EnumListType) under the selected node in the enumeration hierarchy. When you select an enumerated entry and create a new child list under it, it becomes an EnumListType object.
- New Child Entry** – Creates a new child enumerated list entry (EnumEnum) under the selected enumerated list in the hierarchy.
- Import Lists from Clipboard** – Imports enumerated lists that you have copied from a spreadsheet to the Clipboard into the Schema Editor. When you import lists, the Schema Editor tries to map the contents of the Clipboard to the appropriate enumerated list properties based on the imported values. You can change this mapping manually in the **Import from Clipboard** dialog box.
- Import Entries from Clipboard** - Imports enumerated list entries that you have copied from a spreadsheet to the Clipboard into the Schema Editor. When you import entries, the Schema Editor tries to map the contents of the Clipboard to the appropriate enumerated entry properties based on the imported values. You can change this mapping manually in the **Import from Clipboard** dialog box.
- New Sibling** – Creates a new enumerated list or enumerated list entry at the same level as the list or entry selected in the **Enumerated entries** tree.
- Delete** – Deletes the selected node in the **Enumerated entries** tree.

4.4.2 Unit of Measure List Types

The unit of measure (UoM) list type is specialization of the enumerated list type (IUoMListType implies IEnumListType). A unit of measure list contains unit of measure enumerated values (UoMEnum). A UoMEnum contains two conversion factors: A (multiplier) and B (addend). These conversion factors are used to convert the value to SI units.



© 2005. Intergraph Corp.
All Rights Reserved.

For example, the given unit (x) is multiplied by A and then B is added to that value to determine the SI value ($y = Ax + B$).

NOTE: SI stands for International System of Units and is the modern metric system of measurement. Long the language universally used in science, the SI has become the dominant language of international commerce and trade. SI is the basis used for the conversion of UoM's.

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 this 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.

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

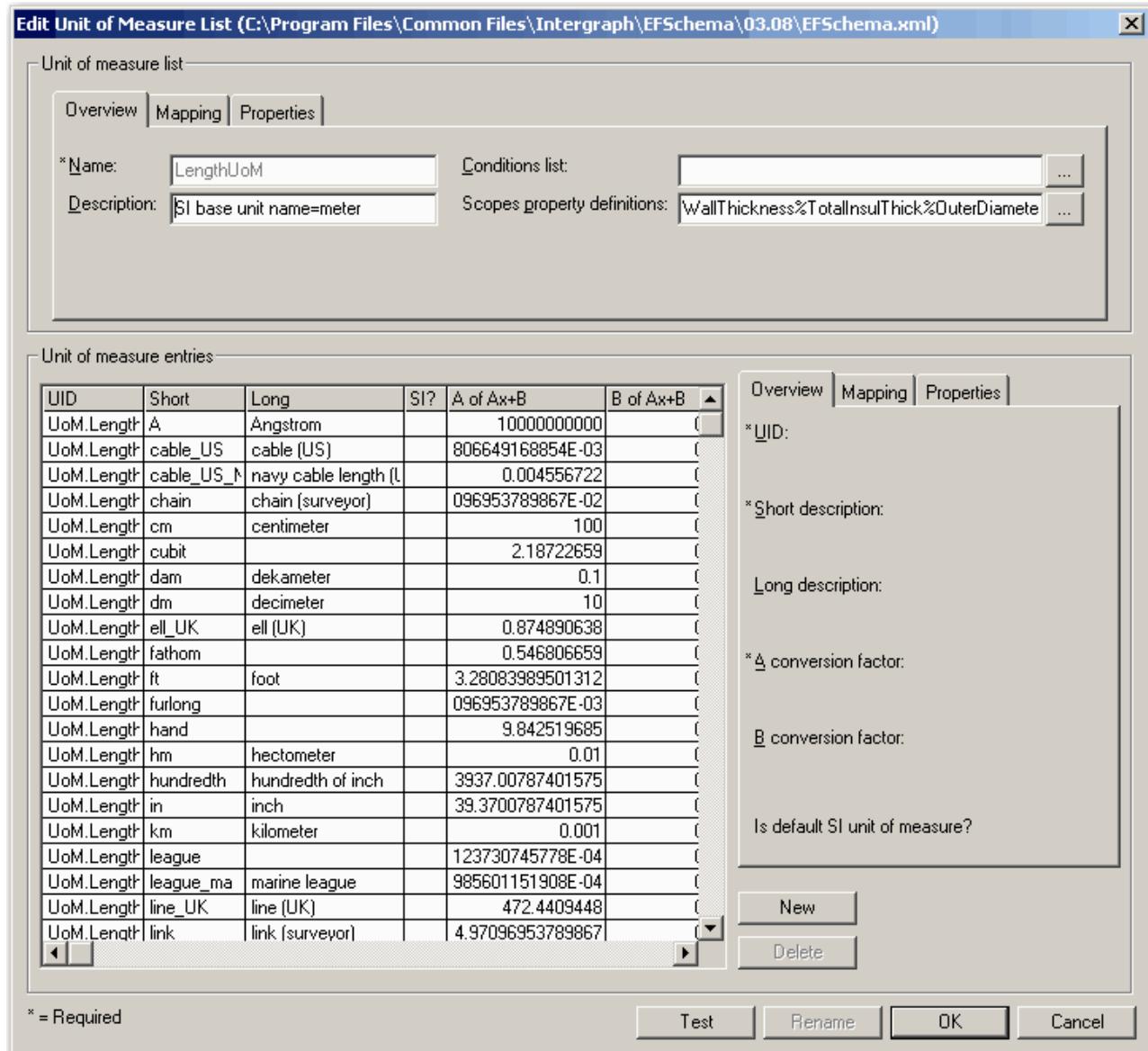
an associated enumerated list that identifies whether the pressure value is absolute or gauge (@ Gauge).

A PropertyDef 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.

4.4.3 Properties of a Unit of Measure List Type

The following options are available when you create a new enumerated list type:

- Name** – Specifies the name of the unit of measure list.
- Description** – Specifies a description for the unit of measure list
- Conditions list** – Identifies the enumerated list that contains the possible conditions for a property scoped by this unit of measure list. For example, the pressure unit of measure list uses a conditions list that identifies whether the pressure value is absolute or gauge.



- Scopes property definitions** – Lists the property definitions that are scoped by this unit of measure list.

- Unit of measure entries** – Displays the unit of measure entries for this list in tabular format.
- UID** – Displays the unique identifier for the selected entry in the list. The UID is automatically assigned by the Schema Editor, but you can change it.
- Short description** – Specifies a short description for the unit of measure list entry. The short description is used as the name for the entry throughout the Schema Editor.
- Long description** – Specifies a long description for the unit of measure list entry.
- A conversion factor** – Specifies the multiplier used to convert from this unit of measure to SI units. Conversions are done using $y = Ax + B$, where x is the value in this unit of measure and y is the value in SI units.
- B conversion factor** – Specifies the addend used to convert from this unit of measure to SI units. Conversions are done using $y = Ax + B$, where x is the value in this unit of measure and y is the value in SI units.
- Is default SI unit of measure?** – Specifies whether this unit of measure is the default SI unit of measure for this list.
- New** – Adds a new unit of measure entry to the list.
- Delete** – Deletes the selected unit of measure entry from the list.
- Test** – Allows you to test conversions among units of measure in this list using the A and B conversion factors defined for each unit of measure and any specified conditions for each unit of measure.

4.5 Interactive Activity – Creating an Enumerated List Property

New Application – The ADC Change Manager

Tool DB → ADC Change Manager

Want to map a class ADC Design Change for Storage in SPF:-

ADC Design Change

- Name
- Description
- Title
- Document Type
- Document Category
- Document Subtype
- Change Detail
- Change Type
- Change Impact
- Signature Proposer
- Signature Approver
- Design Status
(Picklist :- Created/Approved/Required/InProgress/Completed)

Step 6 – Scope the design status property using a “picklist”
an enum list type and enum enums

```

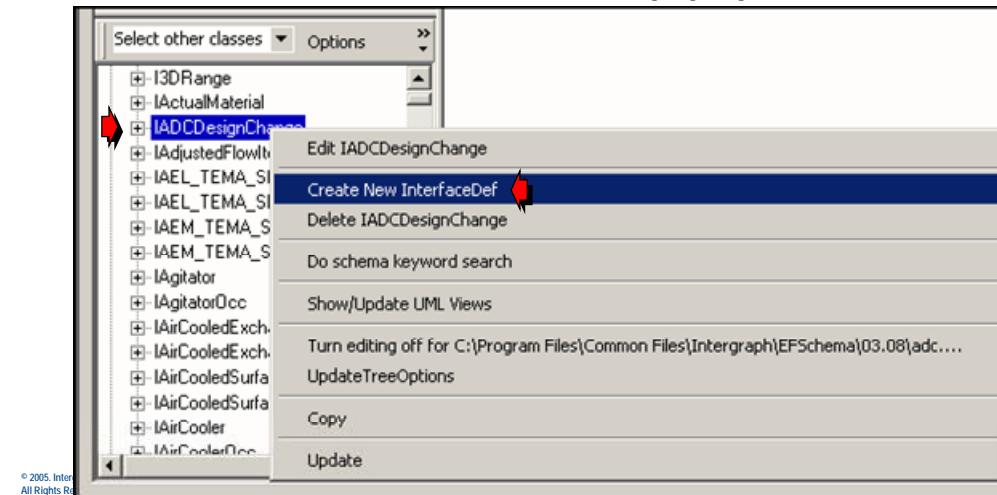
classDiagram
    interface IGenDesignedItem {
        <exposes>
        propertyDef GenDesignedStatus
    }
    class GenDesignedStatus {
        <scoped by>
        enumListType GenDesignStatusPL
    }
    class GenDesignStatusPL {
        <contains>
        enumEnum Created
        enumEnum Approved
        enumEnum Completed
        enumEnum InProgress
        enumEnum Required
    }
    class Created
    class Approved
    class Completed
    class InProgress
    class Required
  
```

© 2005, Intergraph Corp.
All Rights Reserved.



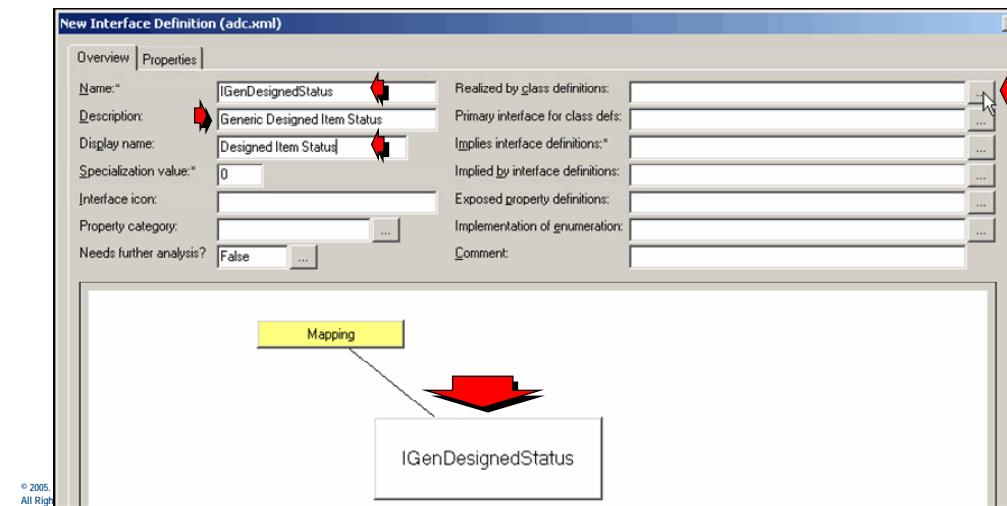
Creating an Enumerated Property

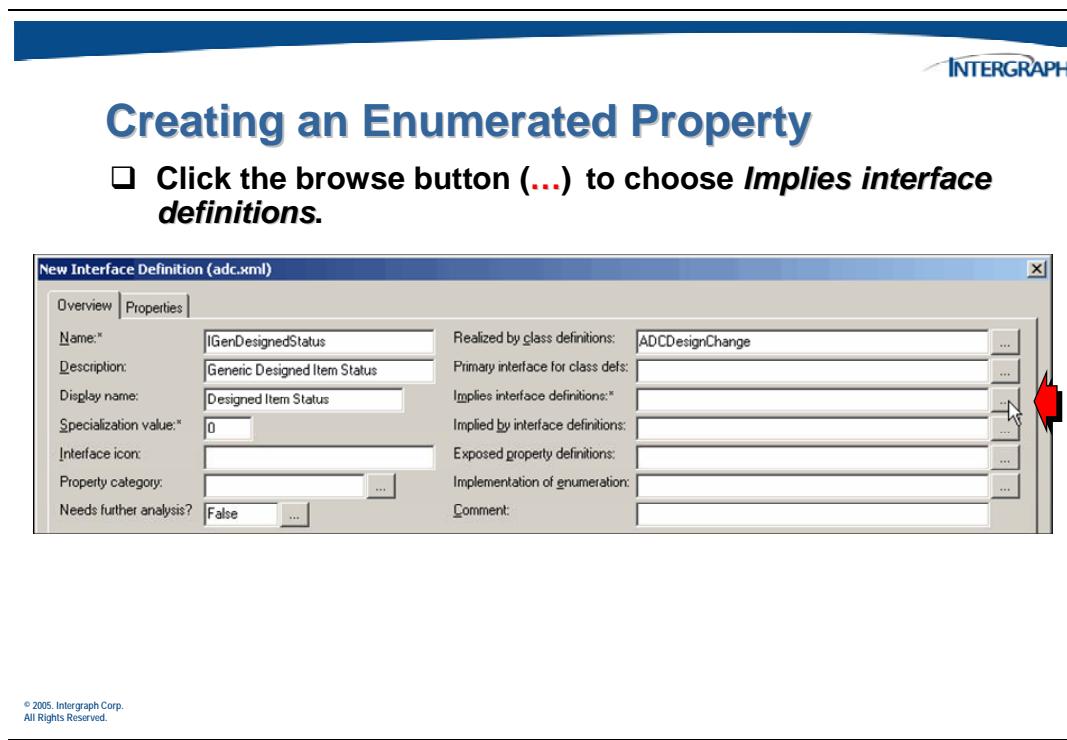
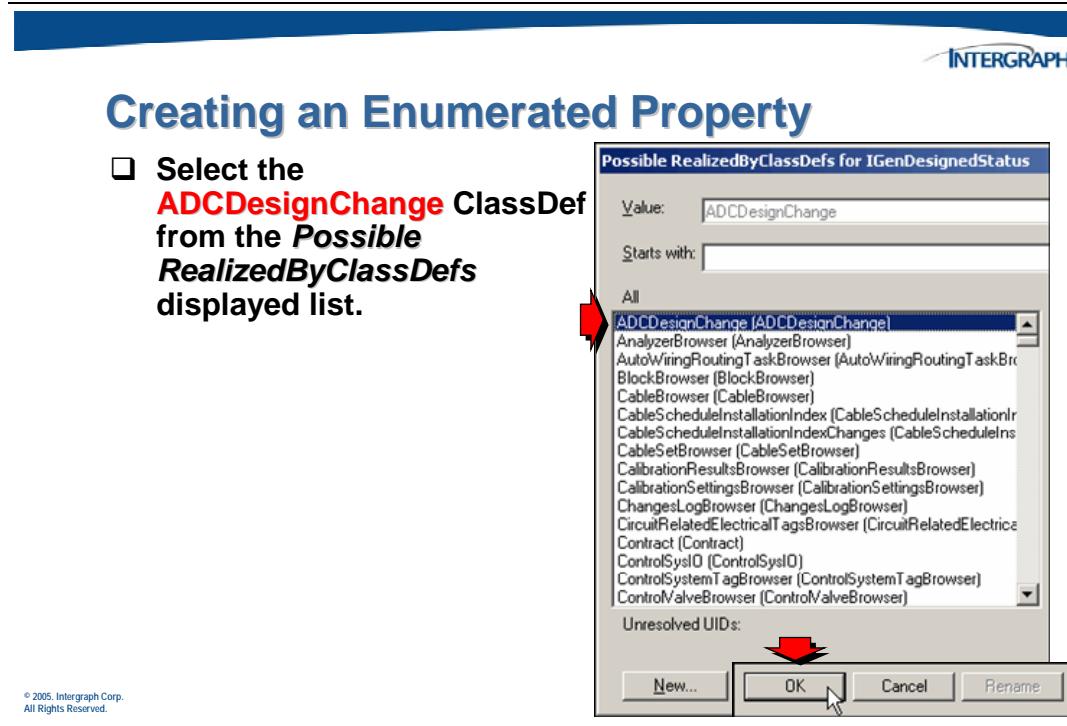
- ❑ Right-click on the **IADCDesignChange** interface and select **Create New InterfaceDef** from the pop-up menu.



Creating an Enumerated Property

- ❑ In the **New Interface Definition** dialog box, enter the information to create a new **InterfaceDef**.

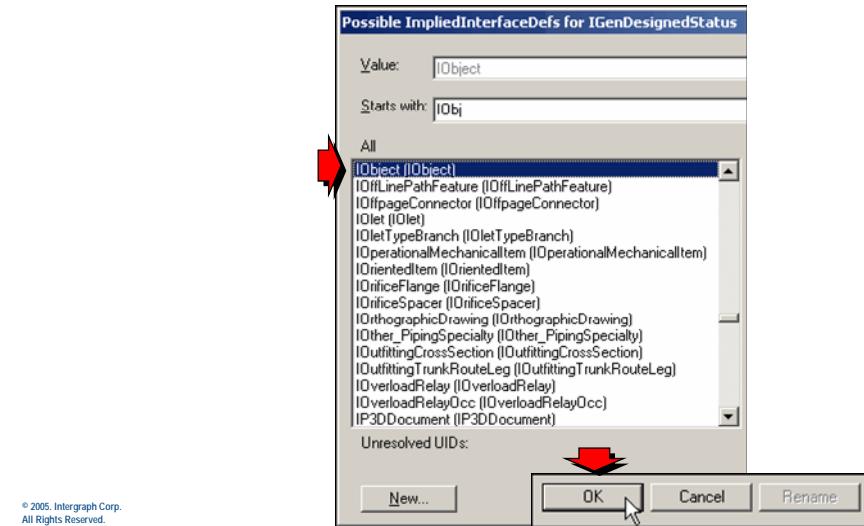






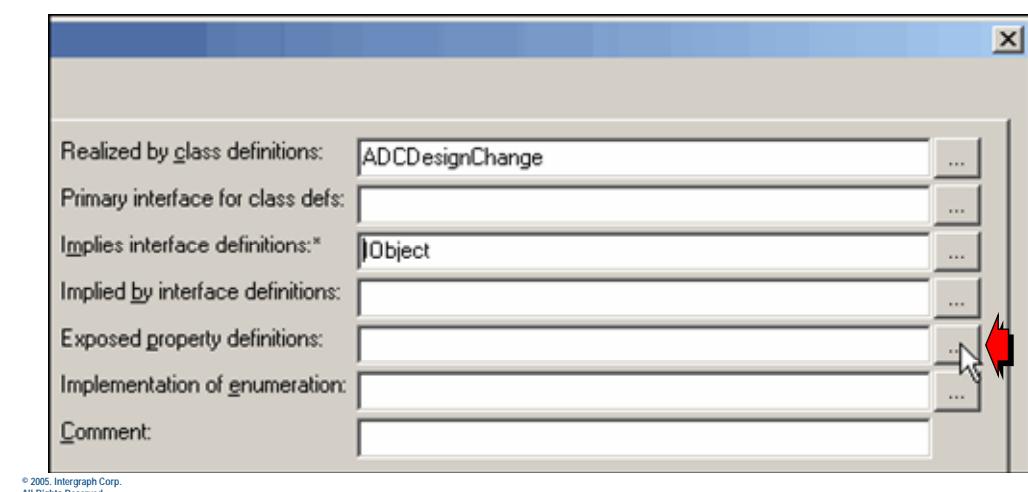
Creating an Enumerated Property

- Select the **IObject** interface from the displayed list.



Creating an Enumerated Property

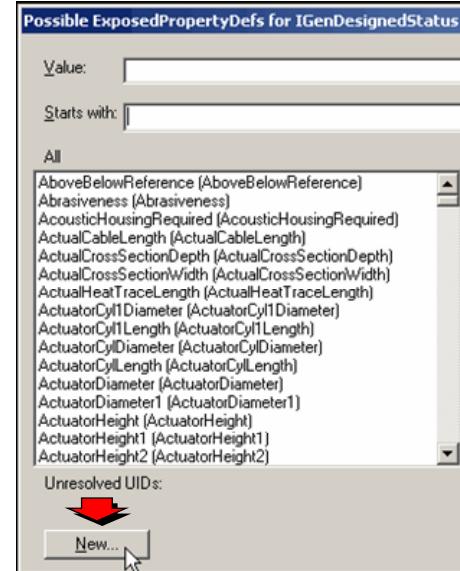
- Choose the **Exposed property definitions** browse (...) button.





Creating an Enumerated Property

- Select the **New** button from the **Possible ExposedPropertyDefs for IGenDesignedStatus** dialog box.

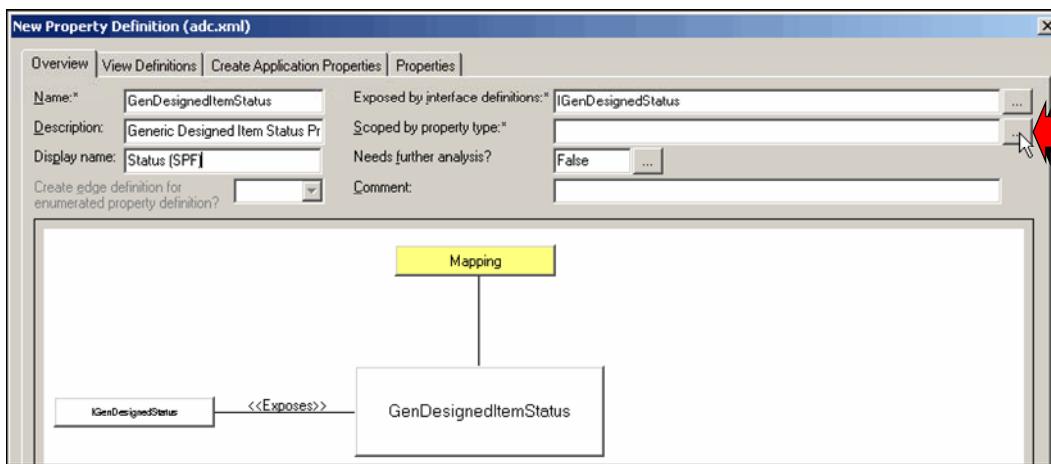


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated Property

- Choose the **Scoped by property type** browse (...) button.

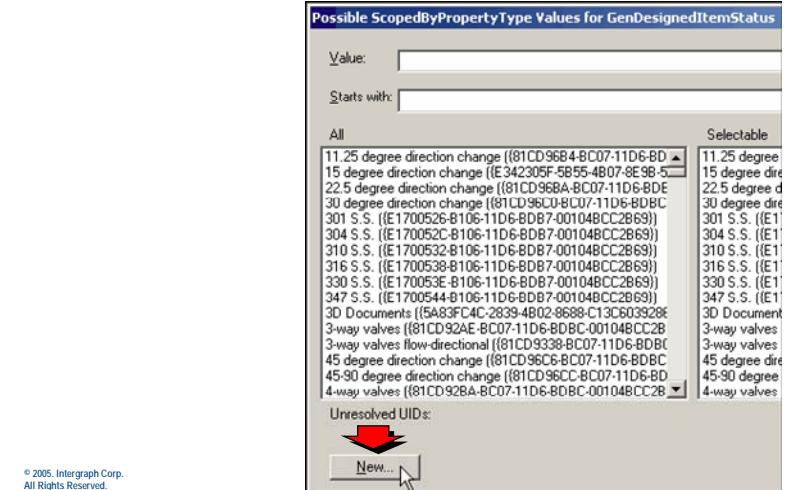


© 2005, Intergraph Corp.
All Rights Reserved.



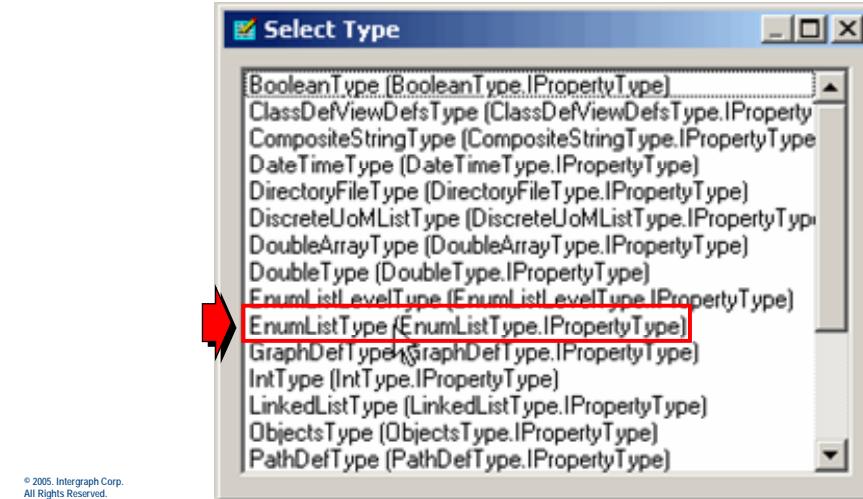
Creating an Enumerated Property

- Select the **New** button from the **Possible ScopedByPropertyType Values** dialog box.



Creating an Enumerated List Type

- Select the **EnumListType** entry from the **Select Type** dialog box.

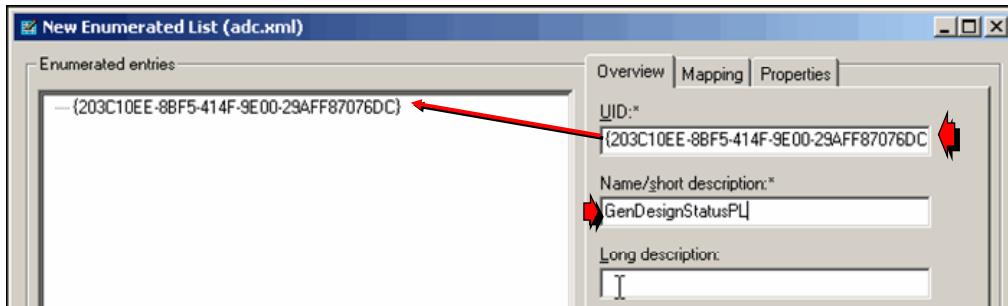


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter a **Short description** in the *New Enumerated List dialog*.



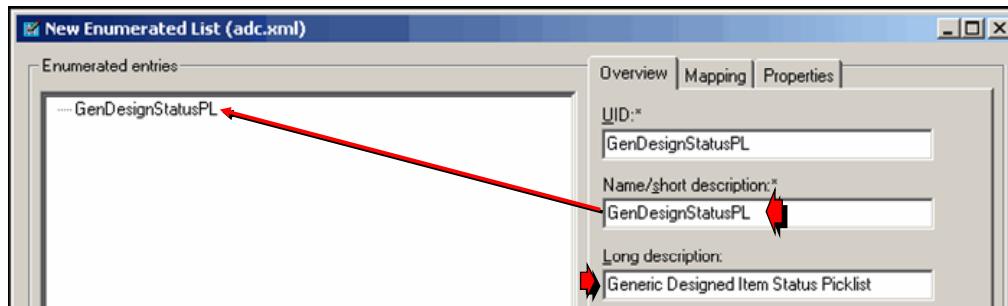
© 2005, Intergraph Corp.
All Rights Reserved.

Also enter a value in the UID field. The purpose for doing this will be explained and demonstrated in chapter 5.



Creating an Enumerated List Type

- The **Short description** will replace the default displayed UID in the *Enumerated entries* field.



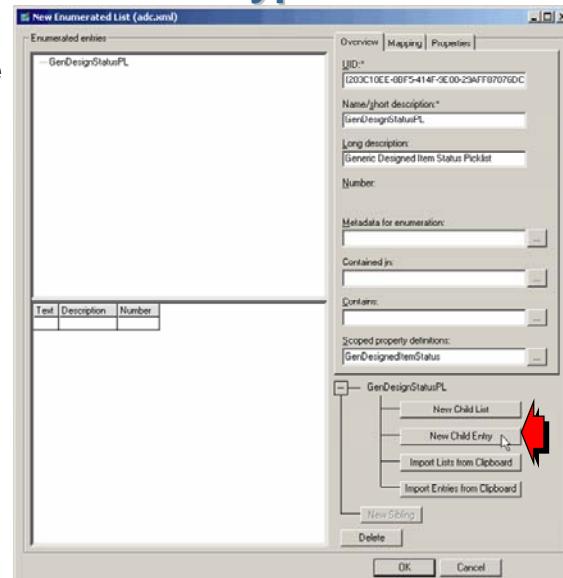
- Also enter a **Long description** for the new Enumerated List.

© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Click the **New Child Entry** button to add the initial enumerated entry.

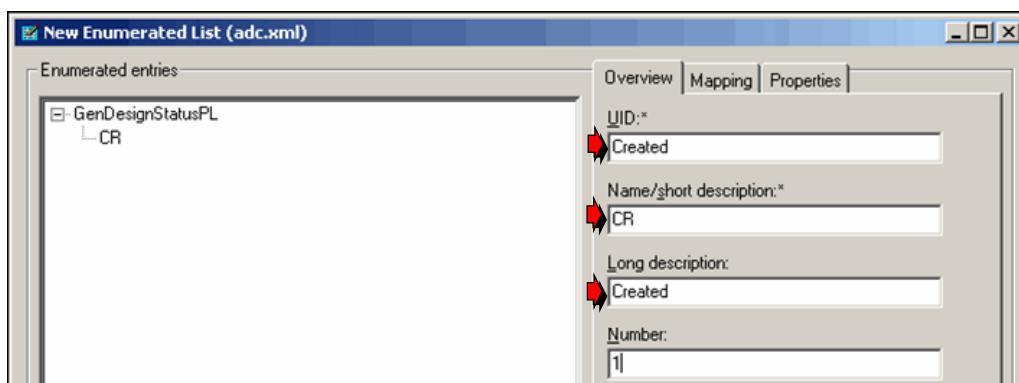


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter an optional **UID**, a **Short description** and an optional **Long Description** for the first enumerated entry in the enumerated list.

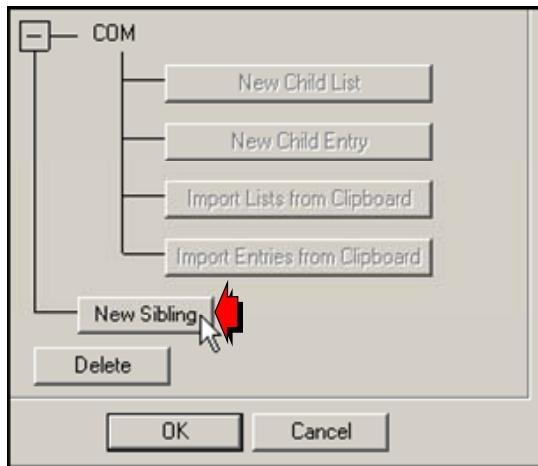


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Click the **New Sibling** button to add another enumerated entry.

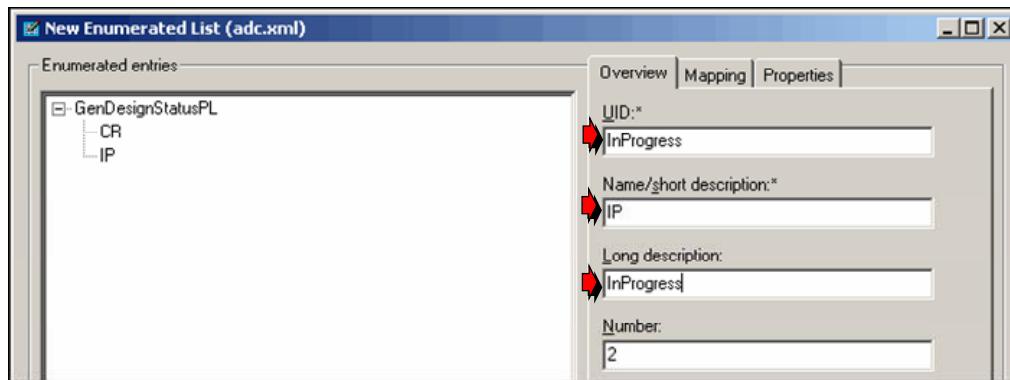


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter an optional **UID**, a **Short description** and an optional **Long Description** for the next enumerated entry in the enumerated list, then click the **New Sibling** button

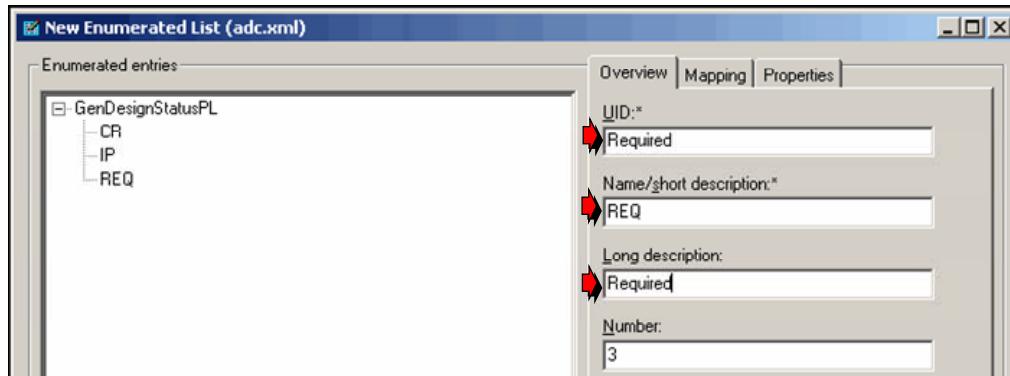


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter an optional **UID**, a **Short description** and an optional **Long Description** for the next enumerated entry in the enumerated list, then click the **New Sibling** button

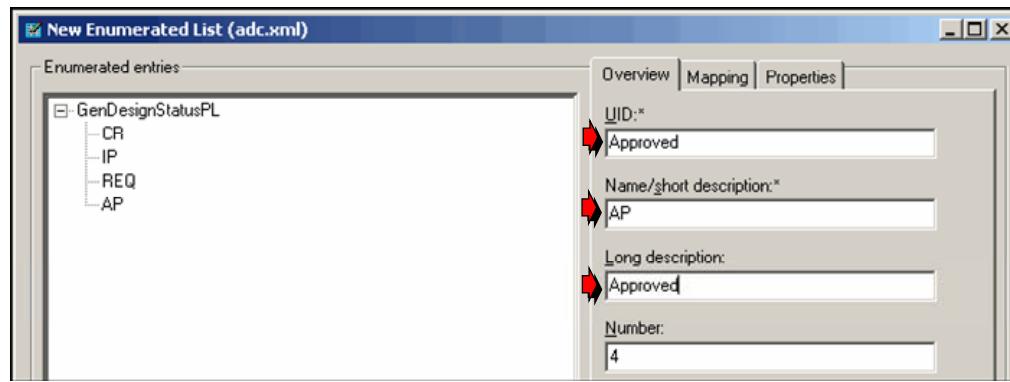


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter an optional **UID**, a **Short description** and an optional **Long Description** for the next enumerated entry in the enumerated list, then click the **New Sibling** button

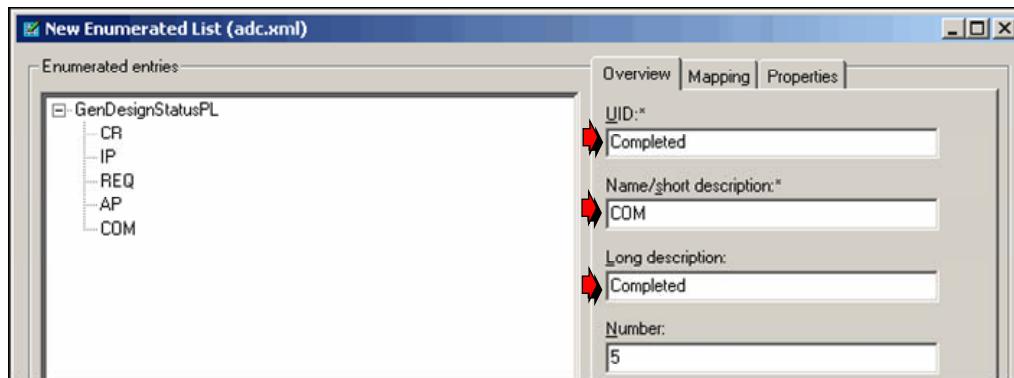


© 2005, Intergraph Corp.
All Rights Reserved.



Creating an Enumerated List Type

- Enter an optional **UID**, a **Short description** and an optional **Long Description** for the last enumerated entry in the enumerated list.

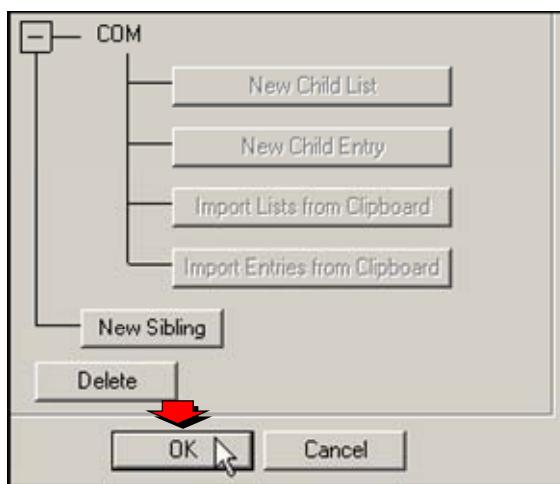


© 2005, Intergraph Corp.
All Rights Reserved.

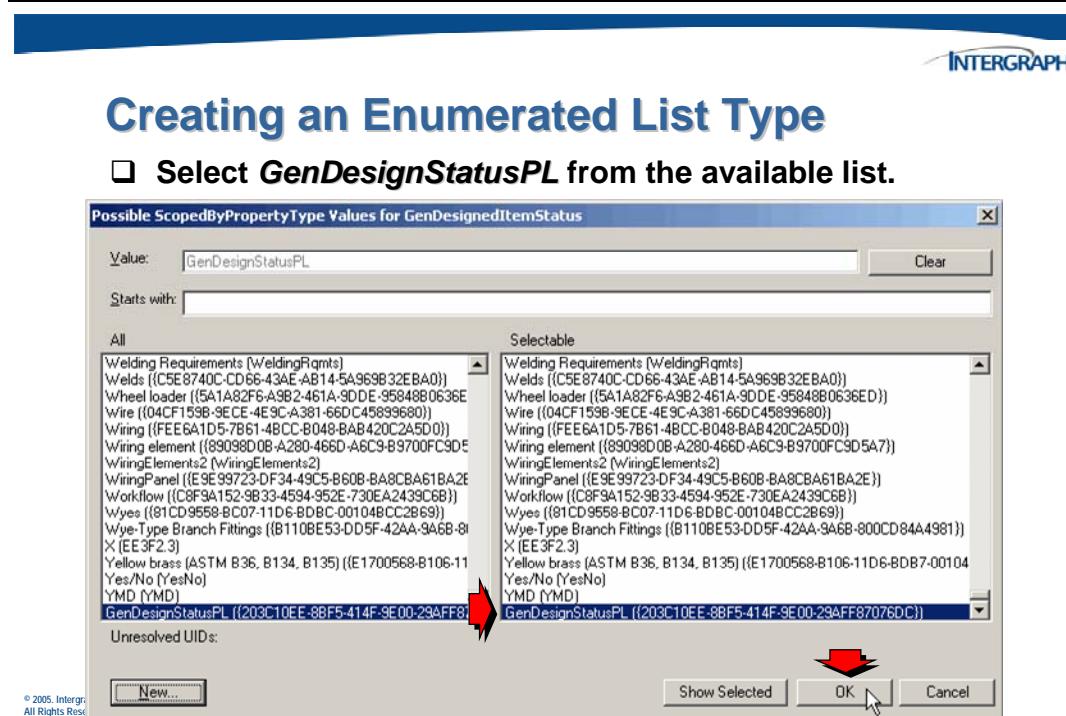
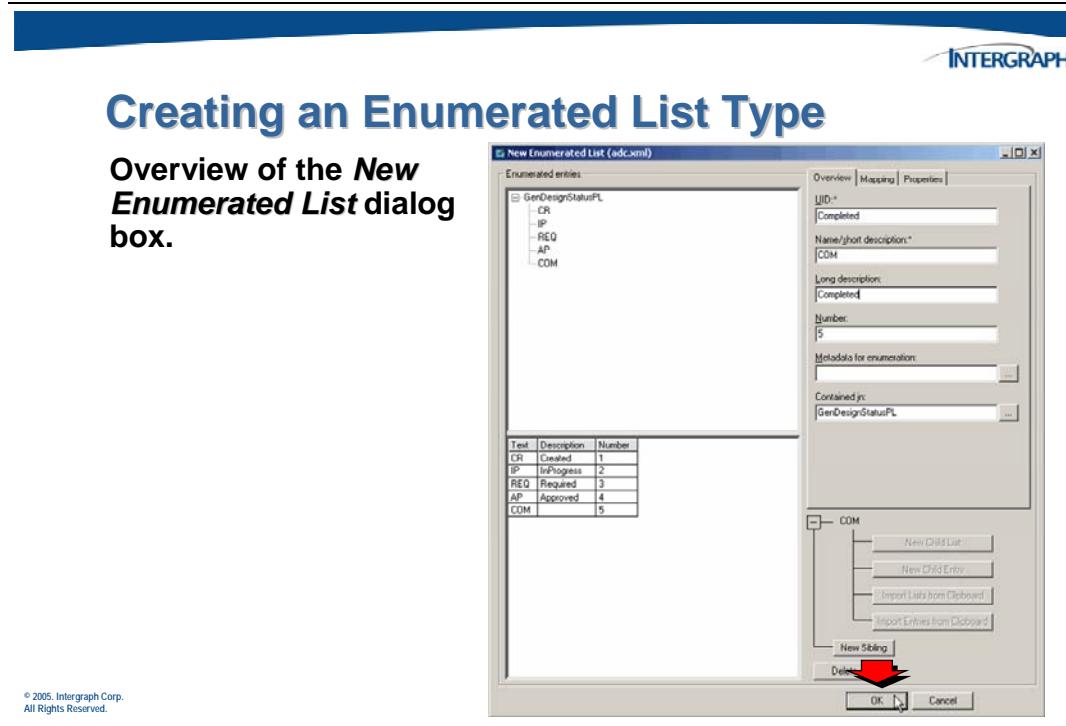


Creating an Enumerated List Type

- Click **OK** to save the changes to the enumerated list type and its enumerated entries.



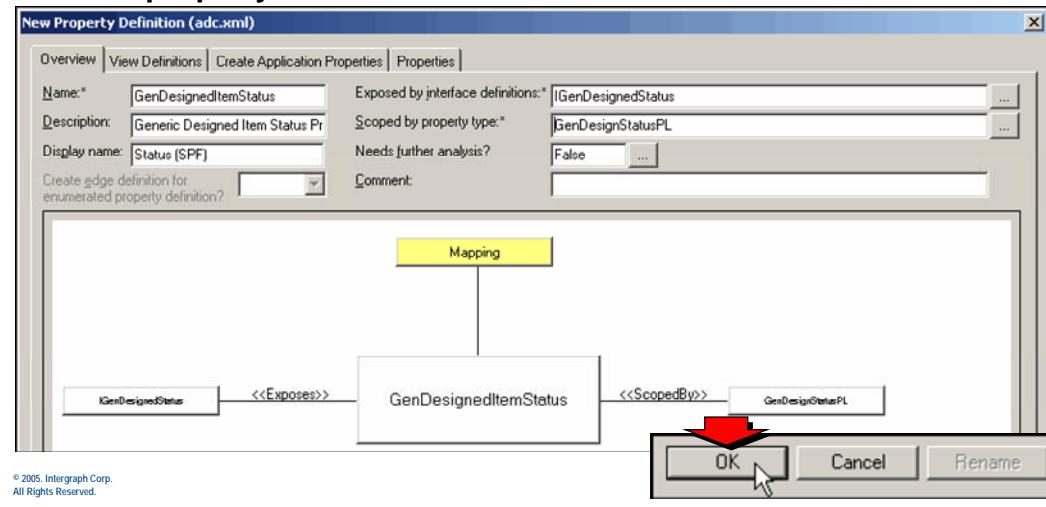
© 2005, Intergraph Corp.
All Rights Reserved.





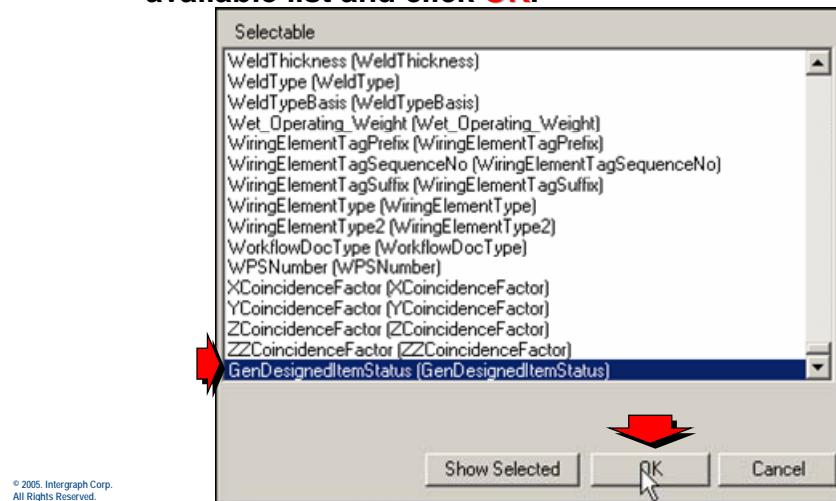
Creating an Enumerated Property

- Click **OK** to create the new **GenDesignedItemStatus** property.



Creating an Enumerated Property

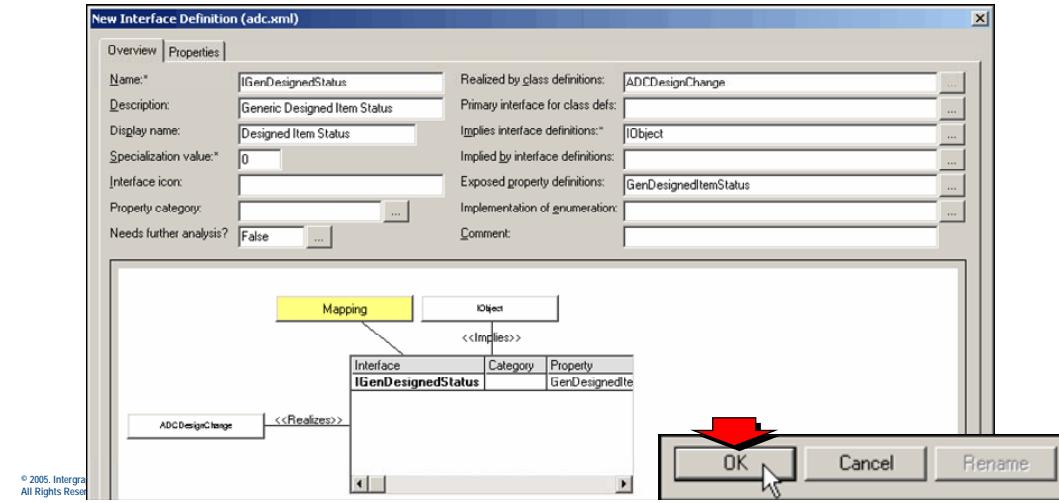
- Now you can select **GenDesignedItemStatus** from the available list and click **OK**.



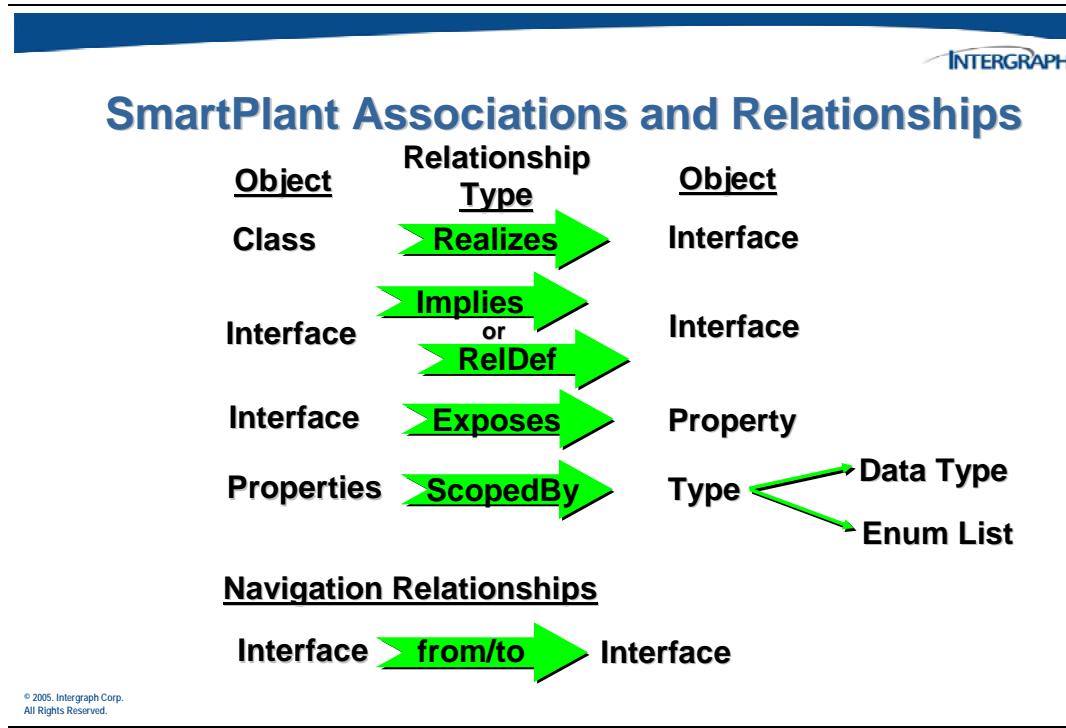


Creating an Enumerated Property

- Click **OK** to create the new *IGenDesignedStatus* interface definition.

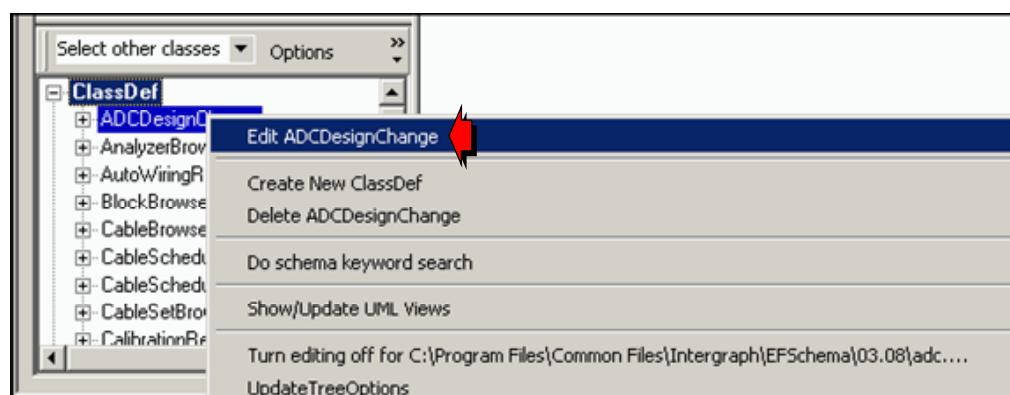


4.6 Interactive Activity – Creating a Realized Interface Definition



Create a Realized Interface Definition

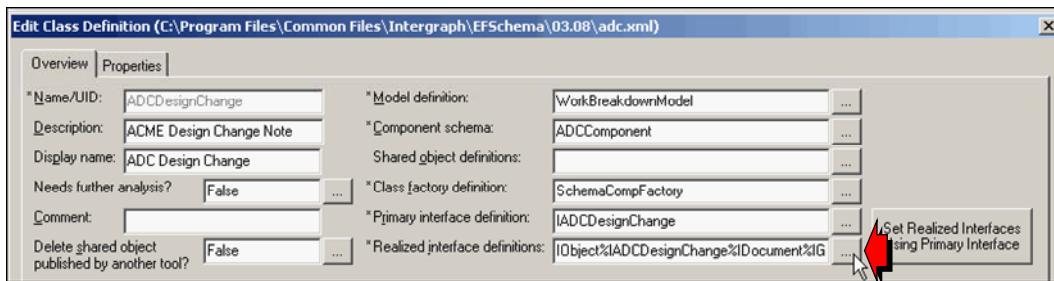
- Right-click on the **ADCDesignChange** class definition and select **Edit ADCDesignChange**.





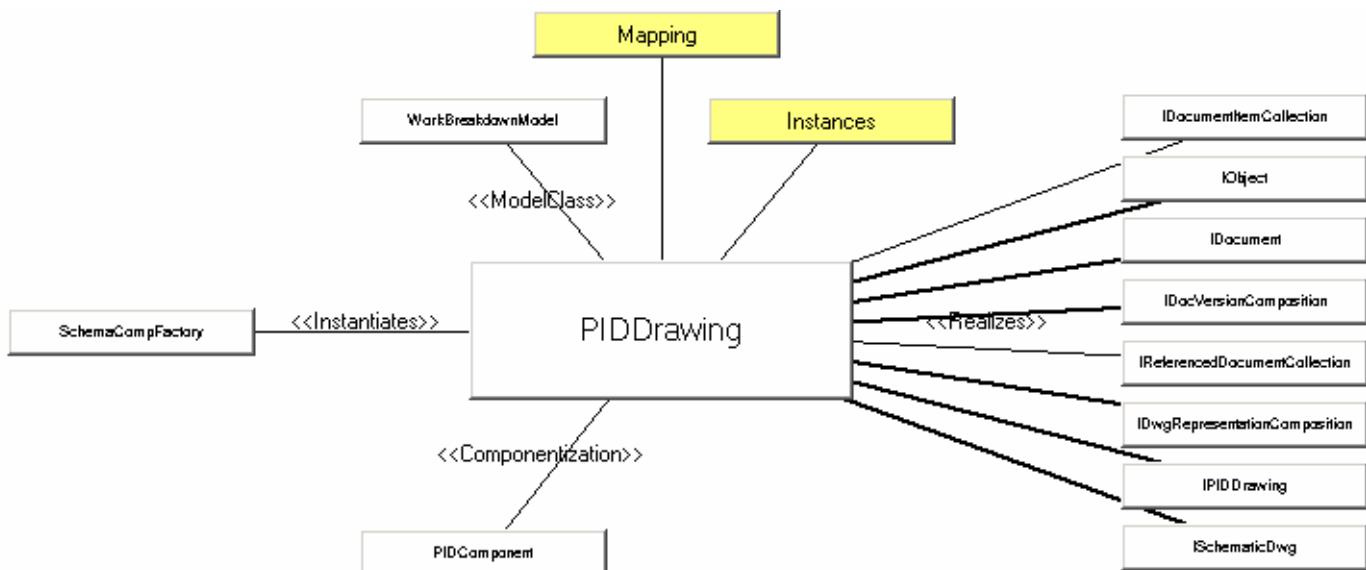
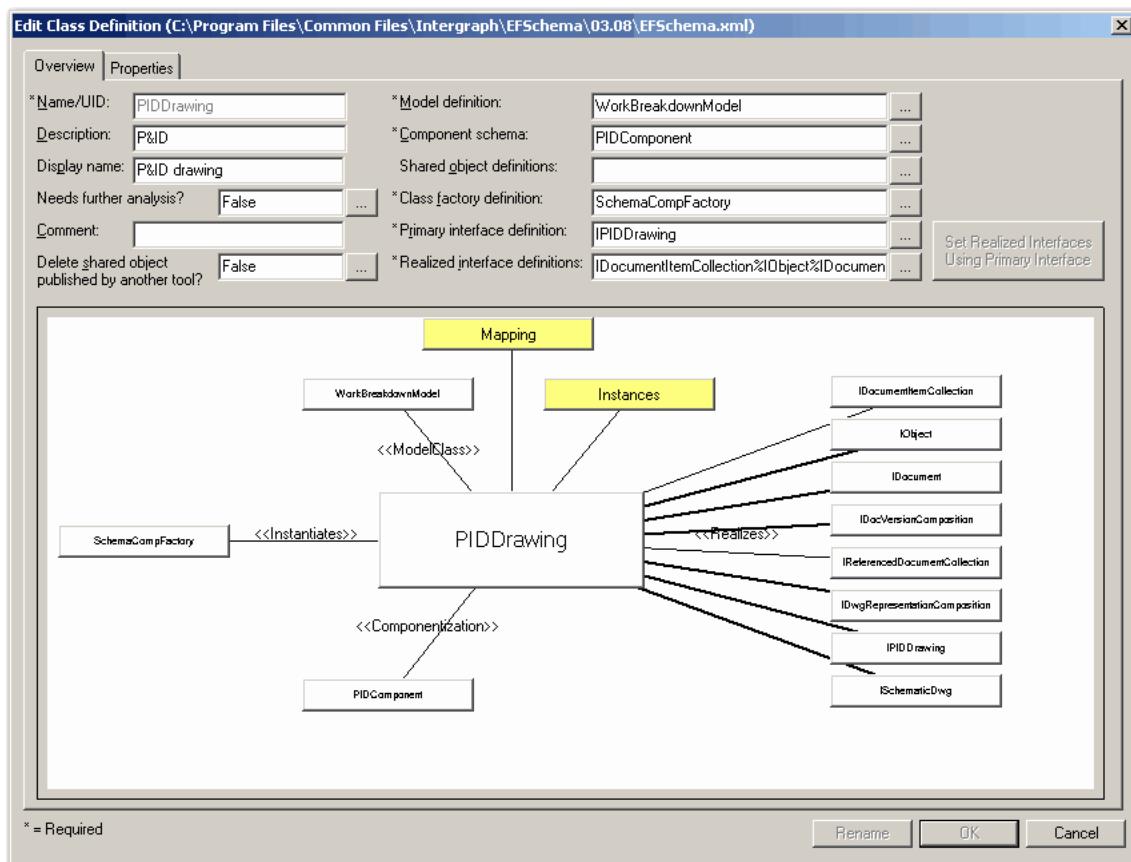
Create a Realized Interface Definition

- Choose the *Realized interface definitions* browse (...) button.

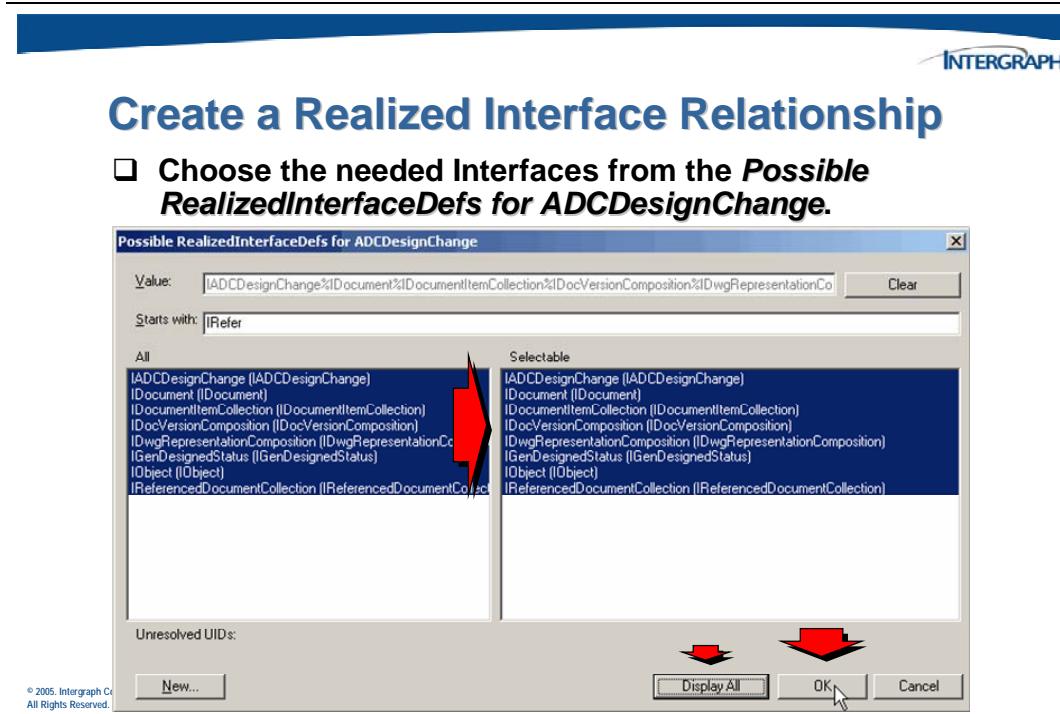


© 2005, Intergraph Corp.
All Rights Reserved.

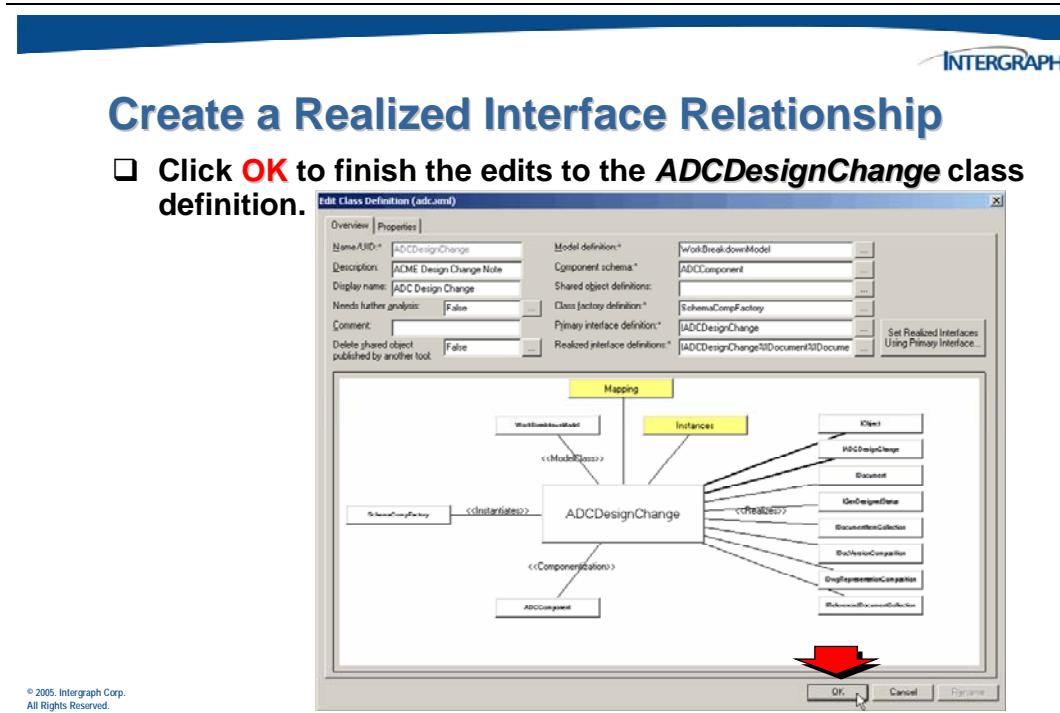
How do you know what other Interfaces will be needed for your new class? You can use an existing class as an example to achieve the same behavior for your new class. For example, you can review the structure for **PIDDrawing** as shown on the next page.



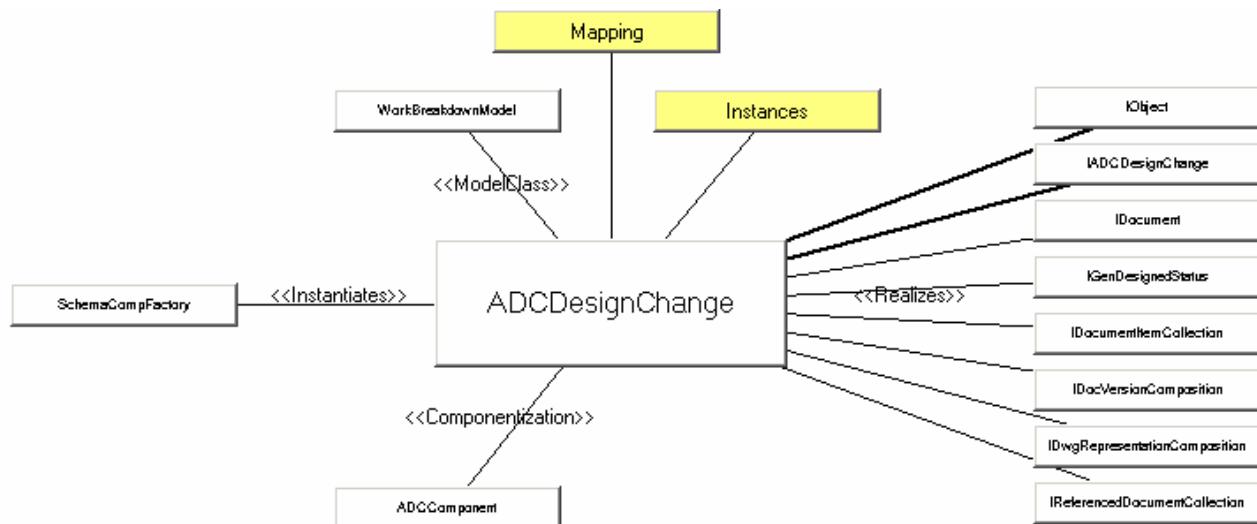
From the list on the left side, highlight the following interfaces; *IDocumentItemCollection*, *IDocVersionComposition*, *IDwgRepresentationComposition*, *IReferencedDocumentCollection*.



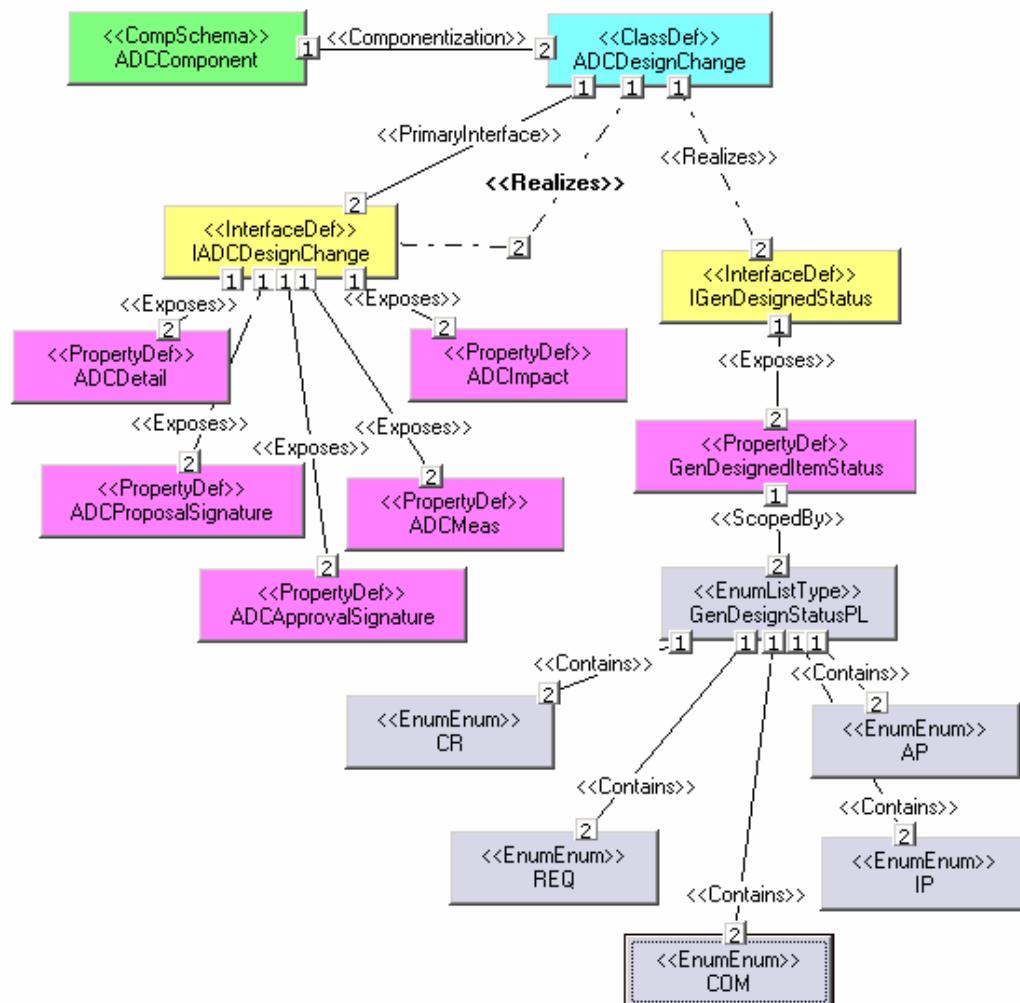
On the right side, click **Show Selected** to verify your choices.



ADC Design Change UML Diagram



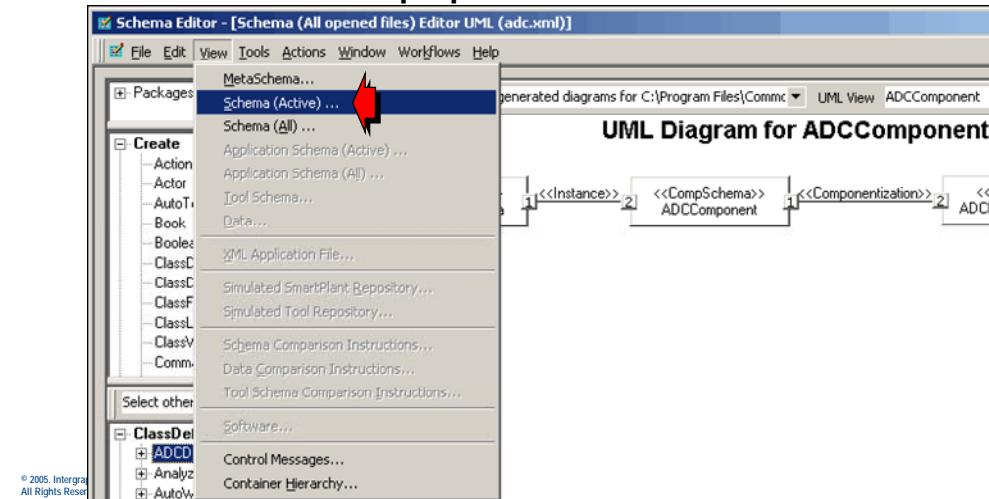
ADC Custom View





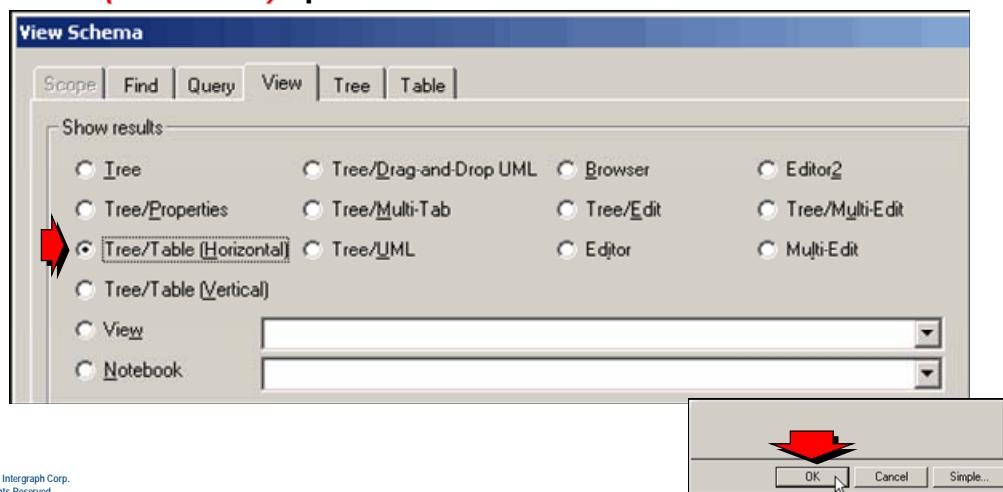
ClassDef/InterfaceDef Properties

- Click the **View> Schema (Active)...** menu command to view the InterfaceDef properties for a ClassDef.



ClassDef/InterfaceDef Properties

- In the **View Schema** dialog box, select the **Tree/Table (Horizontal)** option and click **OK**.





ClassDef/InterfaceDef Properties

The ADCDesignChange properties are displayed in the right pane in a horizontal table.

The screenshot shows the 'ClassDef' node expanded, with 'ADCDDesignChange' selected. The right pane displays a table of properties:

Class	Interface	Property	Type
ADCDDesignChange	IObject	UID	string128
	IADCDDesignChange	Name	
		Description	string
		ADCDetail	
		ADCM eas	FlowRateUoM
		ADCImpact	string
		ADCP roposalSignature	
		ADCA pprovalSignature	
	IDocument	DocCategory	DocCategories
		DocSubtype	DocSubtypes
		DocTitle	string
		DocType	DocTypes
	IGenDesignedStatus	GenDesignedItemStatus	GenDesignStatusPL
	IDocumentItemCollection		
	IDocVersionComposition		
	IDwgRepresentationComposition		
	IRelatedDocumentCollection		

Red arrows point from the 'Name' and 'Description' fields in the left pane to the 'Name' and 'Description' properties in the table. Red boxes highlight the 'IADCDDesignChange' interface and the 'IDocument' interface. A red arrow points from the 'Design Status' field in the left pane to the 'IGenDesignedStatus' property in the table.

All Rights Reserved.



ClassDef/InterfaceDef Properties

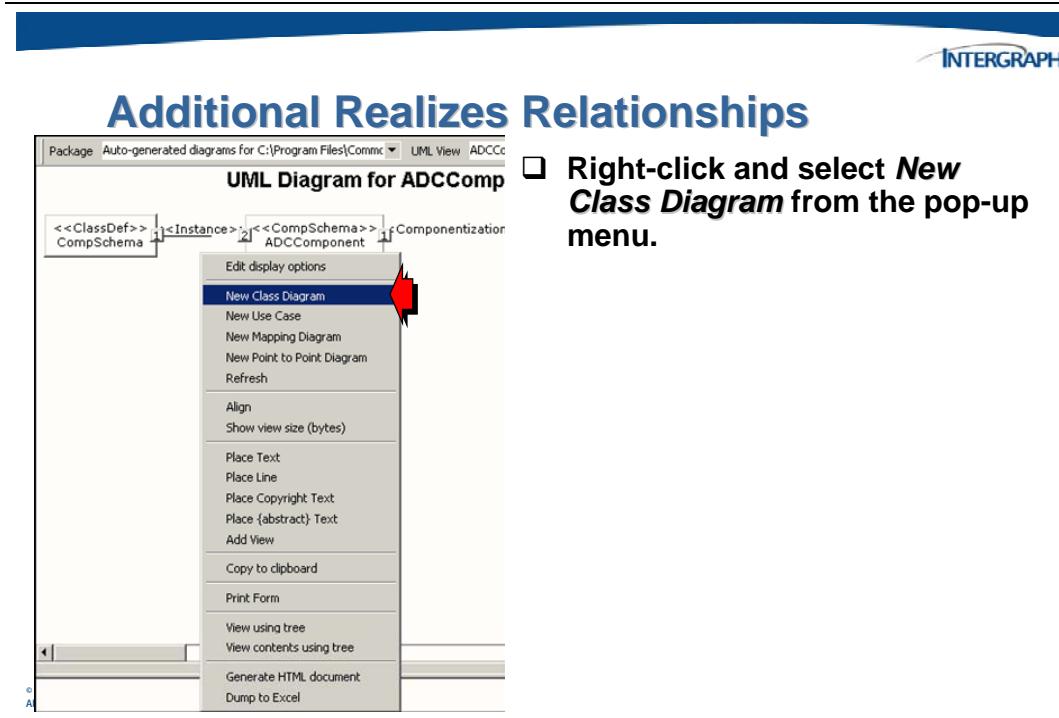
- Click the X to **Close** the Schema Tree/Table window.

The screenshot shows the 'ClassDef' node expanded, with 'ADCDDesignChange' selected. The right pane displays a table of properties. A red arrow points to the close button in the top right corner of the window frame.

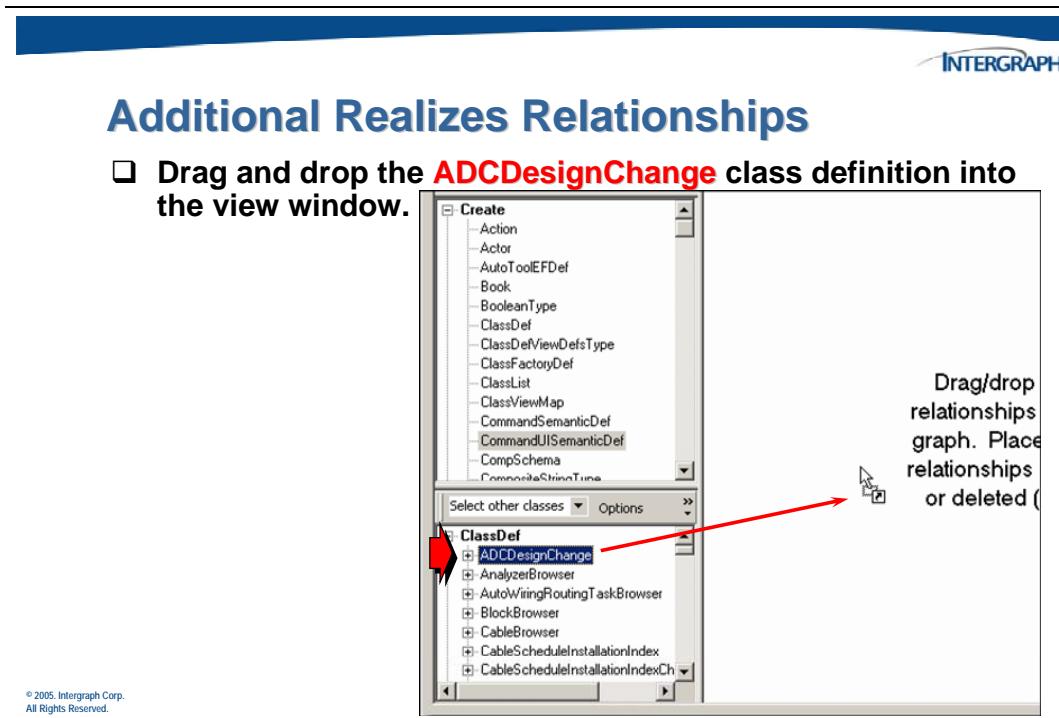
Class	Interface	Property	Type
ADCDDesignChange	IObject	UID	string128
	IADCDDesignChange	Name	
		Description	string
		ADCDetail	
		ADCM eas	FlowRateUoM
		ADCImpact	string
		ADCP roposalSignature	
		ADCA pprovalSignature	
	IDocument	DocCategory	DocCategories
		DocSubtype	DocSubtypes
		DocTitle	string
		DocType	DocTypes
	IGenDesignedStatus	GenDesignedItemStatus	GenDesignStatusPL
	IDocumentItemCollection		
	IDocVersionComposition		
	IDwgRepresentationComposition		
	IRelatedDocumentCollection		

© 2005, Intergraph Corp.
All Rights Reserved.

We are now going to add some additional relationships in preparation for defining *Edge Definitions*, *Graph Definitions*, *View Definitions* and *Class View Mappings*, which will be discussed in chapter 5.



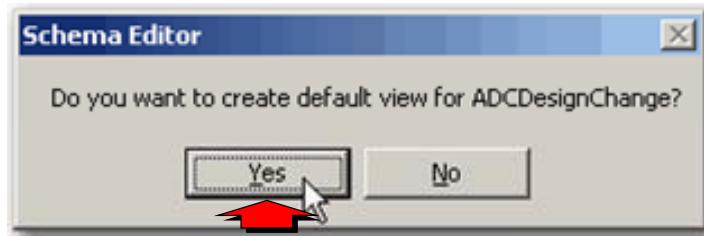
View the new ADCDesignChange class as a UML diagram in order to add the new necessary relationships.





Additional Realizes Relationships

- Click **Yes** to create a default UML view.



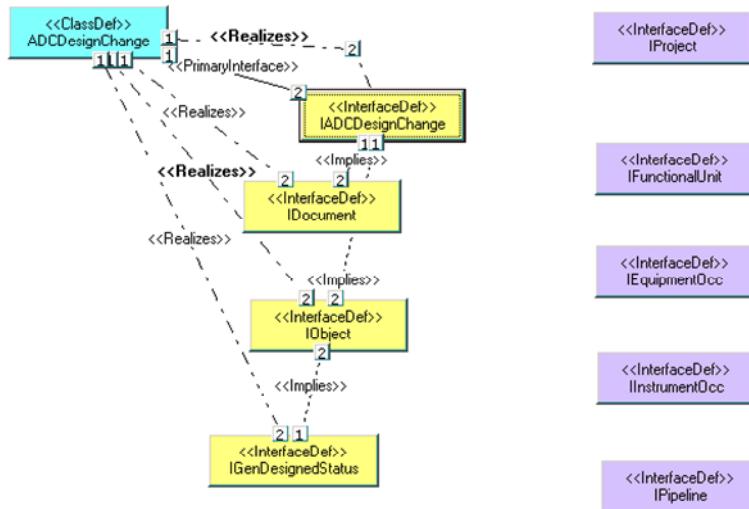
© 2005, Intergraph Corp.
All Rights Reserved.

Add a new *ADCDesignChange* realizes **IWBSItem** relationship. We will be adding the *Design Change* into the work breakdown structure, *Plant->Project->DesignChange*. Contract is also part of the WBS and you just as easily group the design changes by contract. Realizing **IWBSItem** gives you the **PBSWBS RelDef** which we will utilize in follow on steps.



Additional Realizes Relationships

Why create additional relationships?

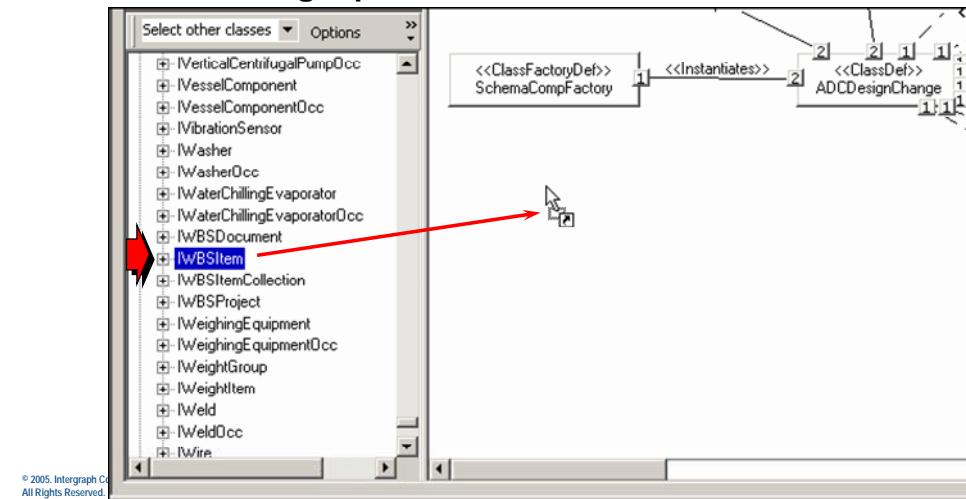


© 2005, Intergraph Corp.
All Rights Reserved.



Additional Realizes Relationships

- Drag and drop the **IWBItem** interface from the tree view into the right pane.

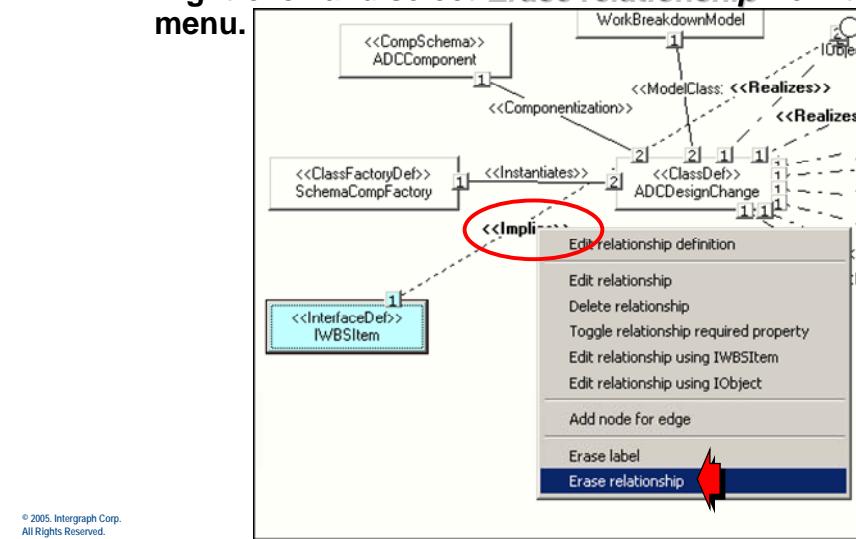


Remove some of the relationships to make the UML easier to view.

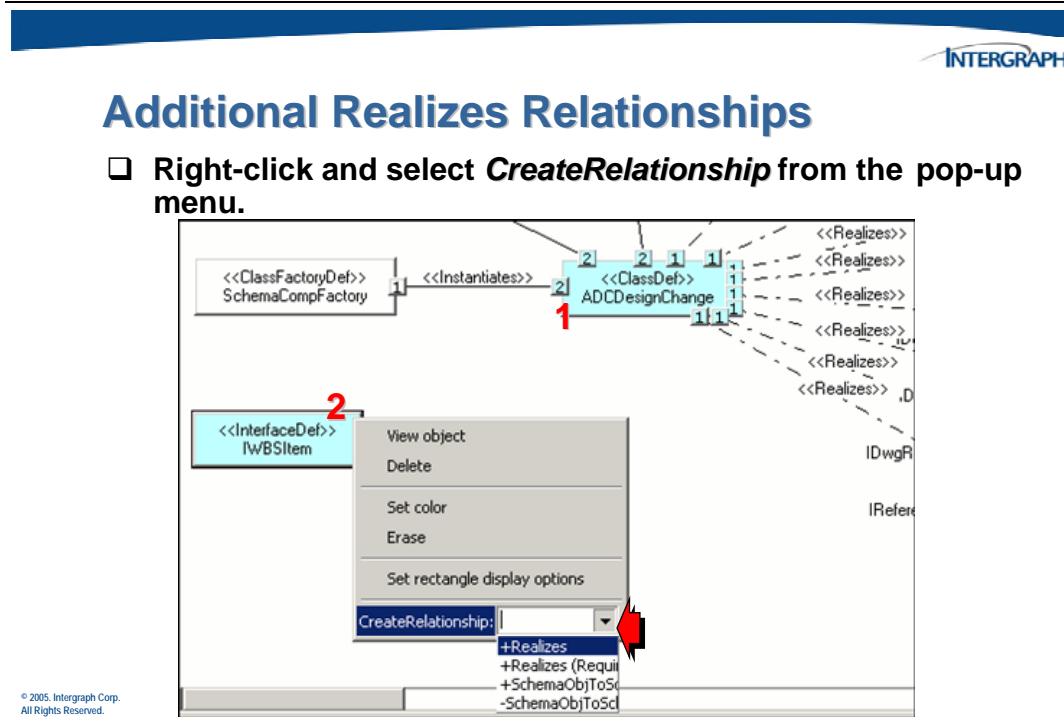


Additional Realizes Relationships

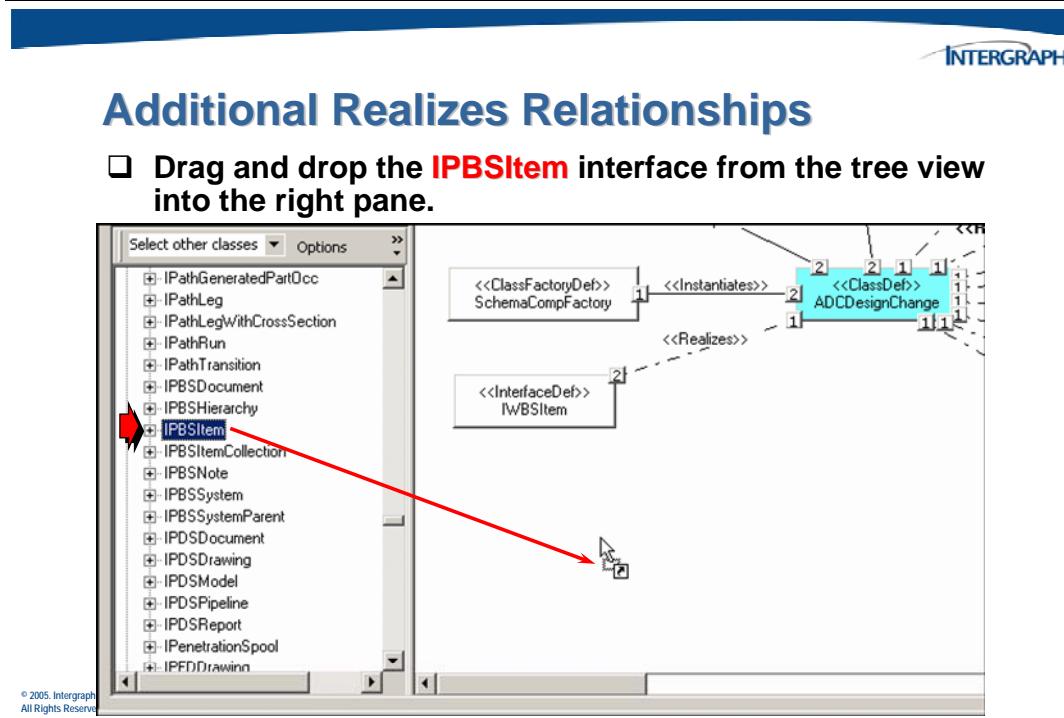
- Right-click and select **Erase relationship** from the pop-up menu.



Click on **ADCDesignChange**, hold the control key down and click on **IWBSSItem** so that both are highlighted.

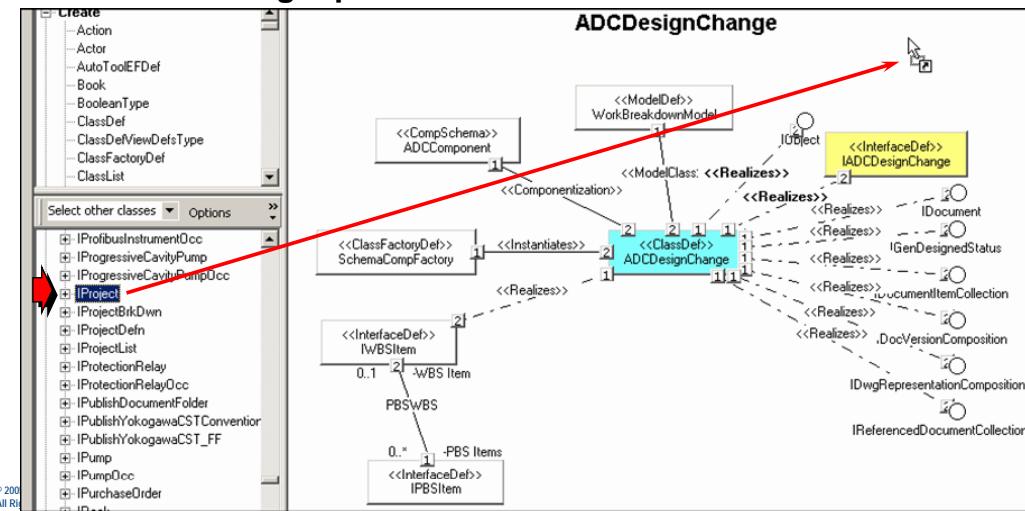


Add the other interfaces we will need.



Additional Realizes Relationships

- Drag and drop the **IProject** interface from the tree view into the right pane.



4.7 Relationship Definitions

All relationship definitions in the schema are of class ***RelDef***, which is part of the meta schema. A RelDef 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.

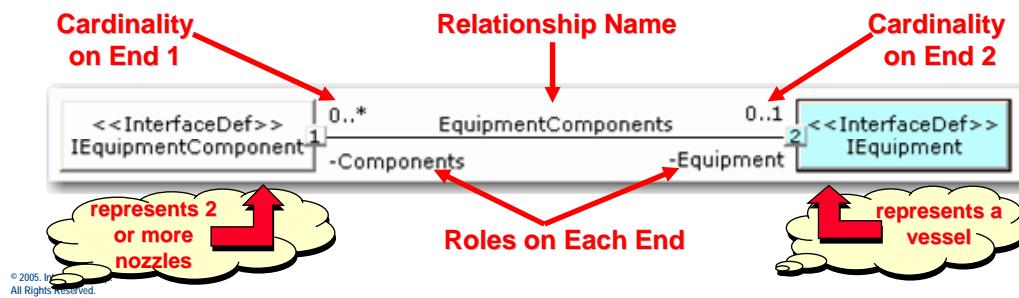


Relationship Definitions

Relationship definitions (**RelDef**) are associations between interface definitions. A RelDef will define two object instances that fulfill the roles on each end of the relationship.

Relationship definitions also define cardinalities on each end of the relationship.

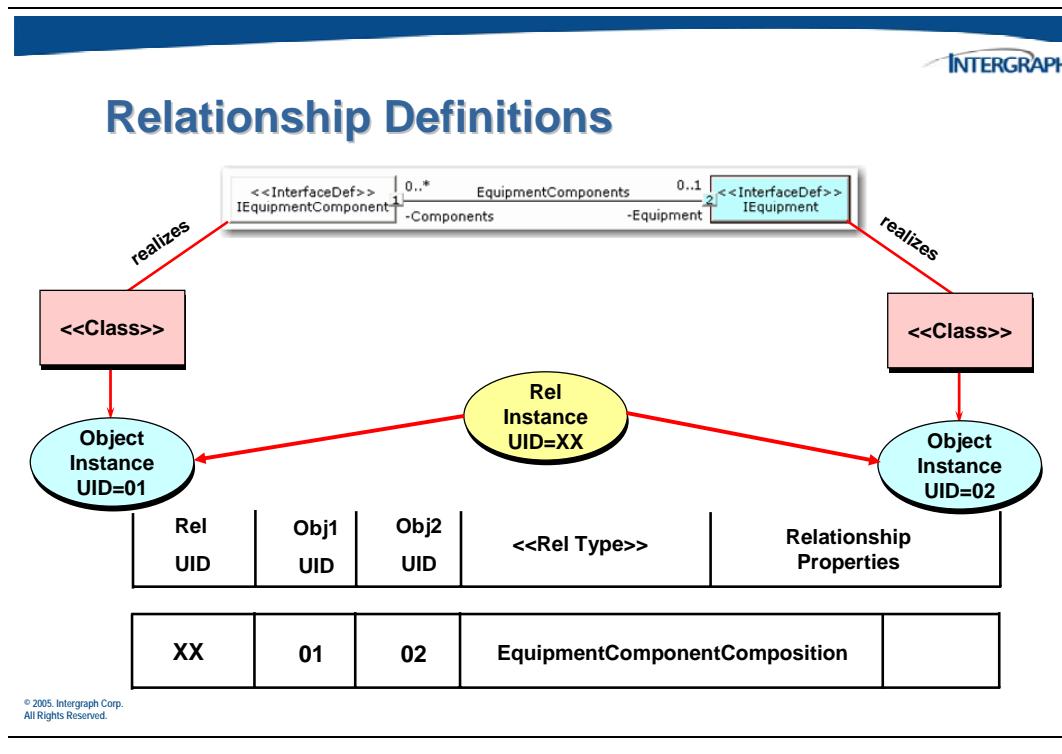
The following is an example of a *relationship* showing equipment components that can be associated with equipment e.g. nozzles on a vessel.



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. For example there could be many nozzles and some may be associated to a vessel but that nozzle instance would not be connected to more than one (i.e. multiple) vessel.

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.

When something is published, a Rel (relationship) is an instantiation of a RelDef.



Relationship Definitions

Engineering tools can publish objects at different times.

Users can publish nozzles ahead of equipment.

The software has to decide “can this relationship be resolved”? An example would be P&ID could publish nozzles first and then vessels are published later. The software would suppress errors because there is no harm in doing this since the vessels CAN be published at a later time.

When defining a relationship definition, **End 1** and **End 2** interfaces depends on which way users are going to query.

The locality of reference, which is shown as part of the role name, shows how this relationship will be instantiated:

- “I’ve got to publish”
- “You have to publish”
- “Someone will publish someday”



Relationship Definitions



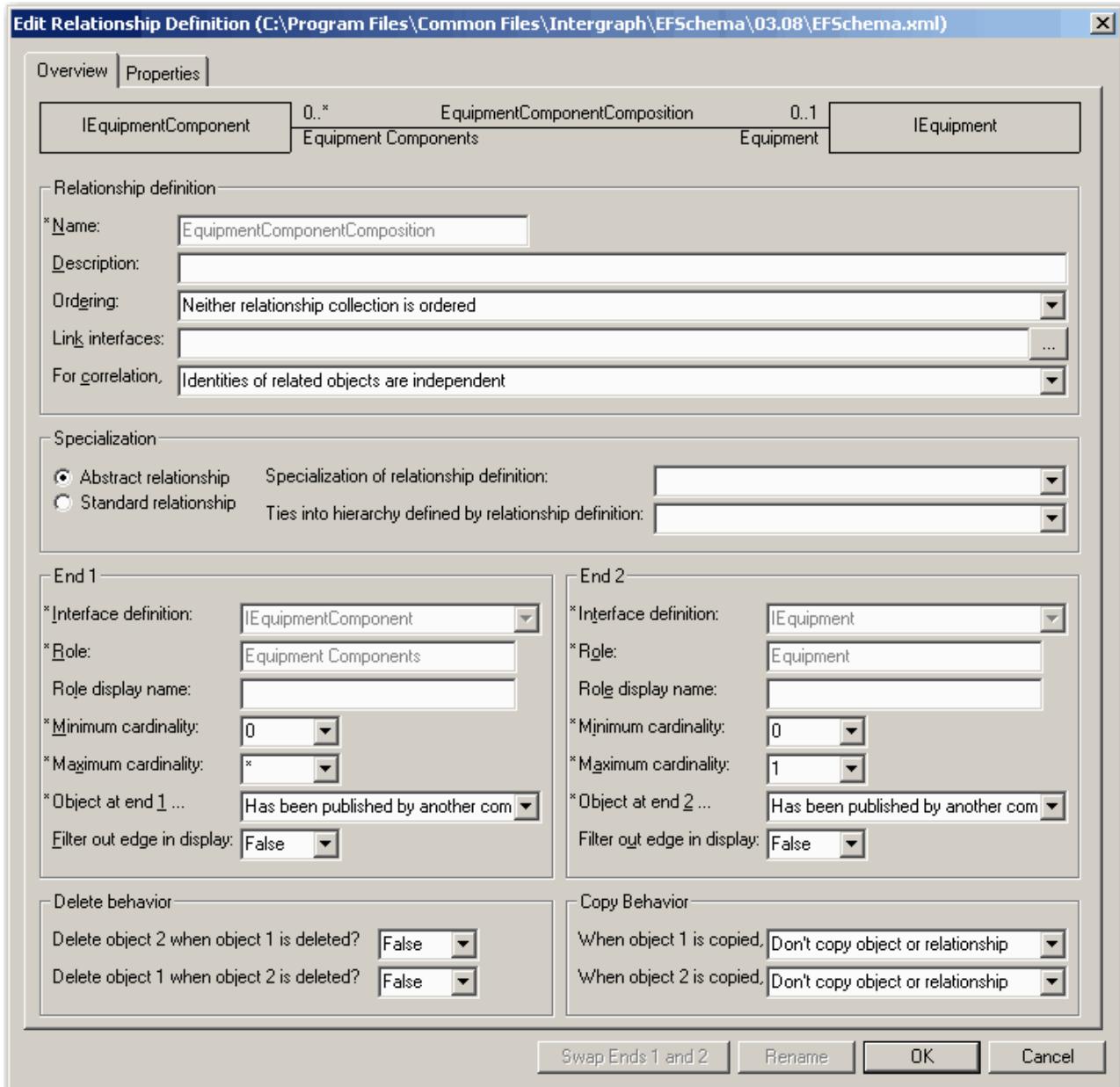
Locality of Reference

- has been published by another component
- + must be in the same container as object 2
- # must be published by the same tool

4.7.1 Properties of Relationship Definitions

When you create or edit a relationship definition in the Schema Editor, you can define the following options:

- Name** – Specifies the name of the relationship definition.
- Description** – Provides a more detailed description of the relationship definition.



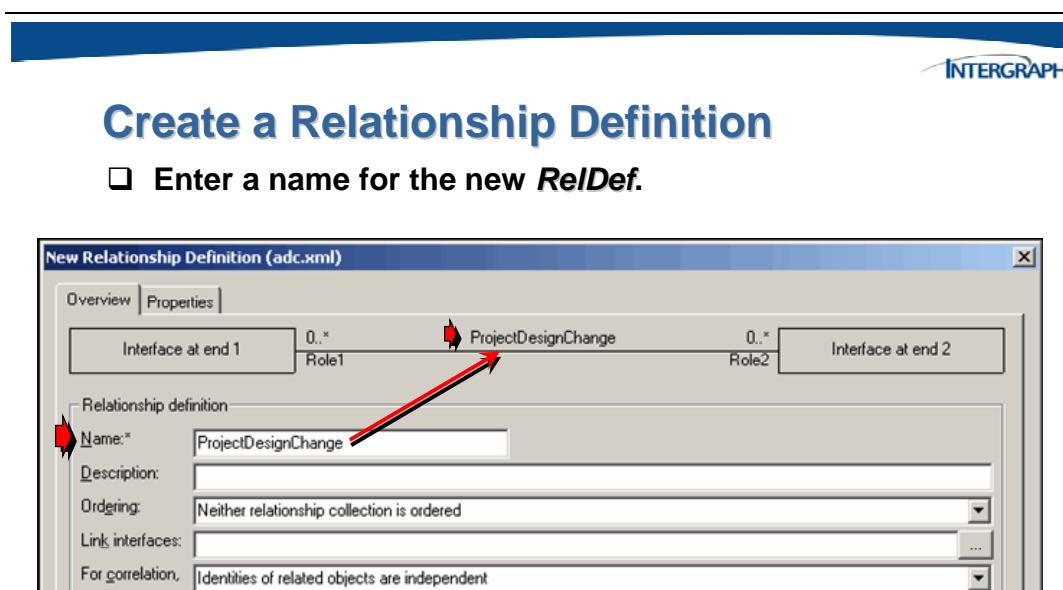
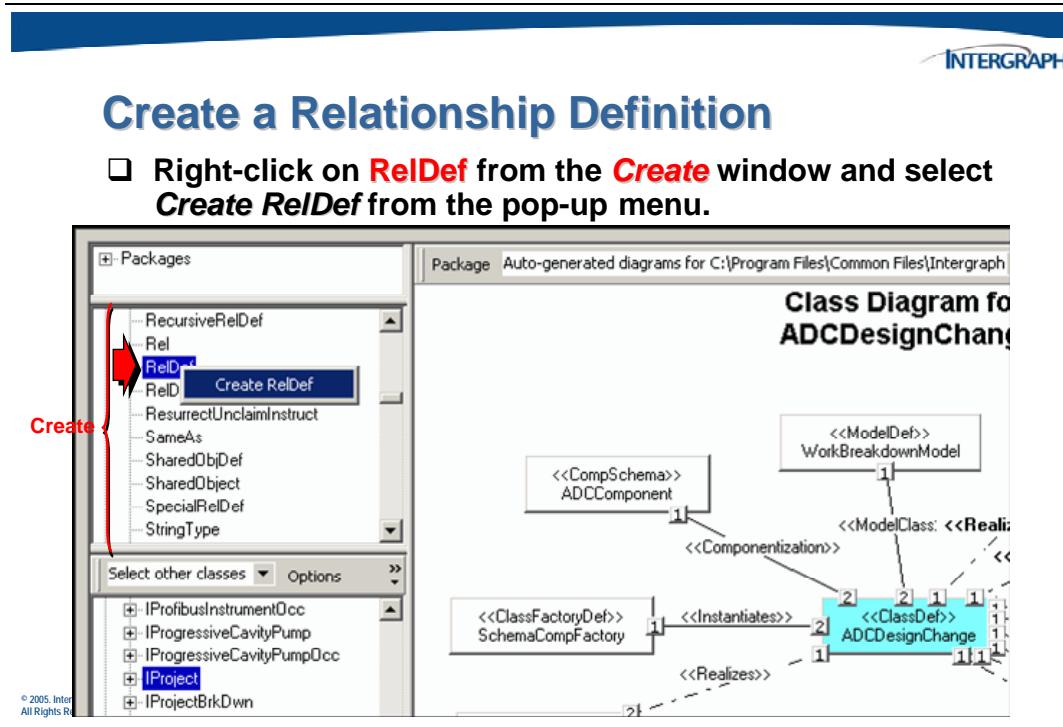
- Ordering** – Specifies whether the relationship is ordered (optional). If objects in a relationship are ordered, they can be ordered from end 1 to end 2 or from end 2 to end 1, but not both. Ordering relationships is important when the

sequence of the related objects is important. For example, the order of the piping ports for a piping connector is important in understanding the logical connectivity. Conditional ordering is not currently supported in SmartPlant.

- Abstract relationship** – Indicates that the relationship definition is abstract. An abstract relationship cannot be directly instantiated (no Rel objects can use the abstract relationship definition). Abstract relationship definitions are used for generic traversal operations. An abstract relationship definition must be specialized by one or more concrete (standard) relationship definitions. (Note: **Do NOT** publish abstract relationships.)
- Standard relationship** – Indicates that the relationship definition is a standard relationship. Standard relationships are also called concrete relationships. Standard or concrete relationship definitions can be used for instantiated relationships. A relationship definition can be either abstract or standard, but not both.
- Specialization of abstract relationship definition** – Specifies the name of the abstract relationship definition of which this standard relationship definition is a specialization. Any standard (concrete) relationship that is a specialization of an abstract relationship is also treated as an instance of that abstract relationship. For example, the PipingEnd1Conn, PipingEnd2Conn, and PipingTapOrFitting concrete relationship definitions are all specializations of the Connections abstract relationship definition. Therefore, a connection at end 1 is a PipingEnd1Conn relationship and also a Connections relationship (one Rel that is a specialization). A connection at end 2 is a PipingEnd2Conn and also a Connections relationship; a tap/fitting connection is a PipingTapOrFitting relationship and also a Connections relationship.
- End 1**
 - **Interface definition** – Identifies the interface definition at end 1 of the relationship definition.
 - **Role** – Specifies the name of the role for end 1 of the relationship.
 - **Minimum cardinality** – Specifies the minimum number of participants allowed on end 1 of the relationship. Possible values are 0, 1, and 2.
 - **Maximum cardinality** – Specifies the maximum number of participants allowed on end 1 of the relationship. Possible values are 1, 2, or * (many).
 - **Object at end 1** – Specifies whether hanging relationships are allowed for the object at end 1 of the relationship. Objects can be required to be located in the same container (typically a document) as the other object in the relationship, to be published by the same authoring tool as the other object, or to be published by another component.
 - **Filter out edge in display** – Specifies whether you want to see the relationship edge automatically created from end 1 to end 2 of this relationship in the Schema Editor user interface. If this option is set to **True**, user access for this edge definition can be controlled in the SPF client by user group security.
- End 2**

- **Interface definition** – Identifies the interface definition at end 2 of the relationship definition.
 - **Role** – Specifies the name of the role for end 2 of the relationship.
 - **Minimum cardinality** – Specifies the minimum number of participants allowed on end 2 of the relationship. Possible values are 0, 1, and 2.
 - **Maximum cardinality** – Specifies the maximum number of participants allowed on end 2 of the relationship. Possible values are 1, 2, or * (many).
 - **Object at end 2** – Specifies whether hanging relationships are allowed for the object at end 2 of the relationship. Objects can be required to be located in the same container (typically a document) as the other object in the relationship, to be published by the same authoring tool as the other object, or to be published by another component.
 - **Filter out edge in display** – Specifies whether you want to see the relationship edge automatically created from end 2 to end 1 of this relationship in the Schema Editor user interface. If this option is set to **True**, user access for this edge definition can be controlled in the SPF client by user group security.
- Delete object 2 when object 1 is deleted?** – Indicates whether you want the software to delete the object on end 2 of the relationship when the object on end 1 is deleted.
- Delete object 1 when object 2 is deleted?** – Indicates whether you want the software to delete the object on end 1 of the relationship when the object on end 2 is deleted.
- When object 1 is copied** – Specifies what you want the software to do when the object on end 1 of the relationship is copied. You can choose whether to copy the object on end 2 as well, copy the object and relationship, leaving the relationship pointing to the same object that the first relationship pointed to, or not to copy the object or relationship at all.
- When object 2 is copied** – Specifies what you want the software to do when the object on end 2 of the relationship is copied. You can choose whether to copy the object on end 1 as well, copy the object and relationship, leaving the relationship pointing to the same object that the first relationship pointed to, or not to copy the object or relationship at all.

Add the **ProjectDesignChange** RelDef between *IADCDesignChange* and *IProject*. The WBSItemCollection RelDef is abstract so we create a specialized RelDef. You can see examples of this in the PlantProject and ProjectContract RelDefs.





Create a Relationship Definition

- Select the Interface names and enter a **Role** for both **End 1** and **End 2** for the new *RelDef*.

The dialog box shows the following configuration:

- Specialization:**
 - Abstract relationship
 - Standard relationship
- Specialization of relationship definition:** [Empty dropdown]
- Ties into hierarchy defined by relationship definition:** [Empty dropdown]
- End 1:**
 - Interface definition: IADCDesignChange
 - Role: DesignChange
 - Role display name: [Empty]
 - Minimum cardinality: 0
 - Maximum cardinality: *
 - Object at end 1 ... : Has been published by another com
 - Filter out edge in display: False
- End 2:**
 - Interface definition: IPProject
 - Role: Project
 - Role display name: [Empty]
 - Minimum cardinality: 1
 - Maximum cardinality: *
 - Object at end 2 ... : Has been published by another com
 - Filter out edge in display: False

© 2005, Intergraph Corp.
All Rights Reserved.



Create a Relationship Definition

- Select OK to create the new *RelDef*.

The dialog box shows the following configuration:

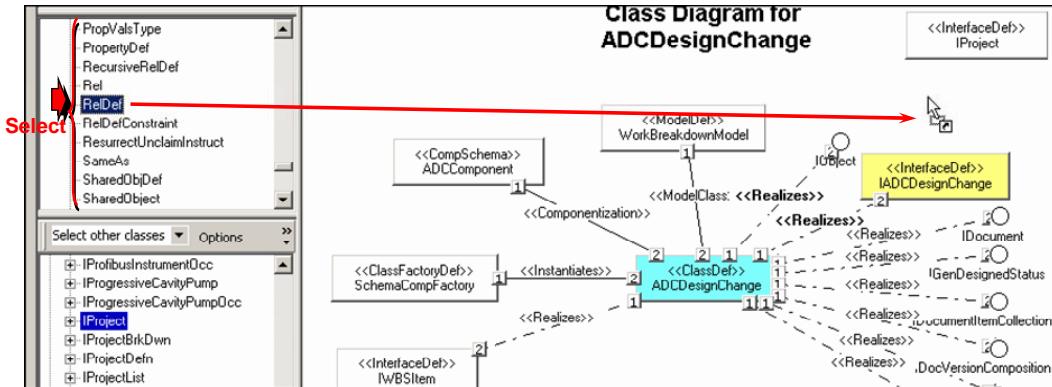
- Delete behavior:**
 - Delete object 2 when object 1 is deleted: False
 - Delete object 1 when object 2 is deleted: False
- Copy Behavior:**
 - When object 1 is copied: Don't copy object or relationship
 - When object 2 is copied: Don't copy object or relationship
- Buttons:**
 - Swap Ends 1 and 2
 - OK (highlighted with a red arrow)
 - Cancel
 - Rename

© 2005, Intergraph Corp.
All Rights Reserved.



Create a Relationship Definition

- Drag and drop RelDef from the Select tree view into the right pane.

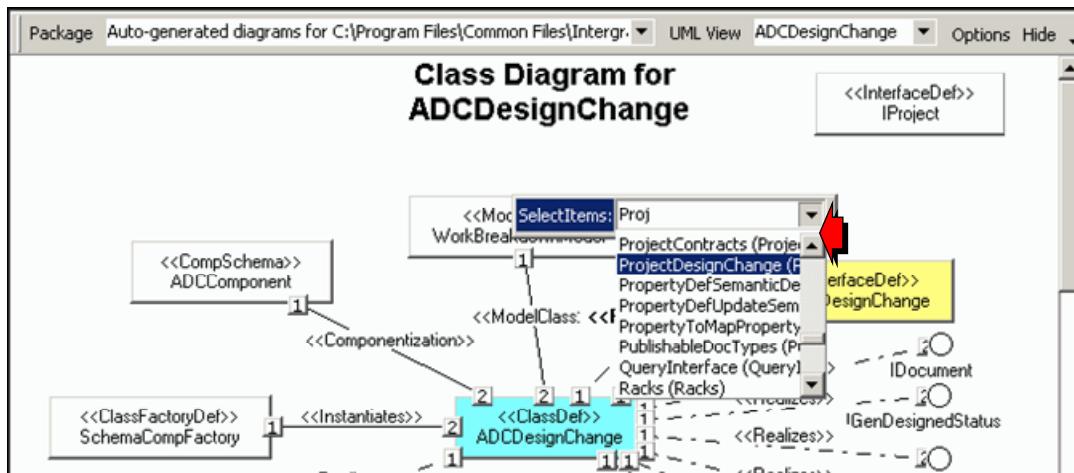


© 2005, Intergraph Corp.
All Rights Reserved.



Create a Relationship Definition

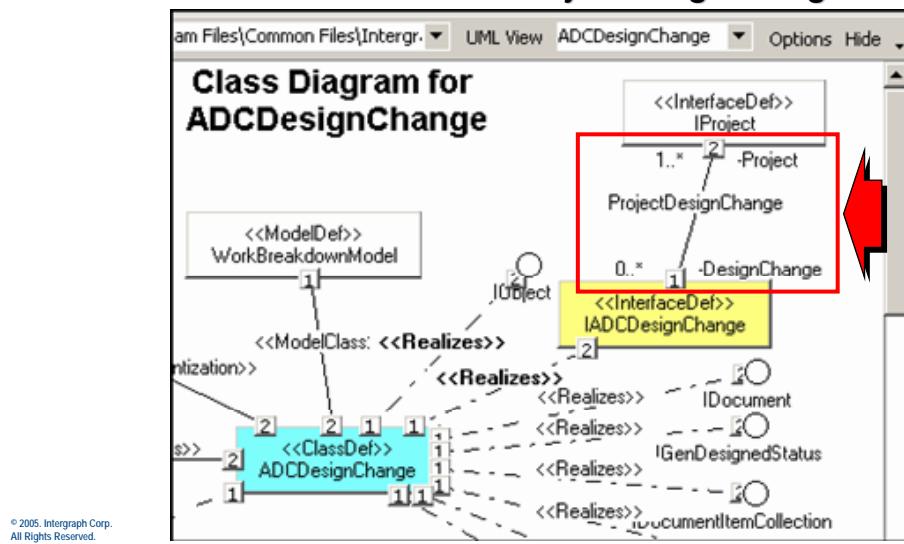
- Select the ProjectDesignChange RelDef from the pop-up box.



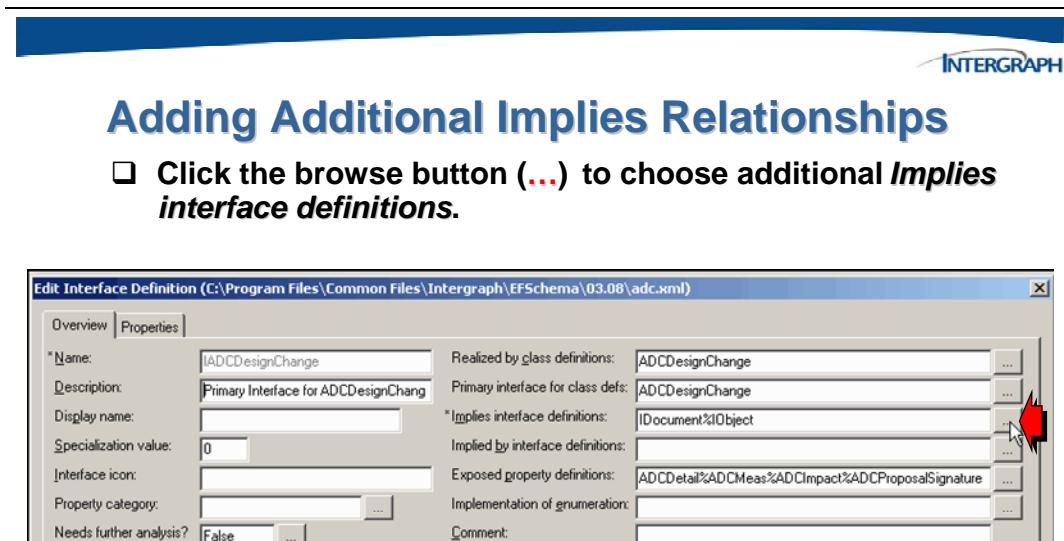
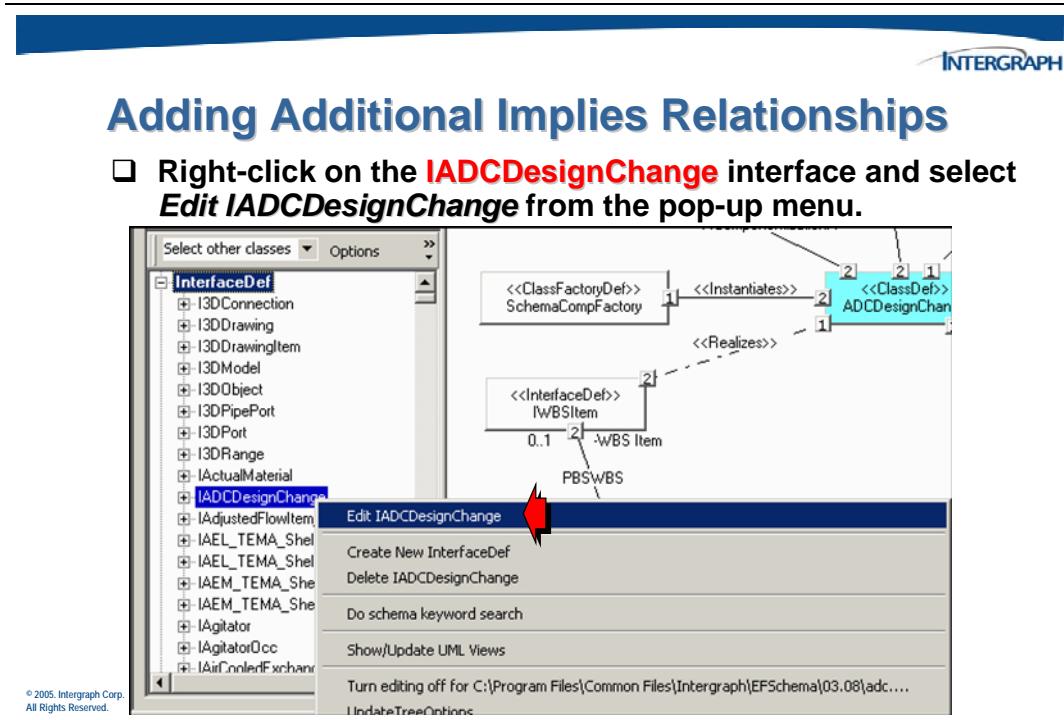
© 2005, Intergraph Corp.
All Rights Reserved.

ADCComponent UML Diagram

Zoomed in view of the new *ProjectDesignChange* RelDef.



Add the **IADCDesignChange** implies **IDocVersionComposition**, **IReferencedDocumentCollection**, and **IPBSItemCollection** relationships. The primary interfaceDef should imply everything. In this case, it is needed to properly create the *EdgeDefs* and *GraphDefs*.





Adding Additional Implies Relationships

Add the additional needed relationships.

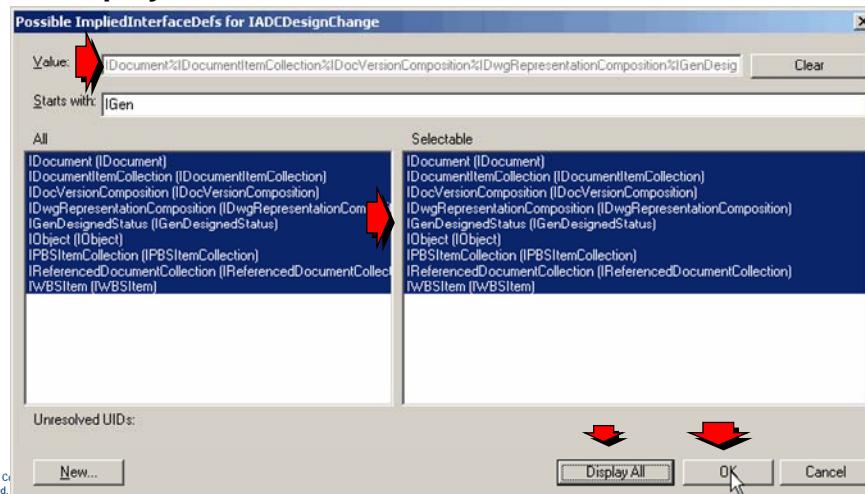
- Select the **IDocumentItemCollection** interface from the displayed list.
- Select the **IDocVersionComposition** interface from the displayed list.
- Select the **IDwgRepresentationComposition** interface from the displayed list.
- Select the **IPBSItemCollection** interface from the displayed list.
- Select the **IReferencedDocumentCollection** interface from the displayed list.
- Select the **IWBSSItem** interface from the displayed list.
- Select the **IGenDesignedStatus** interface from the displayed list.

© 2005, Intergraph Corp.
All Rights Reserved.



Adding Additional Implies Relationships

- Select the **IGenDesignedStatus** interface from the displayed list.

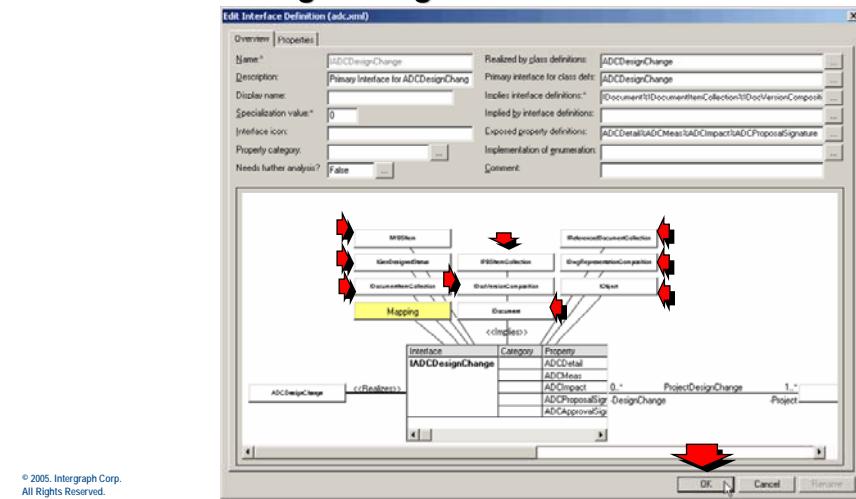


© 2005, Intergraph C
All Rights Reserved.



Adding Additional Implies Relationships

- Click the **OK** button to save the changes to the **IADCDesignChange** interfacedef.

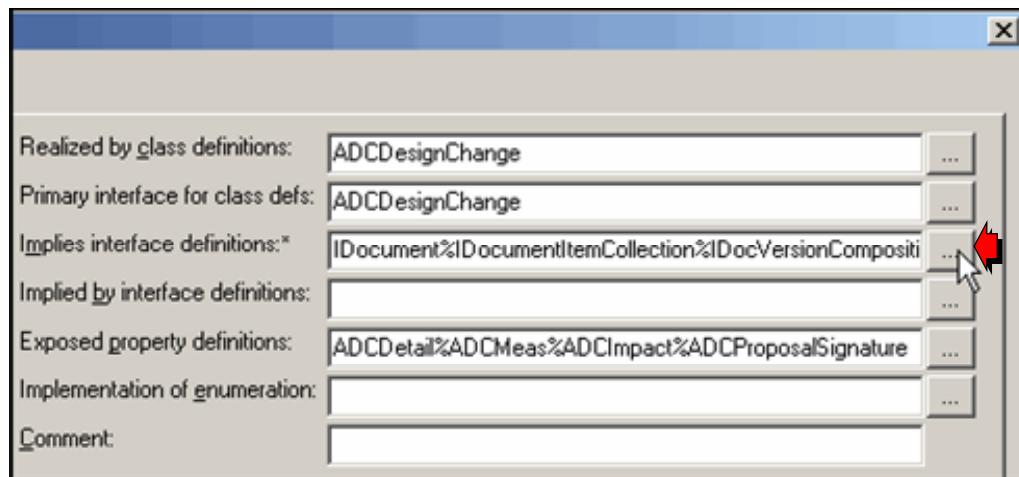


© 2005, Intergraph Corp.
All Rights Reserved.



Adding Additional Implies Relationships

Zoomed in view of the *Edit Interface Definition* dialog box.

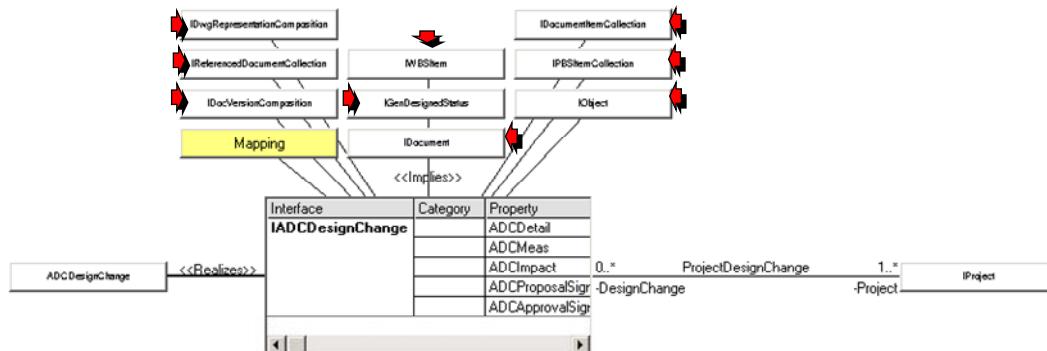


© 2005, Intergraph Corp.
All Rights Reserved.



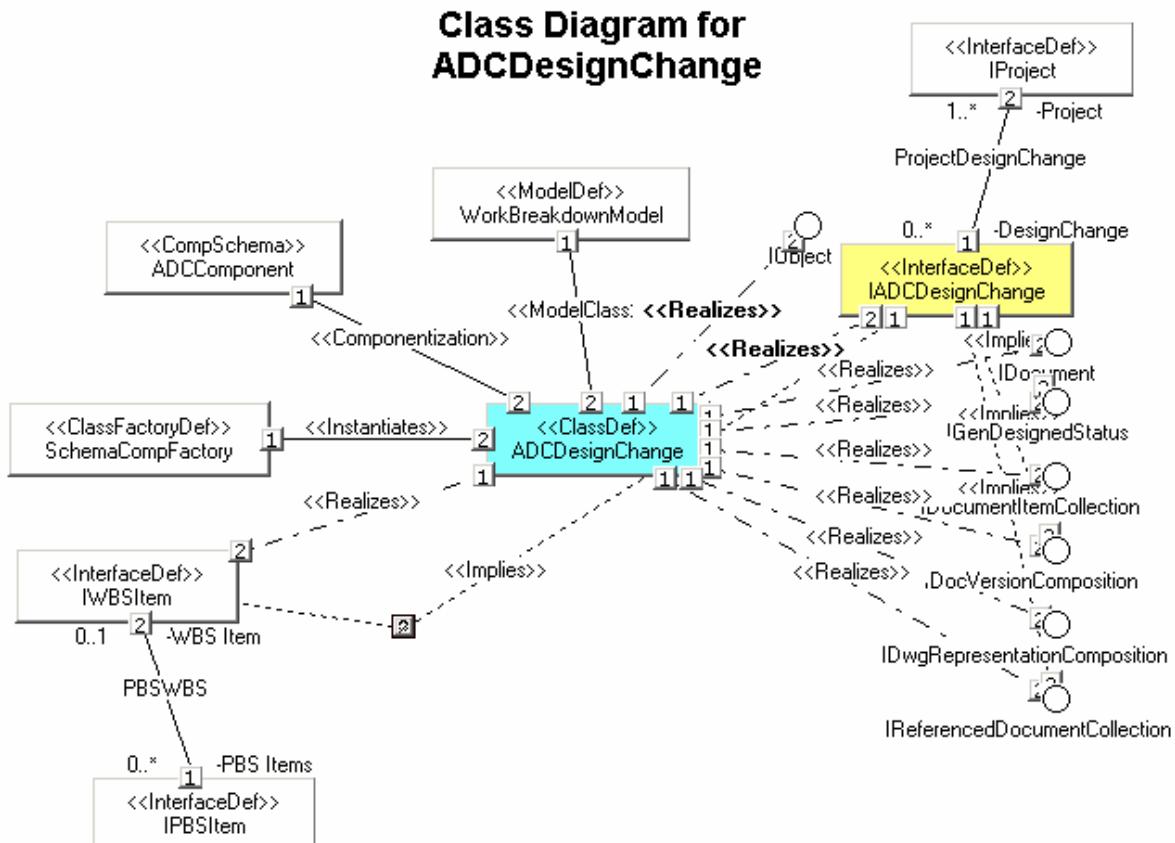
Adding Additional Implies Relationships

Zoomed in view of the *Edit Interface Definition* dialog box.



© 2005, Intergraph Corp.
All Rights Reserved.

The UML diagram shows the changes that have been made to the schema.



4.8 Shared Object Definitions

A shared object definition is used to group together similar classes that define the same object in different domains. For example, in a *Process Flow Diagram* (PFD) a user places a pump called P100 and publishes the document.

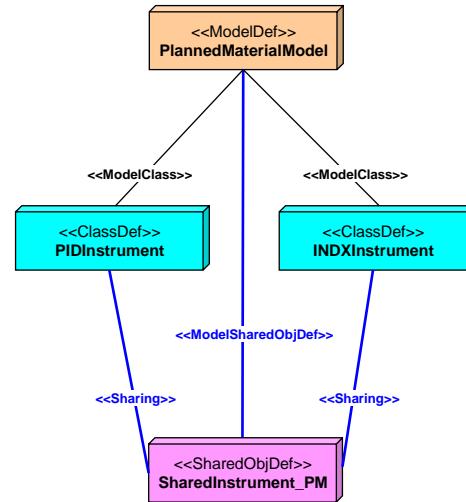


Shared Object Definitions

Shared object definitions (`SharedObjDef`) are used to group together similar classes that define the same object in different domains.

In concurrent engineering, two different tools can create and publish the same object. For example, instruments can be created and published in both SmartPlant P&ID and SmartPlant Instrumentation.

ClassDefs that can be shared have a Sharing relationship with SharedObjDefs.



© 2005, Intergraph Corp.
All Rights Reserved.

At the front end of the engineering process, not a lot of information 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 SmartPlant.

In SmartPlant P&ID, engineers start adding value to the design by adding more and more information, such as the type of pump needed and so on. When SmartPlant P&ID publishes the document 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 Zygad, there are thousands of properties describing the same object, further enriching it.

In each tool, the pump (P100) is published with a different class definition, including PFDProcessEquip, PIDProcessEquip, and EQDCentrifugalPump. However, a *SharedObjDef* indicates that these three classes definitions all define the same object in SmartPlant. The three class definitions coexist in a sharing relationship. The *SharedObjDef* 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).

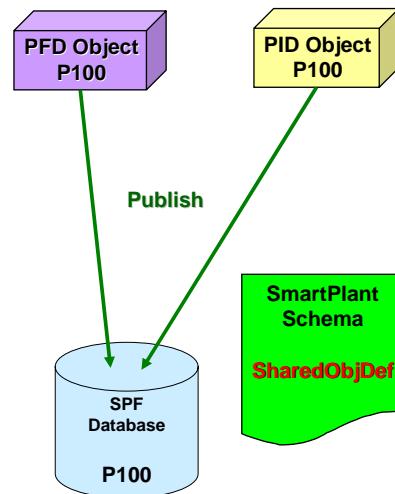
The two publishes establish a “*same as*” relationship. The same as relationship indicates that object AA1 is the same as object AA2.



Shared Object Definitions

In SmartPlant Foundation, interfaces and properties that are shared are overlaid (merged).

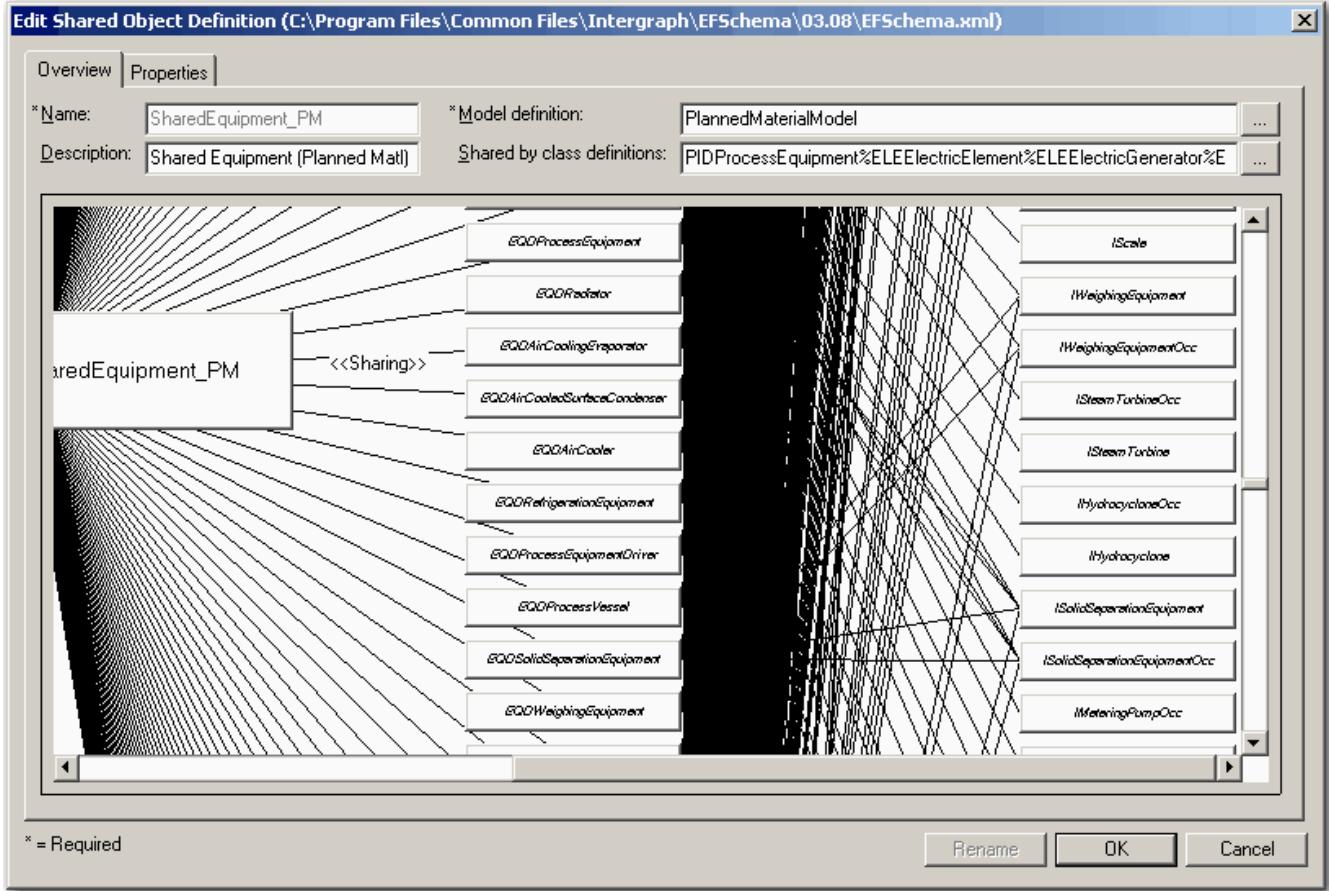
Interfaces that are not shared are appended to the object in the SPF data database.



4.8.1 Properties of a Shared Object Definition

Shared object definitions have the following properties:

- Name** – Specifies the name of the shared object definition.
- Description** – Specifies the description of the shared object definition.



- Model definition** – Specifies the model definition to which the shared object definition belongs. Objects can only be shared if they exist in the same model.
- Shared by class definitions** – Specifies the class definitions that have a sharing relationship with this SharedObjDef.

4.9 Activity – Creating Properties, Enumerated Lists and Relationships

You will continue to add to your model in this activity which is a continuation of the activity 2 from Chapter 3.

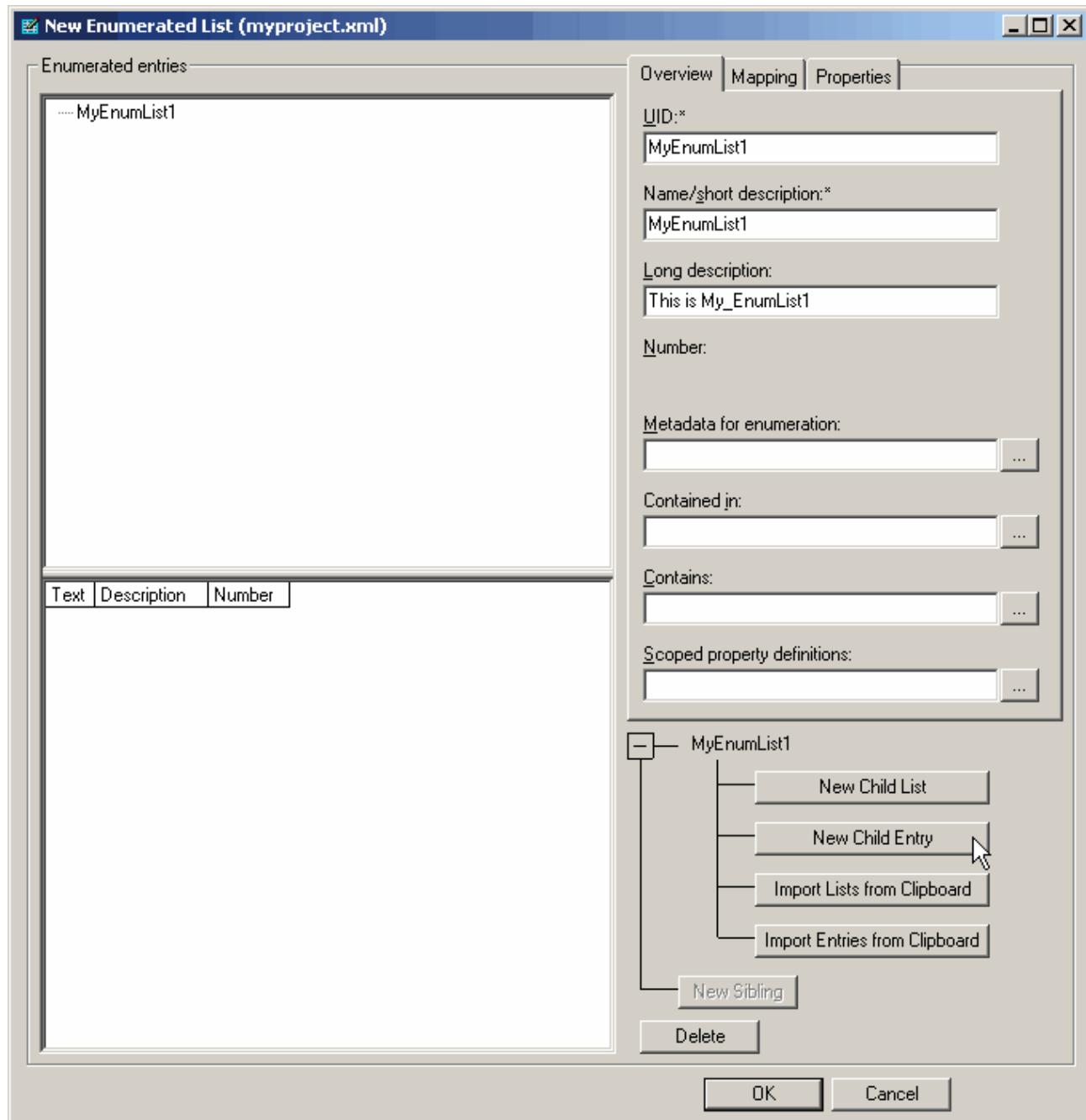
1. Open the Schema Editor and the saved configuration file, “myproject.cfg”.
 - Click the **File Configurations** button and select **Open Configuration...** from the *Schema Editor Workflows* dialog
 - Select the **myproject.cfg** file and click **Open** (Path is C:\Program Files\Common Files\Intergraph\EF Schema\03.08)
2. Open a package and saved view from the previous session so that you can continue with your modeling exercise.

Create an Enumerated List Type

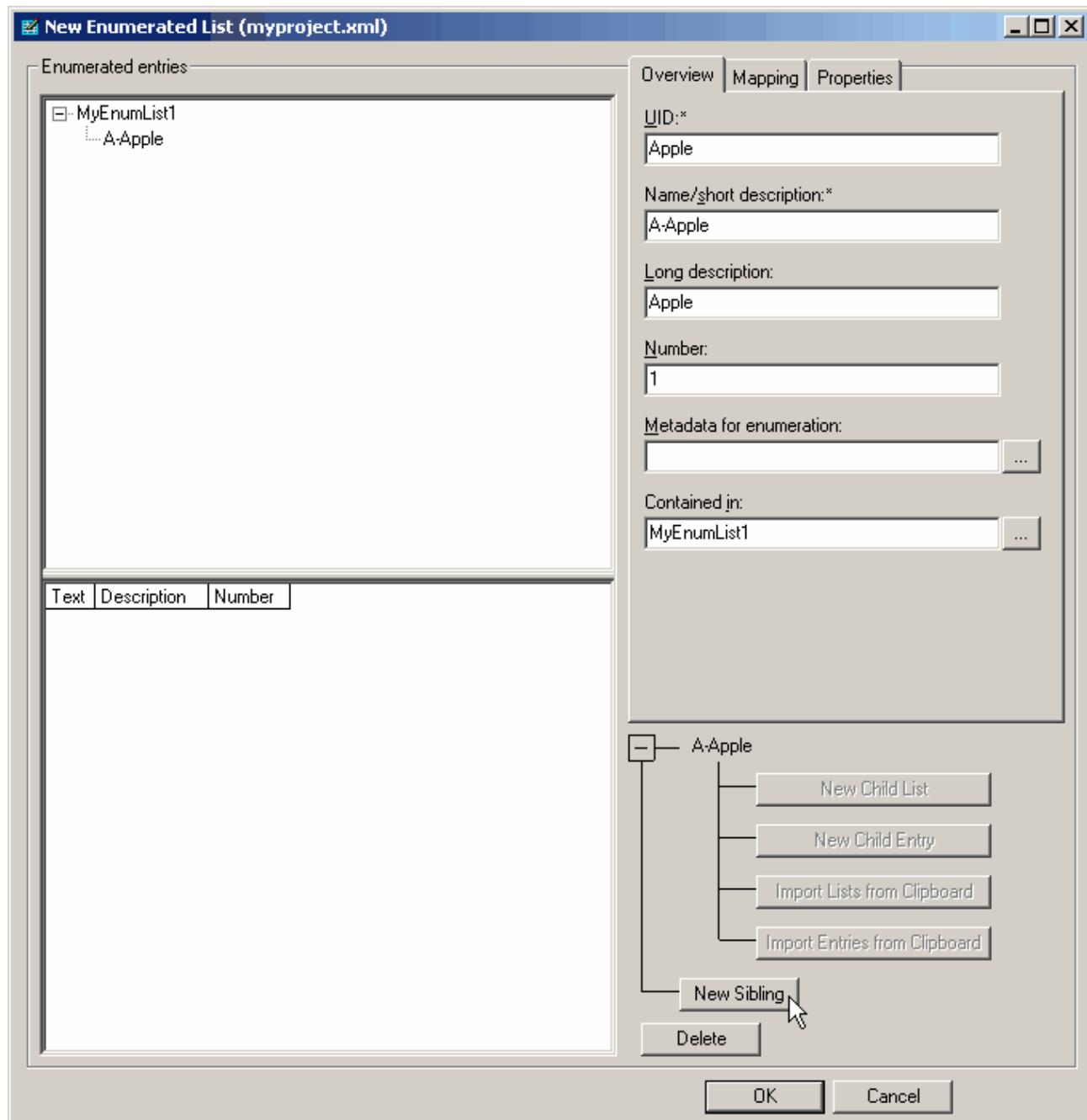
3. Use the Editor view to view and edit your model. Drag **EnumListType** from the **Create** tree to the UML view.

Note: You can also double-click the **EnumListType** node or highlight the node and click the right mouse button to create a new interface definition.

4. In the *New Enumerated List* dialog box, type a *UID*, a *Short description* and a *Long description* as shown below:



5. Click the **New Child Entry** button and enter the following info:



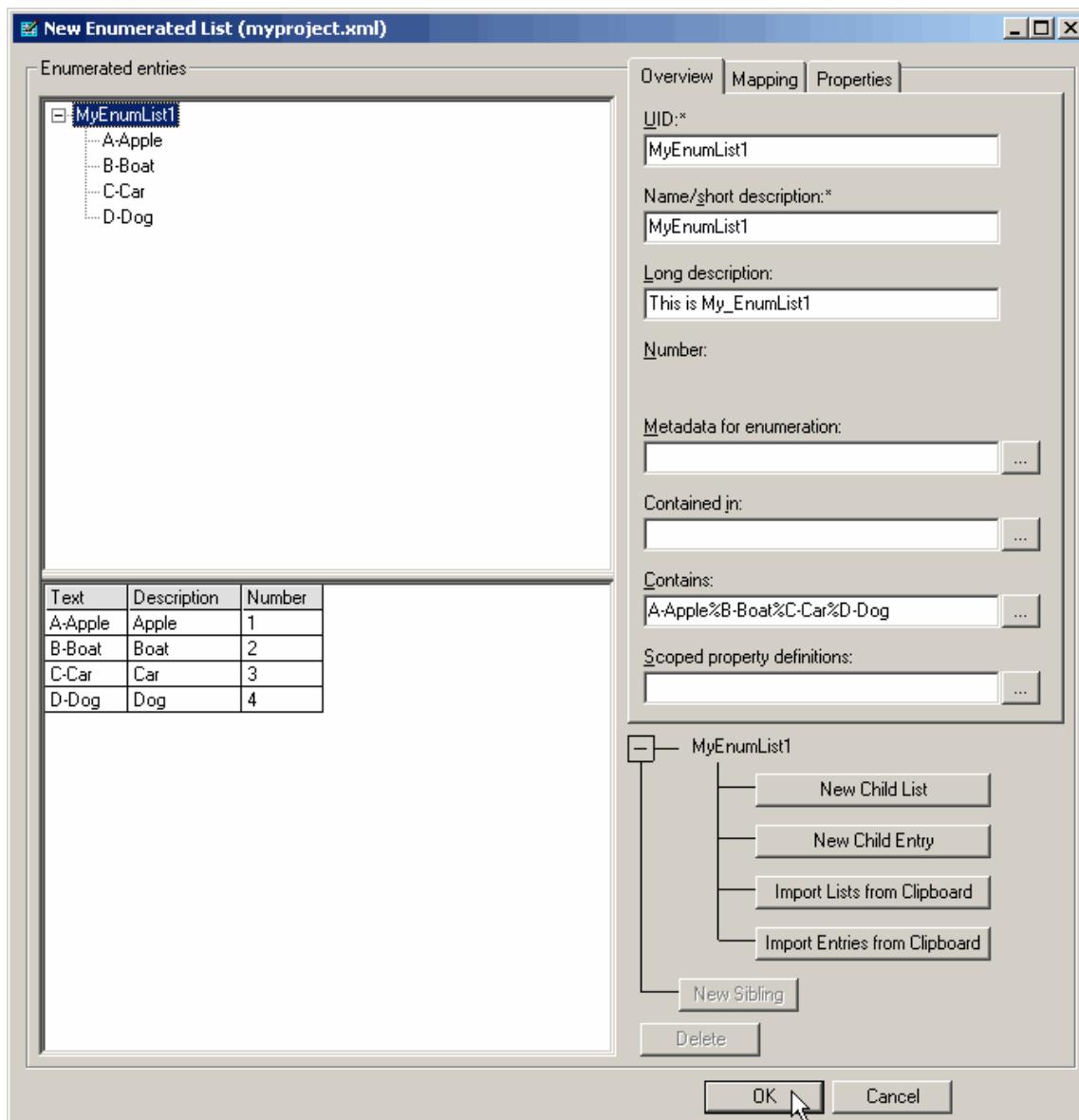
6. Click the **New Sibling** button and enter the following info:

- UID: Boat**
- Short name: B-Boat**
- Long name: Boat**

Continue to add siblings as following:

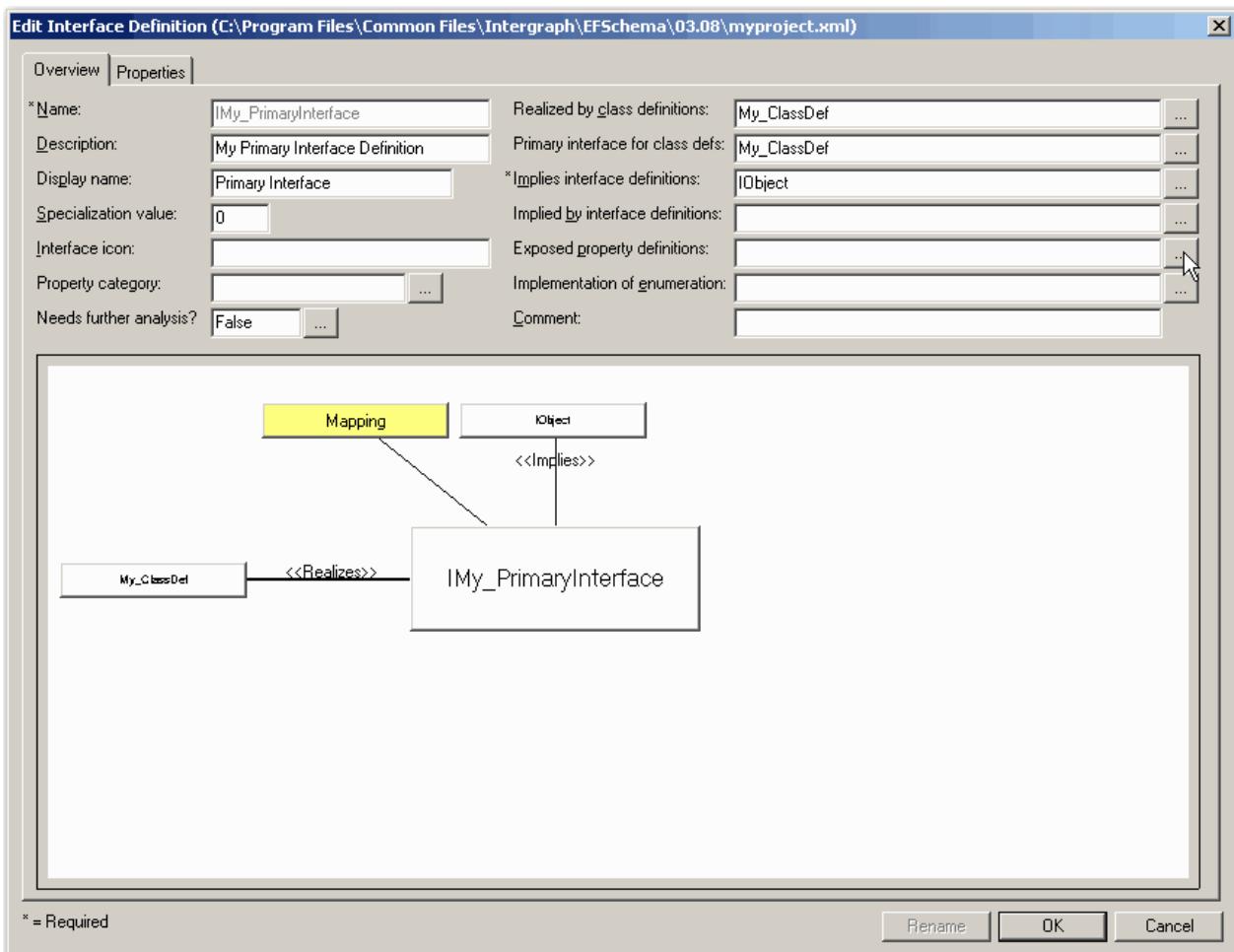
- UID: **Car**
- Short name: **C-Car**
- Long name: **Car**
- UID: **Dog**
- Short name: **D-Dog**
- Long name: **Dog**

7. Click **OK** to create the new *Enumerated List Type* with the new values.

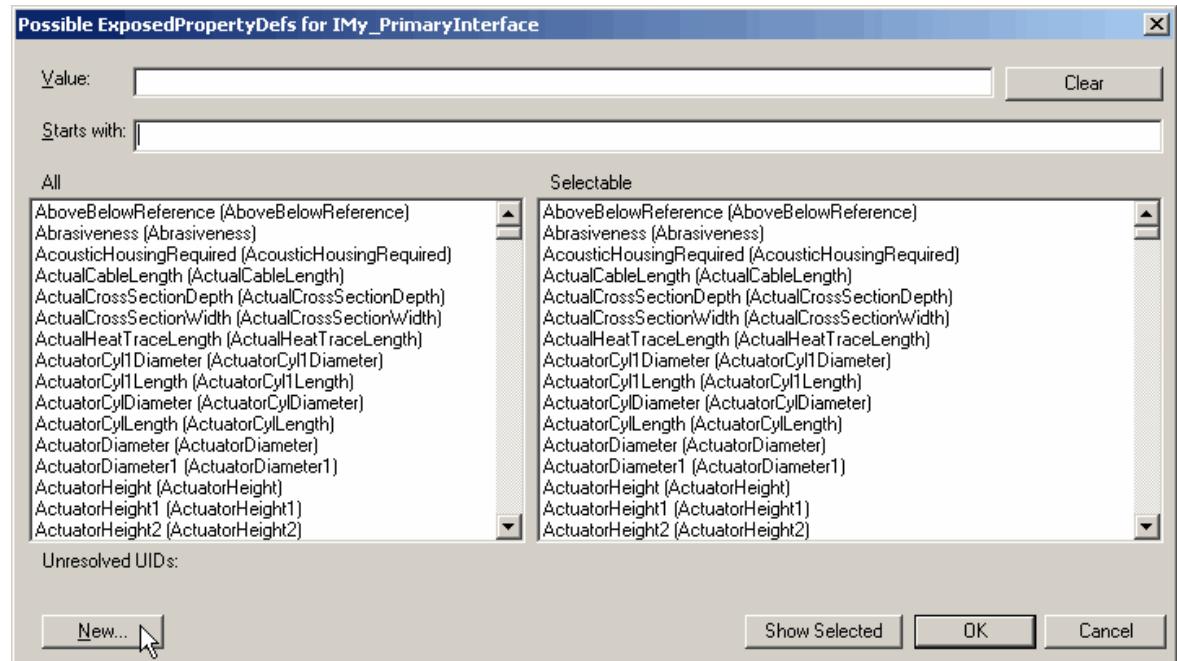


8. Use the Editor view to view and edit your model. Right-click on the **IMy_PrimaryInterface** interface def from the tree view and choose the **Edit** command to edit this interface def.

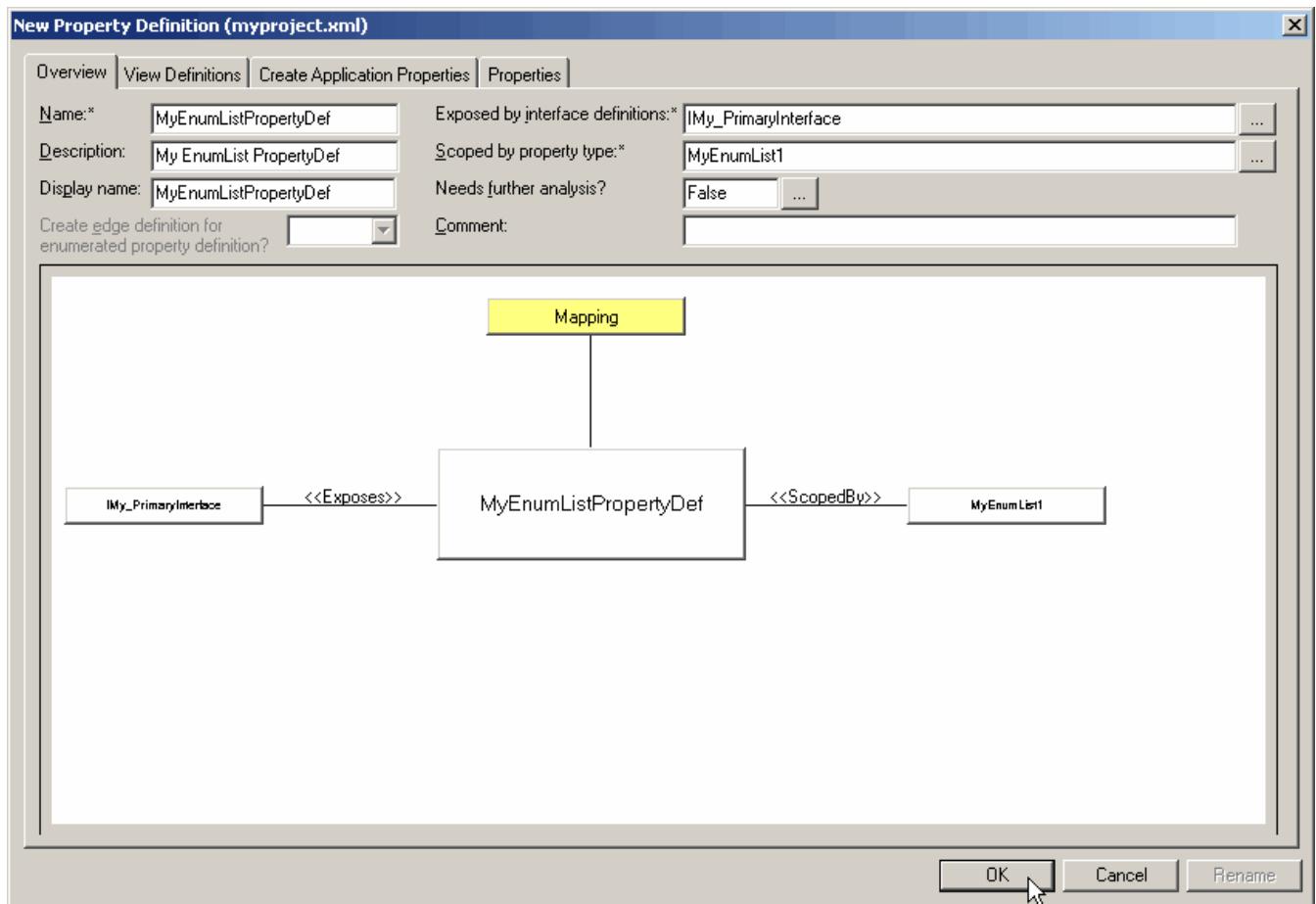
9. Click  next to the *Exposed property definitions* box.



- ❑ Click **New** in the *Possible ExposedPropertyDefs* for *IMy_PrimaryInterface* dialog box.



10. Create the **Property Def** as shown on the next page:



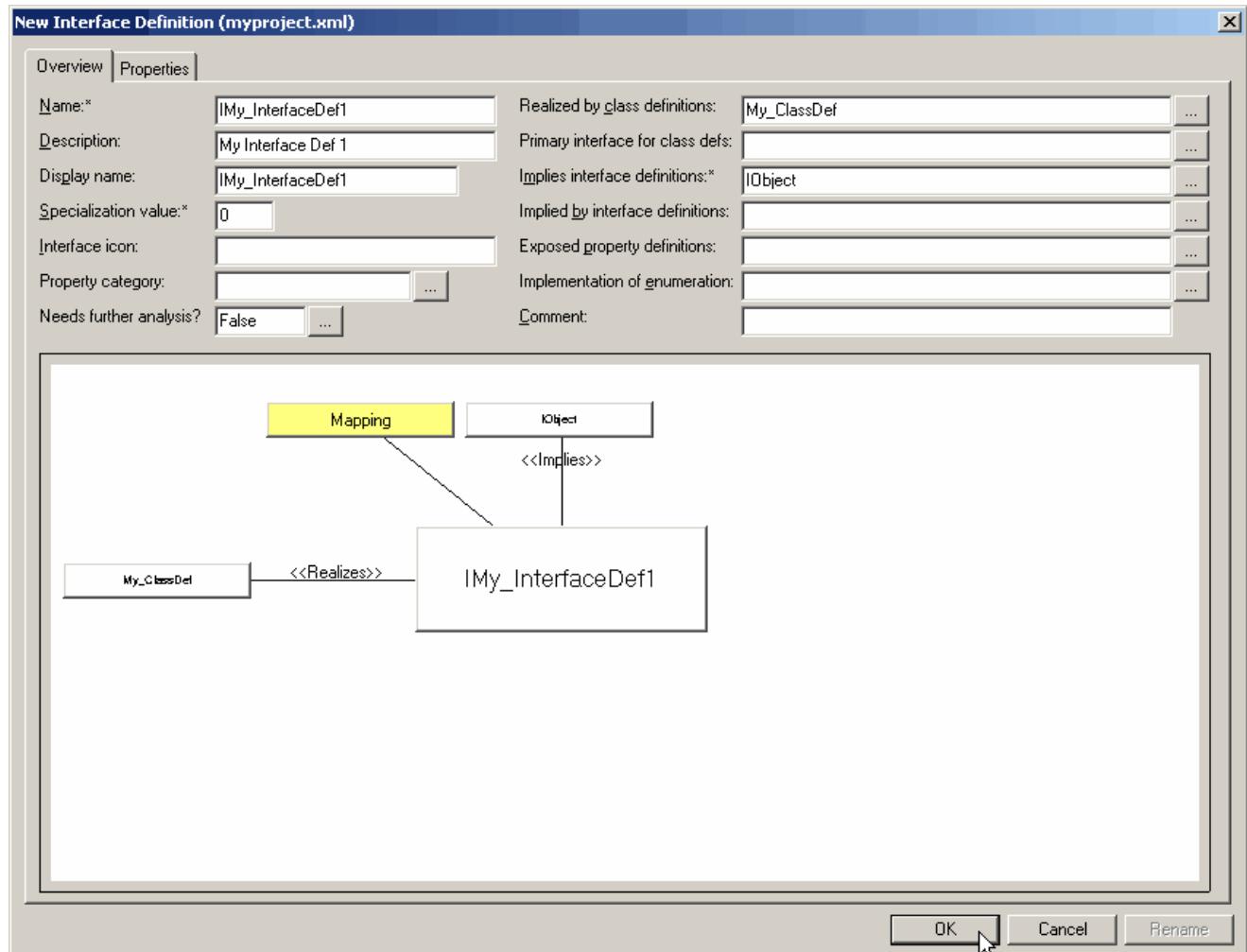
- Use the new **EnumListType**, **MyEnumList1**, that was created earlier in this exercise as the **.PropertyType**.
 - Click **OK**.
11. Click **OK** in the *Possible ExposedPropertyDefs for IMy_PrimaryInterface* dialog box to save the new **Property Definition**.
12. Click **OK** in the *Edit Interface Definition* dialog box.

Creating another Interface Definition

13. Drag **InterfaceDef** from the **Create** tree to the UML view.

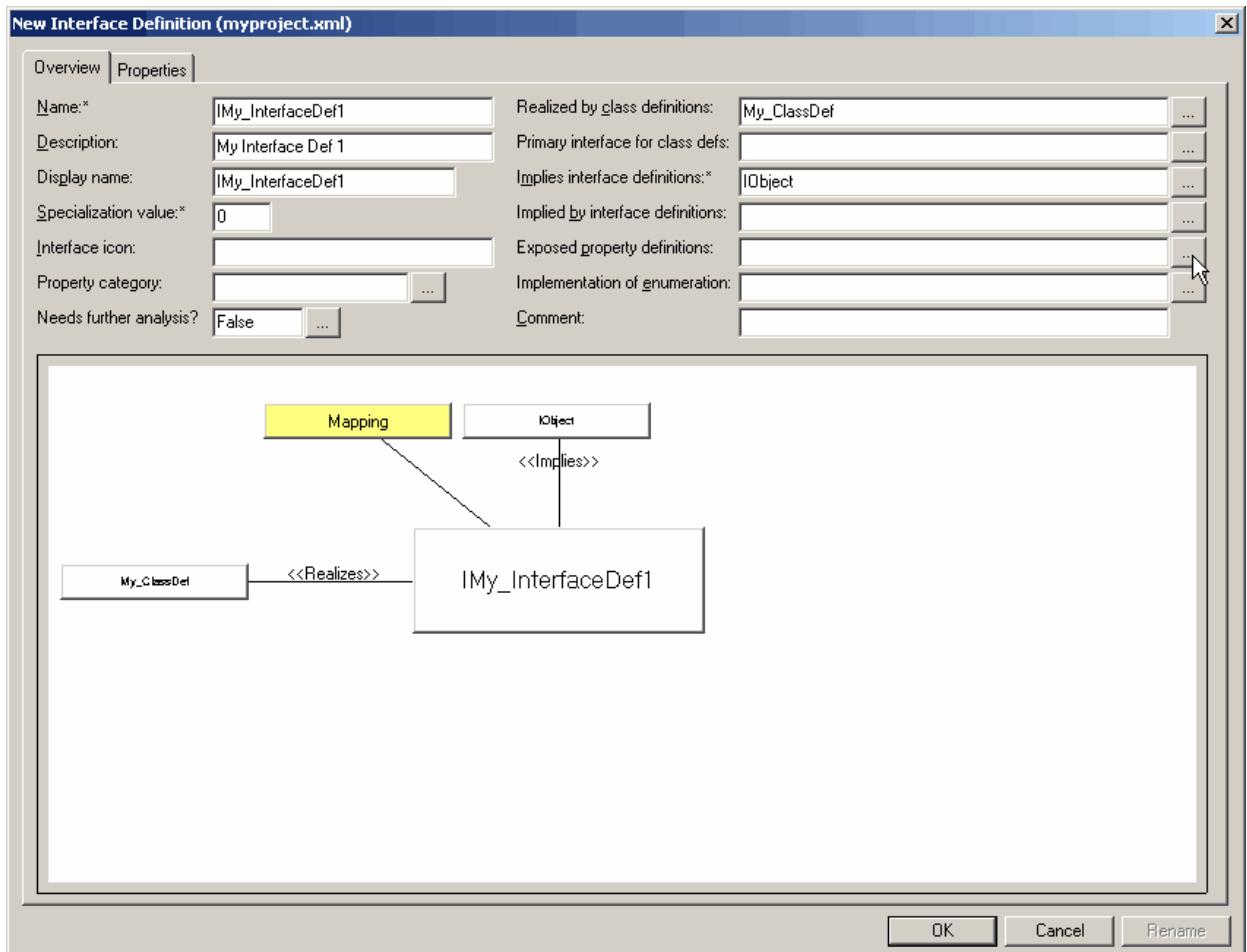
Note: You can also double-click the **InterfaceDef** node or highlight the node and click the right mouse button to create a new interface definition.

14. In the *New Interface Definition* dialog box, type **IMy_InterfaceDef1** for the **Name**, **My InterfaceDef 1** for the **Description**, and **IMy_InterfaceDef1** for the **Display Name**, and select **My_ClassDef** in the *Realized by class definitions* box and **IObject** in the *Implies interface definition* box as shown below:

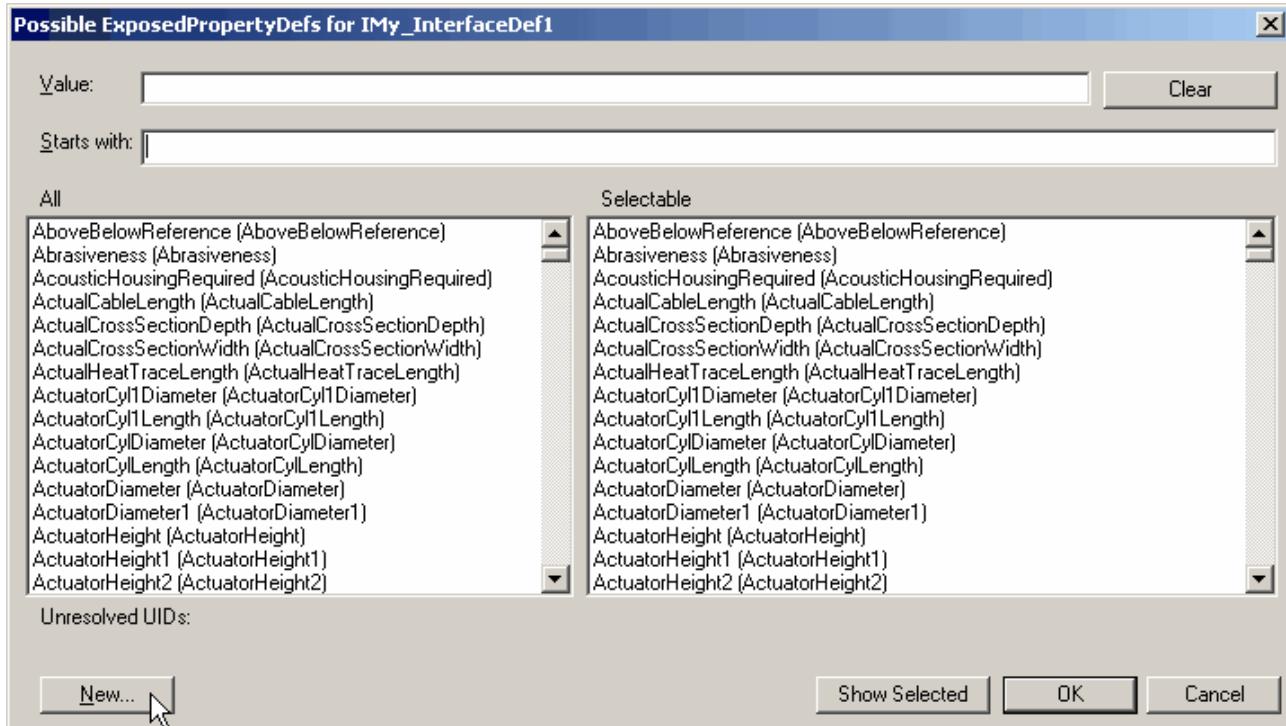


Note: You can type the object names in the text boxes on the right side of the **New Interface Definition** dialog box instead of using the button to display the relationship help dialog box.

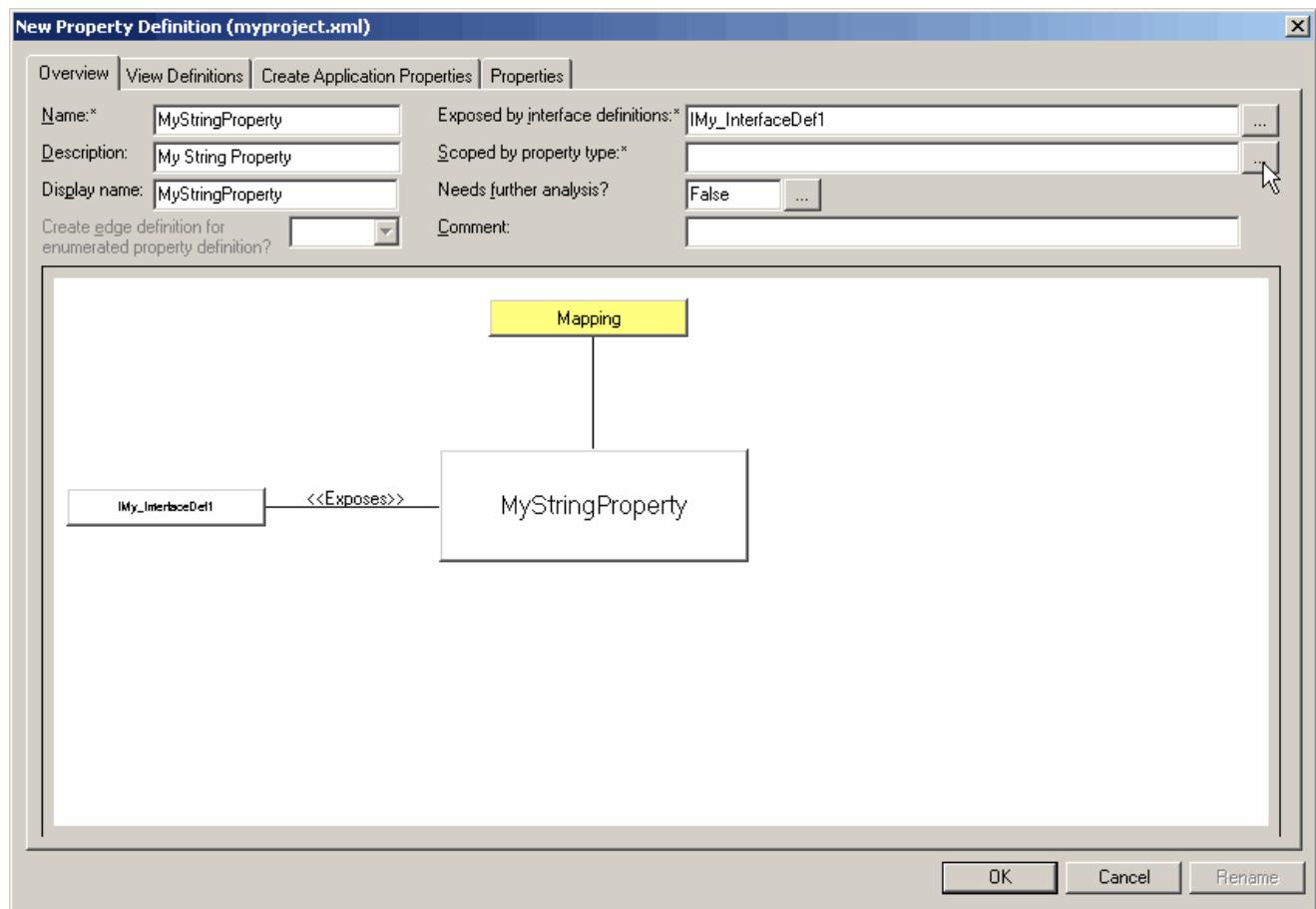
- Click  next to the *Exposed property definitions* box.



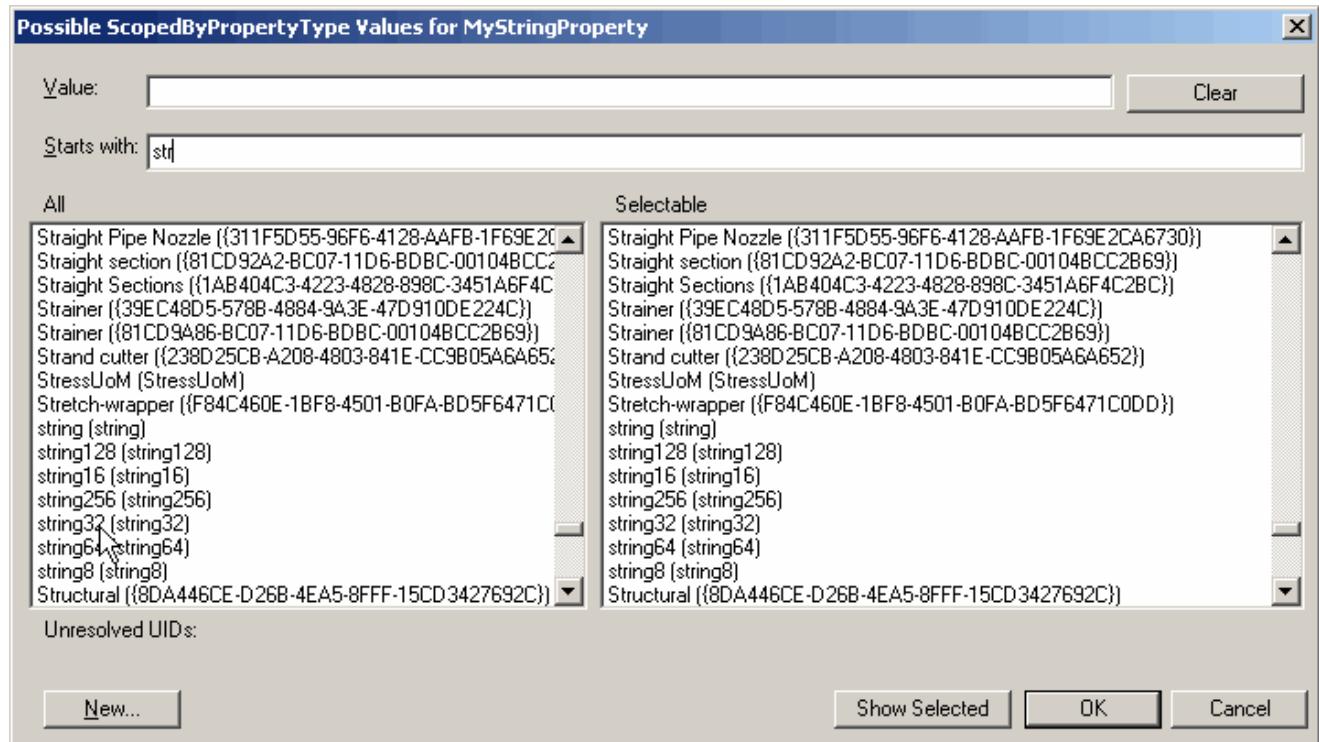
- ❑ Click **New** in the *Possible ExposedPropertyDefs for IMy_InterfaceDef1* dialog box.



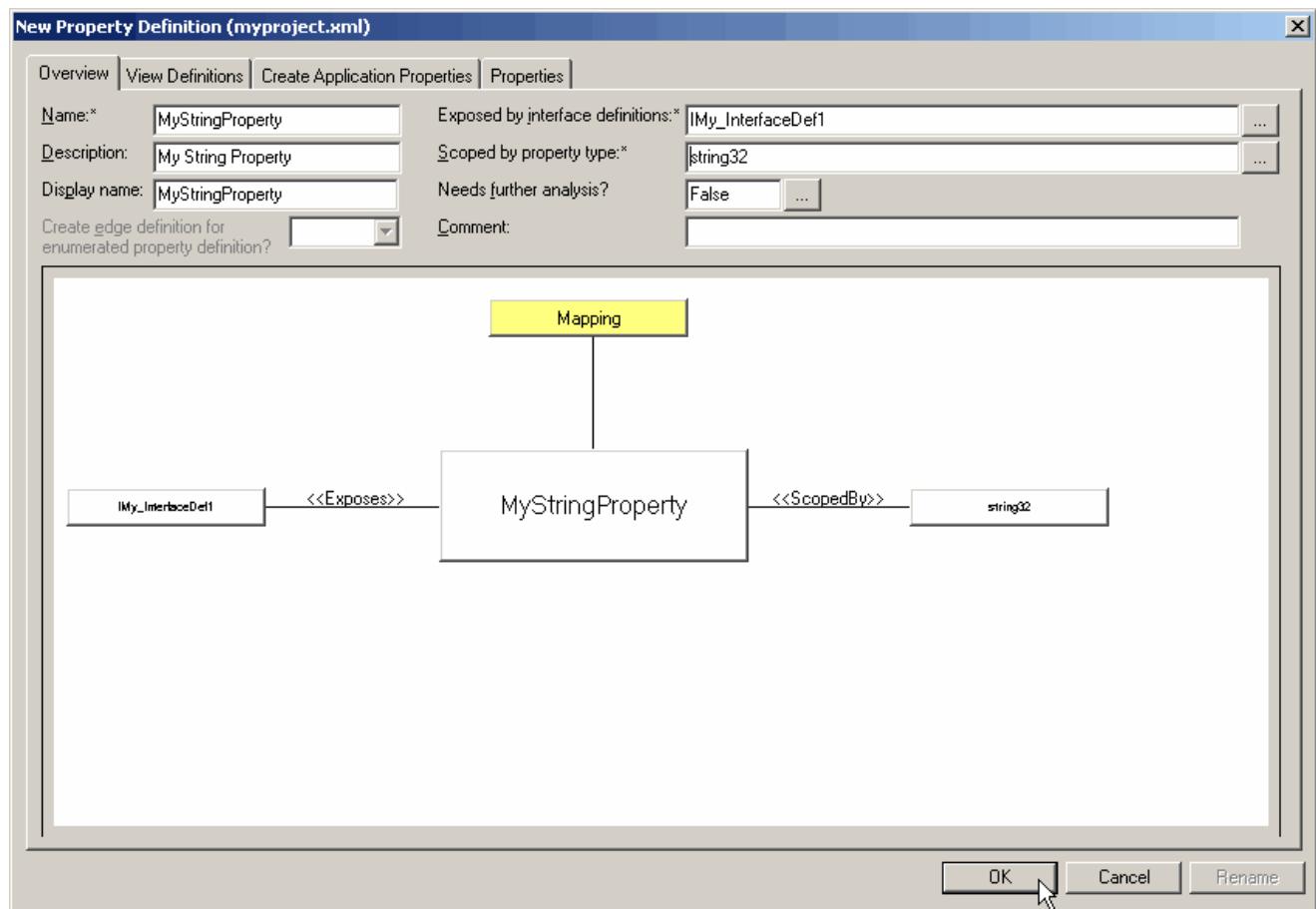
15. Create the **Property Def** as shown on the next page:



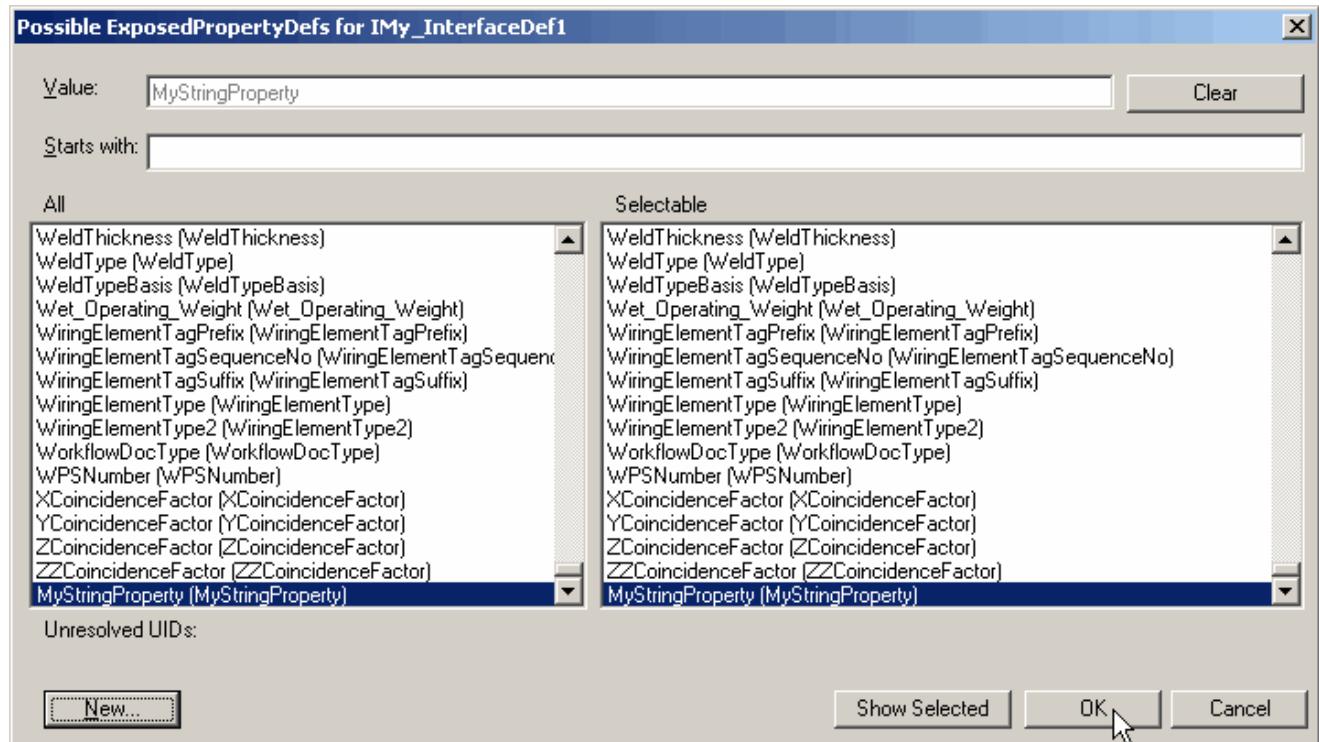
- Click next to the *Scoped by property type* box.



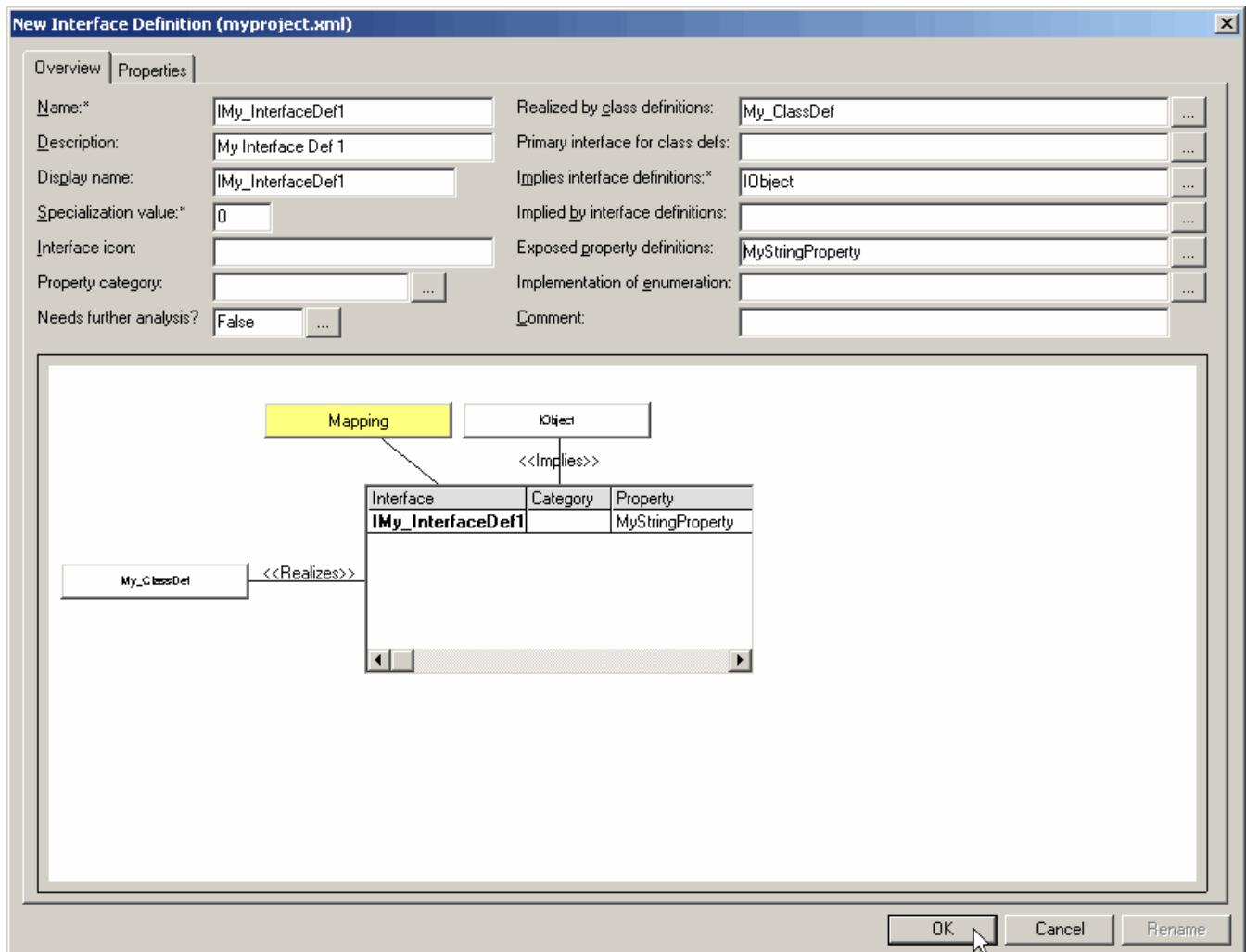
- Select **string32** in the *Possible ScopedByPropertyType Values* dialog.



16. Click **OK** to create the new property definition.



17. Click **OK** to accept the new property definition.



18. Click **OK** to create the new interface definition.

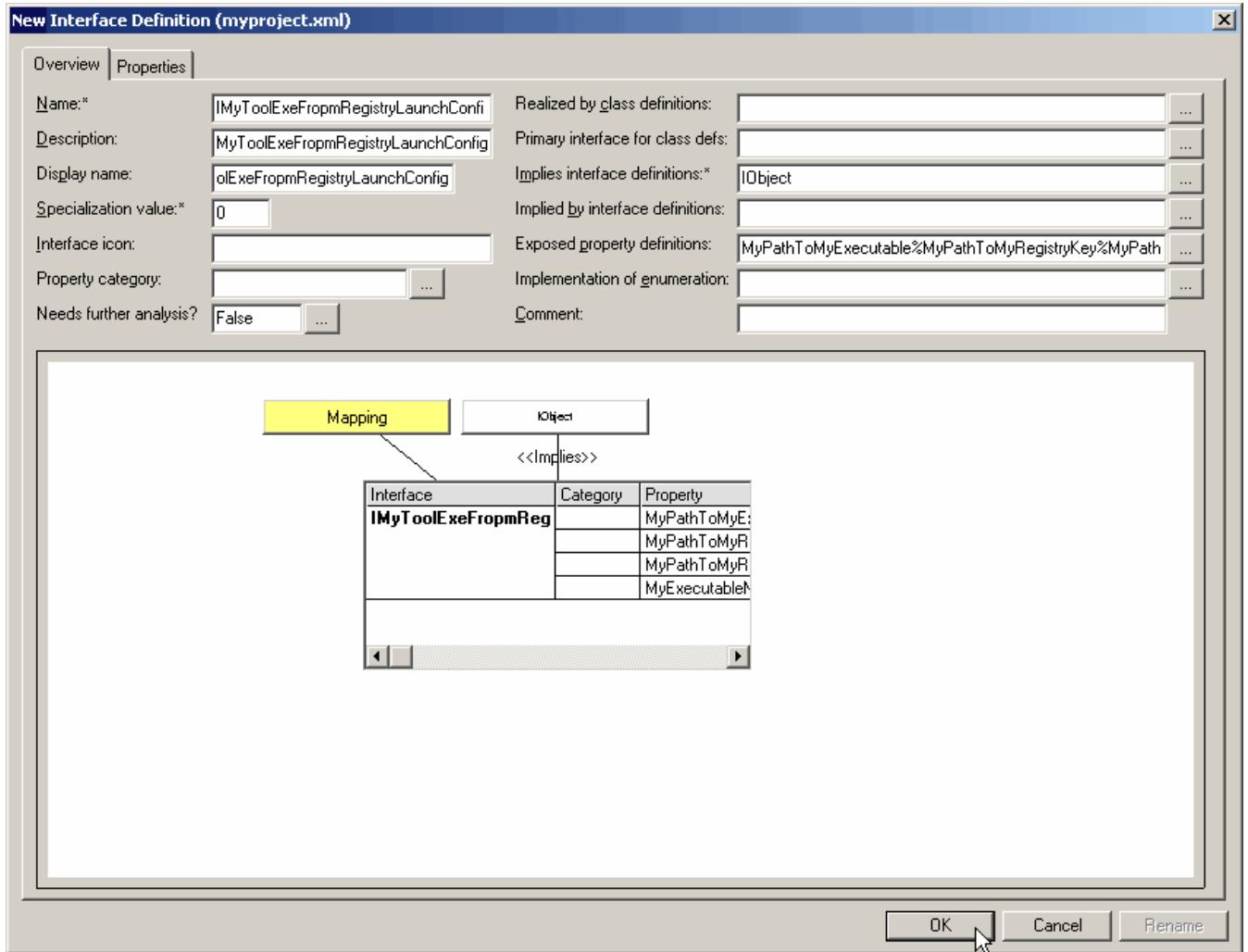
Creating Additional Objects

19. Create two Enumerated List types with the following values:

- EnumListType:** Name: MyPathToExe
- Values:** ClassesRoot, CurrentConfig, CurrentUser, DynData, LocalMachine, PerformanceData, Users
- EnumListType:** Name: MyNeworSelect
- Values:** New, Select

20. Create an Interface Definition with the following properties:

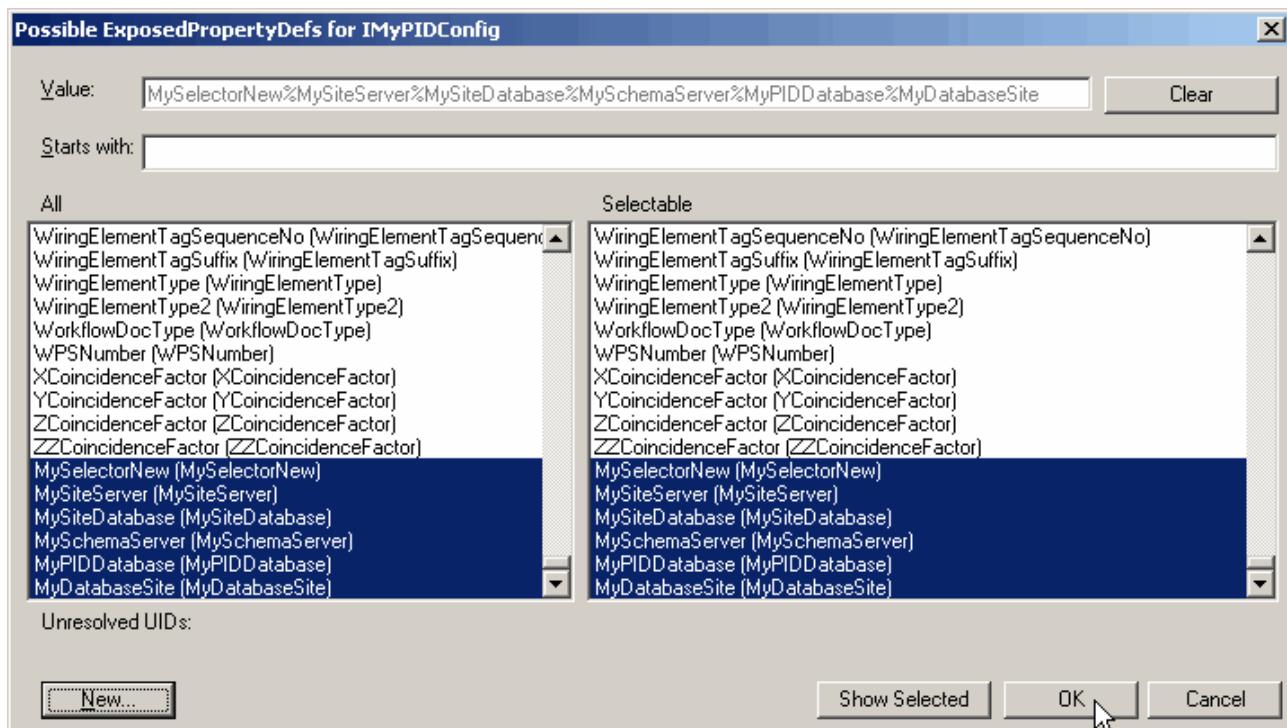
- InterfaceDef:** IMyToolExeFromRegistryLaunchConfig
- PropertyDef:** MyPathToMyExecutable - Property Type: MyPathToExe
- PropertyDef:** MyPathToMyRegistryKey - String128
- PropertyDef:** MyPathToMyRegistryKeyValueName -String128
- PropertyDef:** MyExecutableName - String128
- Implies InterfaceDefs:** IObject



21. Create a ClassDef with a Primary Interface and the following properties:

- ClassDef:** MyPIDConfig
- PrimaryInterface:** IMyPIDConfig
- PropertyDef:** MySelectorNew - Property Type: MyNeworSelect

- PropertyDef:** MySiteServer - String128
- PropertyDef:** MySiteDatabase - String128
- PropertyDef:** MySchemaServer - String128
- PropertyDef:** MySchemaDatabase - String128
- PropertyDef:** MyPIDDATABASE - String128
- PropertyDef:** MyDatabaseSite - String128
- Implies InterfaceDefs:** IObject
- Other Realized InterfaceDefs:** IMyToolExeFromRegistryLaunchConfig, IMyPIDConfig

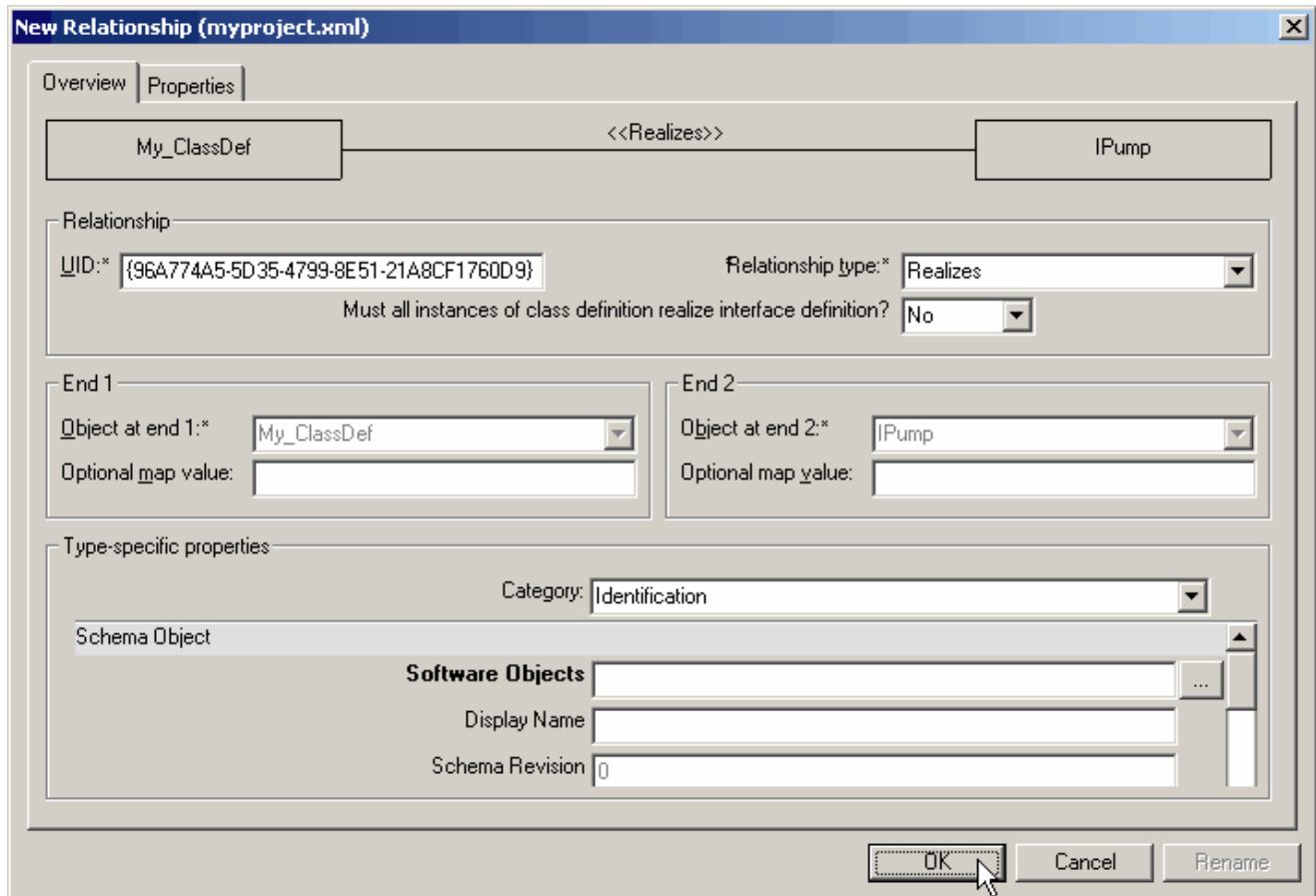


Creating Relationships

22. To create relationships using the editor views, do the following:

- Drag **Rel** from the **Create** tree to the UML view.

- In the *New Relationship* dialog box, select **Realizes** from the relationship type pick list.



23. Select **My_ClassDef** as the End1 Object.

24. Select **IPump** as the End2 Object.

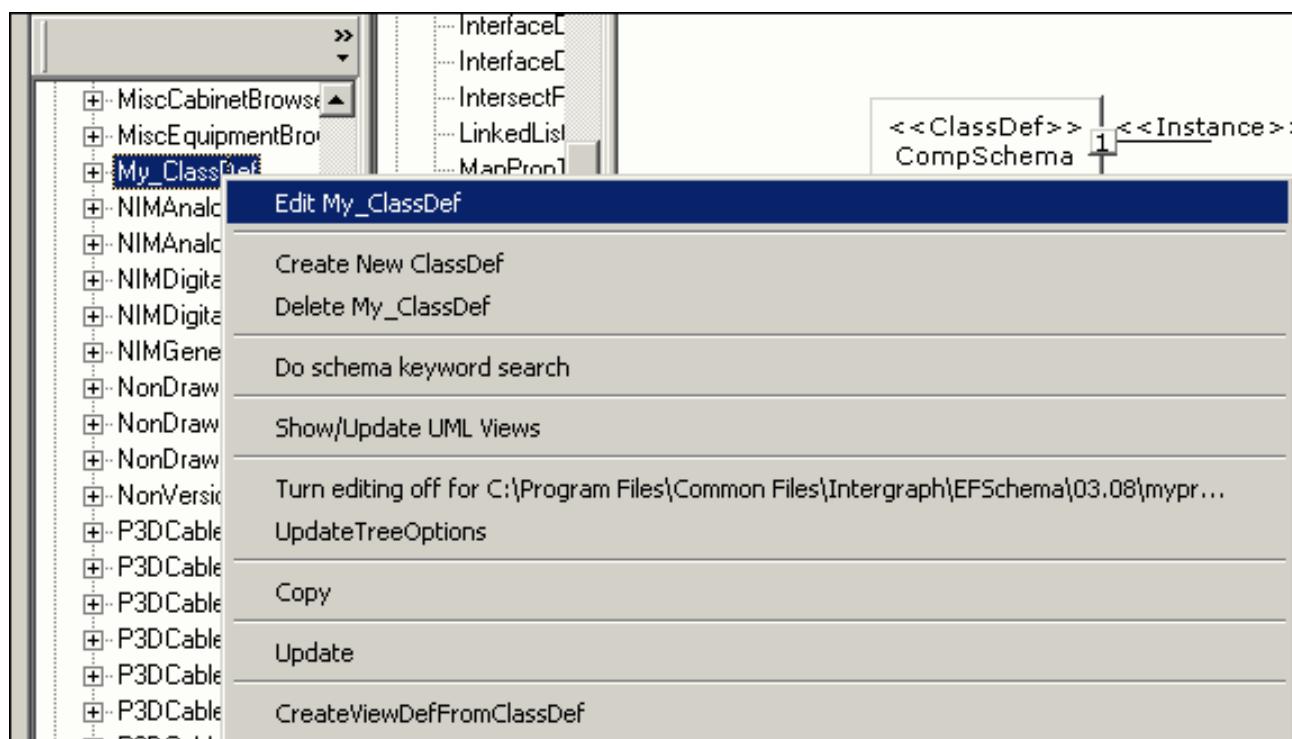
25. Click **OK** to create the new relationship.

Note: 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.

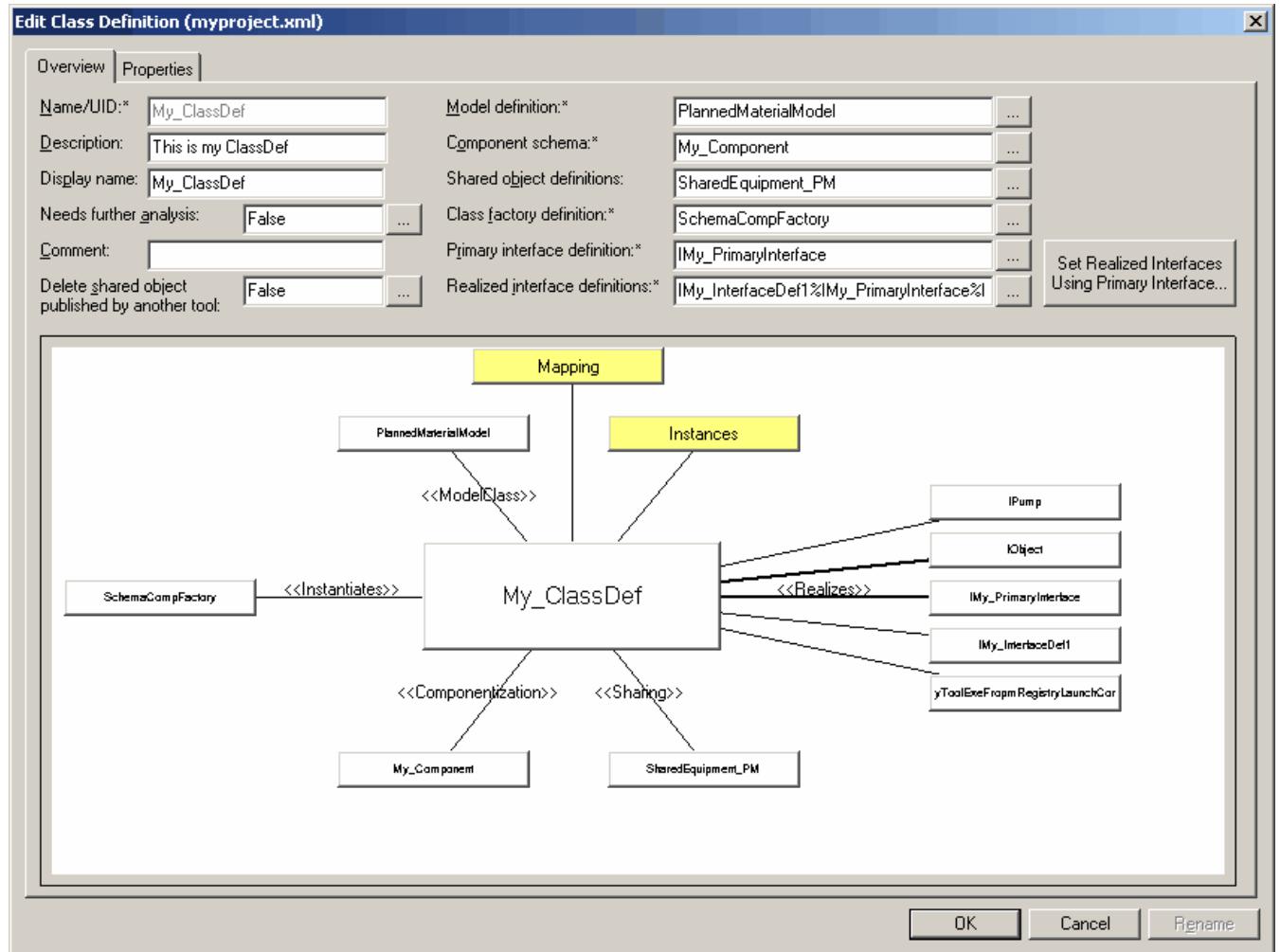
26. Expand ClassDef.

27. Navigate to My_ClassDef.

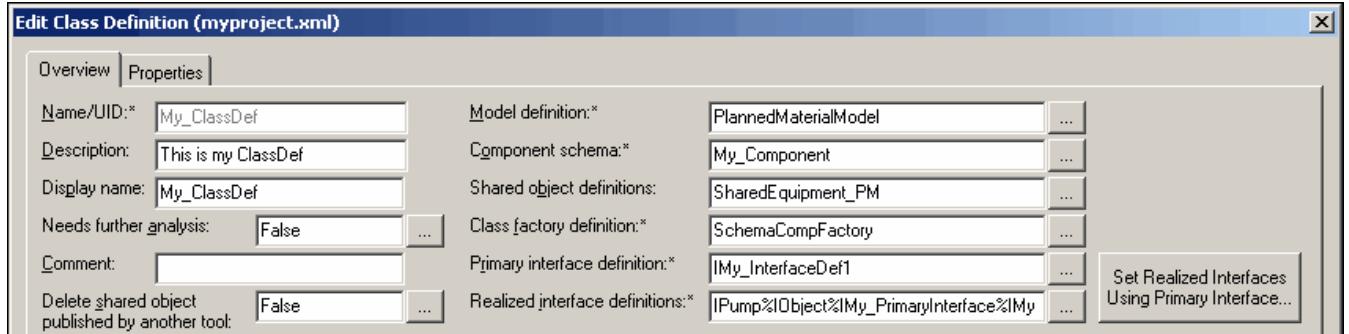
28. Right-click **My_ClassDef** and click **Edit My_ClassDef**.



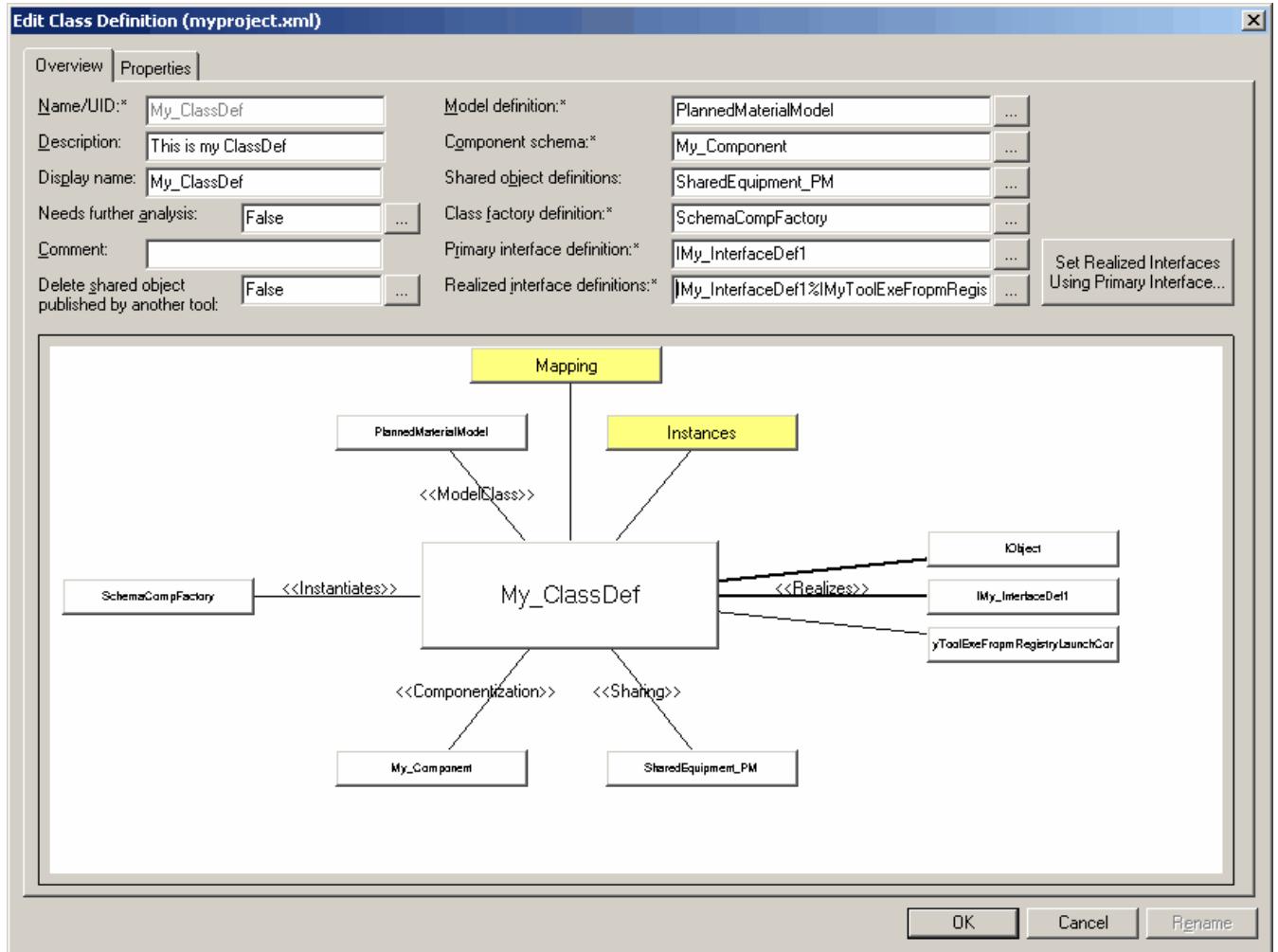
29. The *Edit Class Definition* dialog box show appear as shown below (Look at the UML to better see the relationships that currently exist for My_ClassDef):



30. In the *Edit Class Definition* dialog box, change the **Primary interface definition** from **IMy_PrimaryInterface** to **IMy_InterfaceDef1** as shown below:



31. Remove **IMy_PrimaryInterface** and **IPump** from the **Realized interface definitions**.



Note: If **IMy_InterfaceDef1** is not in the **Realized interface definitions** box, add it.

Note: To remove existing relationships, click selected objects in the list to clear them.

32. Click **OK** to save the new relationships.

Note: 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.

33. Click **View > Schema (Active)**.

34. On the **View** tab, click the **Tree/Drag-Drop UML** view.

35. Click **OK**.

36. Expand **ClassDef**.

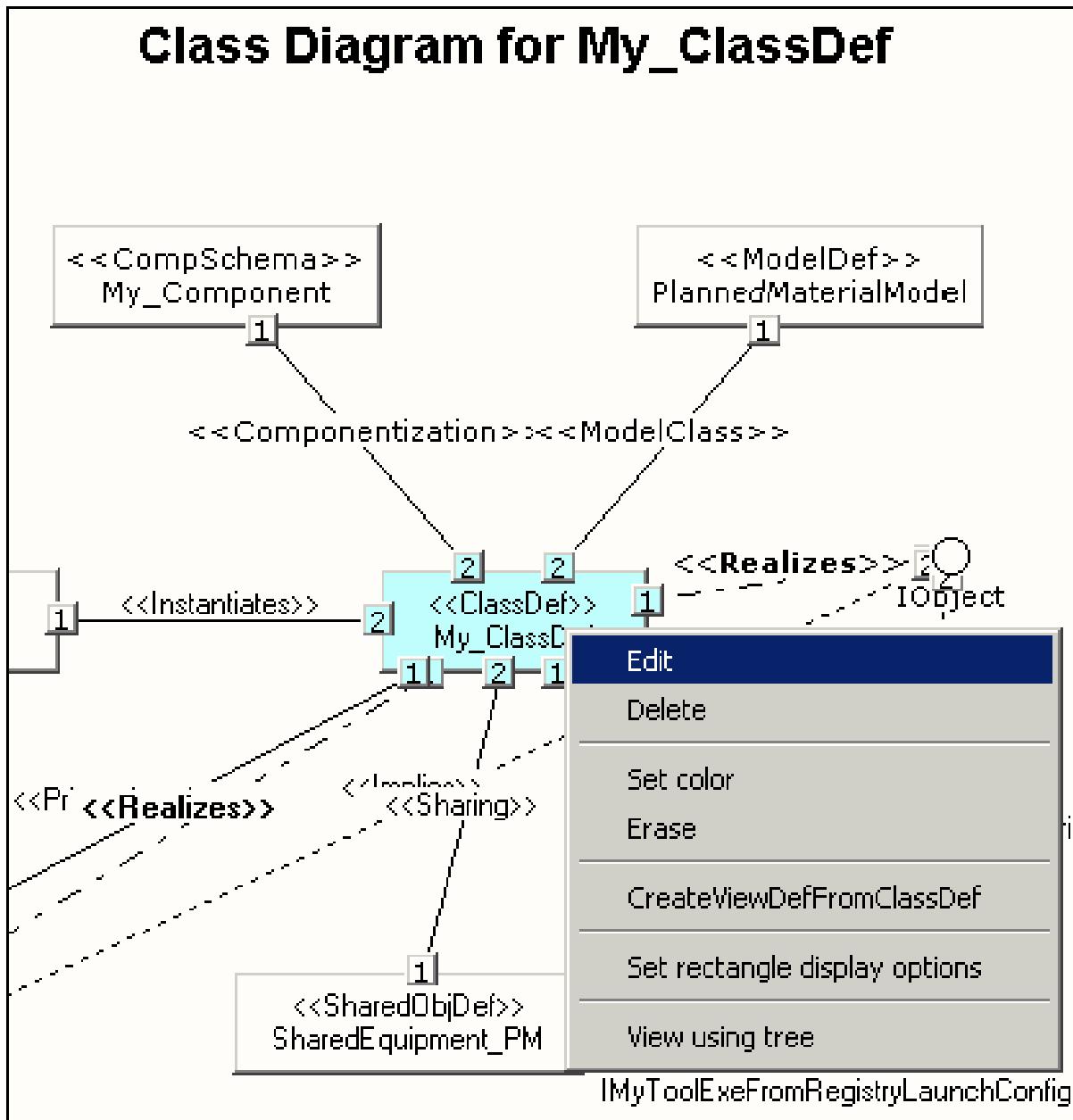
37. Navigate to **My_ClassDef**.

38. Drag **My_ClassDef** onto the UML view.

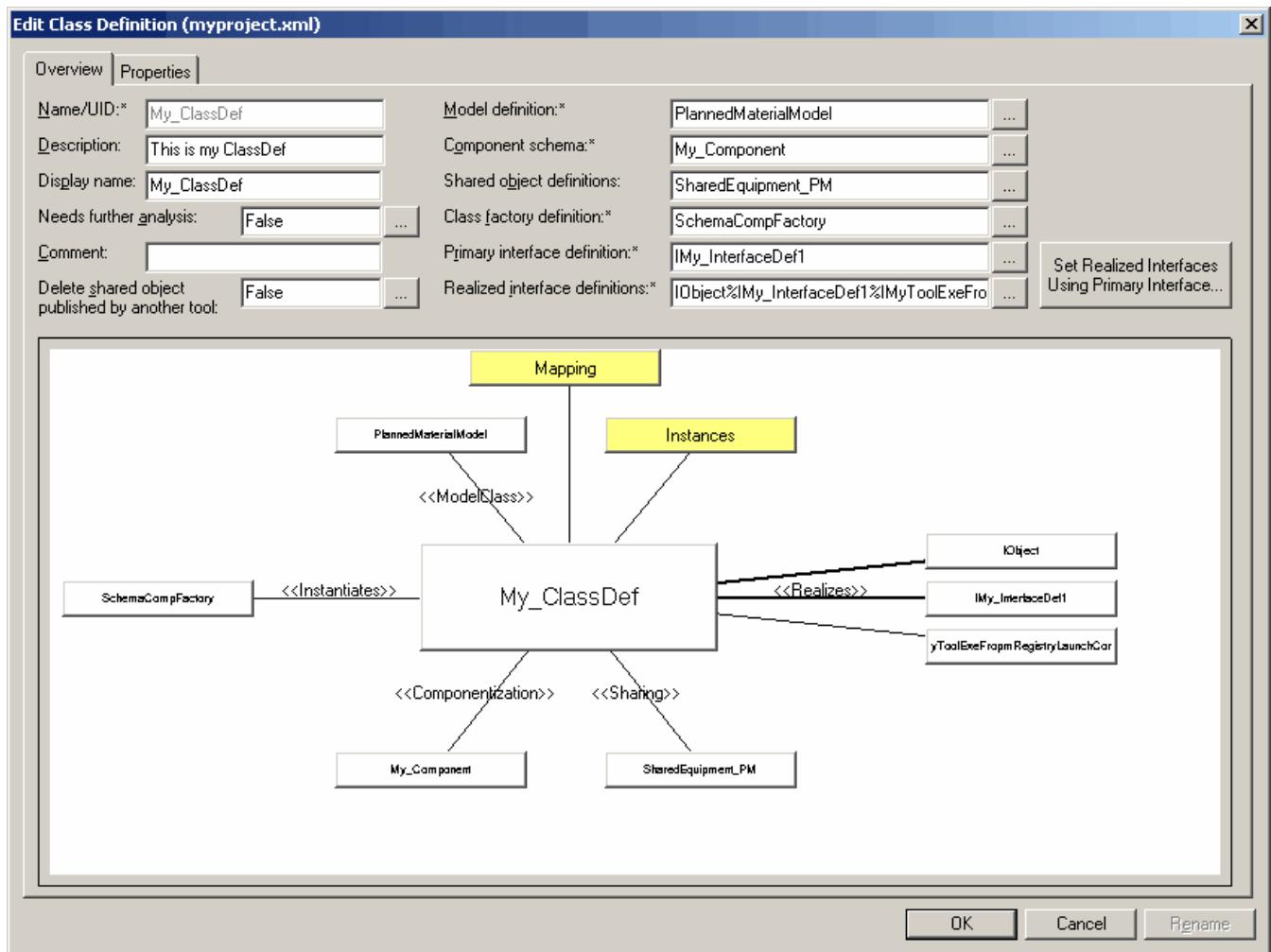
39. To view the standard UML diagram associated with the object, click **Yes** in the **Schema Editor** dialog box.

40. In the UML view, right-click the **My_ClassDef** object to modify the relationships.

41. On the shortcut menu, click **Edit**.



42. The *Edit Class Definition* dialog box should appear as shown below:



43. In the *Edit Class Definition* dialog box, change the **Primary interface definition** from **IMy_InterfaceDef1** back to **IMy_PrimaryInterface**.

44. Add **IPump** to the *Realized interface definitions* box as shown above.

45. Click **OK** to save the updated relationships.

46. Exit the Schema Editor and save the changes to **myproject.xml**.

47. Once you have exited the Schema Editor, you may take a short break until the other students have finished this activity.

5

C H A P T E R

Creating EdgeDefs, Graph Defs and View Defs

5. Creating EdgeDefs, Graph Defs, View Defs and Class View Maps

You will continue to add components to your data model in this chapter. This chapter will cover the object to object navigation functionality that is used by the SmartPlant Foundation Desktop client. For discussion purposes, we will refer to these object types as navigation objects.

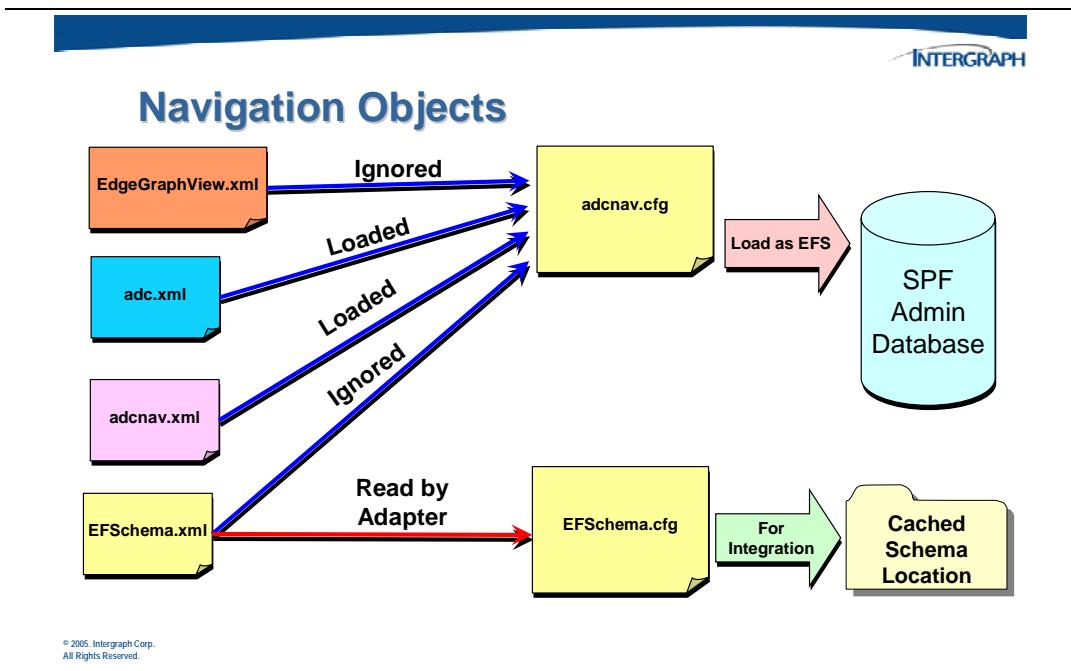
The ability to navigate from one object in the model to another object in the model can be determined by **Edge** definitions. This allows you to specify which interface to start the navigation, and which direction to use to see information associated with a different interface.

Once these navigation edges have been defined, they can be grouped together to form a Directed **Graph** definition. These are a connected network of edge definitions.

Finally a **View** definition is a way of filtering lists of properties that will be available in the user interface.

These object types are **not required** in order to publish data from an engineering authoring tool. They will only be used by SmartPlant Foundation once they have been loaded to the SPF Admin database.

Because of the way these objects are used by SmartPlant, the diagram below will illustrate the work process recommended to add the navigation objects.



Navigation objects are only used by the SmartPlant Foundation client and are not used by the integrated engineering tools. Because of this, EdgeDefs, GraphDefs and ViewDefs are delivered in a separate xml file called **EdgeGraphView.xml**. This file is included in the configuration along with the master schema (**EFSchema.xml**) and the project schema. Any custom navigation objects should be created in another schema file (**adcnavigation.xml** for example), separate from the original project schema.

To load the navigation object definitions into the SmartPlant Foundation admin database, the *EFSchema*, the *EdgeGraphView* extension schema, the *adc* extension schema and the custom navigation schema extension (*adcnavigation.xml*) will need to be included into a single schema configuration file.

The following describes the different schema files shown in the above figure:

EFSchema.xml – The SmartPlant “master” schema that is initially delivered with SmartPlant Enterprise.

EdgeGraphView.xml – A schema file delivered that contains the extensions for the out of the box EdgeDef, GraphDef and ViewDef navigation object types.

adc.xml – The custom project schema containing user-defined classes, interfaces, properties and relationships.

adcnavigation.xml – An additional custom extension schema containing user-defined navigation object types; EdgeDefs, GraphDefs, ViewDefs and ClassViewMaps.

5.1 Edge Definitions

Edge definitions are used to combine multiple relationships into a single definition and have the following properties:

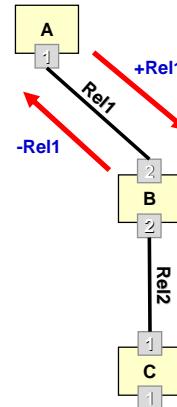
- A starting interface definition
- A path definition that specifies the relationships to traverse from the starting interface definition along related interface definitions and the direction to traverse those relationships (+ or -)



Edge Definitions

Edge definitions (EdgeDef) are single or multiple relationship definitions with direction. In the schema, an edge definition is used to traverse from a starting object to related objects.

Two implicit edges exist for every relationship 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).



© 2005, Intergraph Corp.
All Rights Reserved.

- Discrimination criteria that can be applied to objects at the end of the path
- Position criteria that can select a specific object from the set of objects that satisfy the discrimination criteria at the end of the path

When you create an interface definition in the Schema Editor, the software automatically creates a corresponding edge definition. This edge definition is given a UID and name that is a combination of + and the UID for the interface definition. When you create a relationship definition in the Schema Editor, two edge definitions exist: one for traversing from end 1 of the relationship to end 2 and the other for traversing from end 2 to end 1.

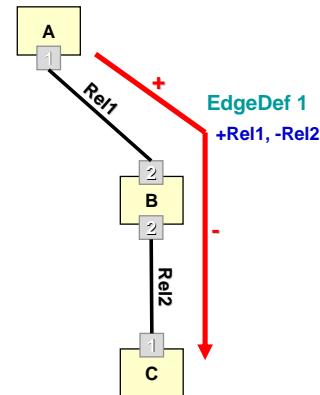
In addition to these automatically generated edge definitions (implicit EdgeDefs), you can explicitly define edge definitions for a variety of purposes in the Schema Editor. Explicit edge definitions may be used to traverse across multiple relationships, to select only certain objects from the destination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



Edge Definitions

EdgeDefs collect one or more relationship edges.

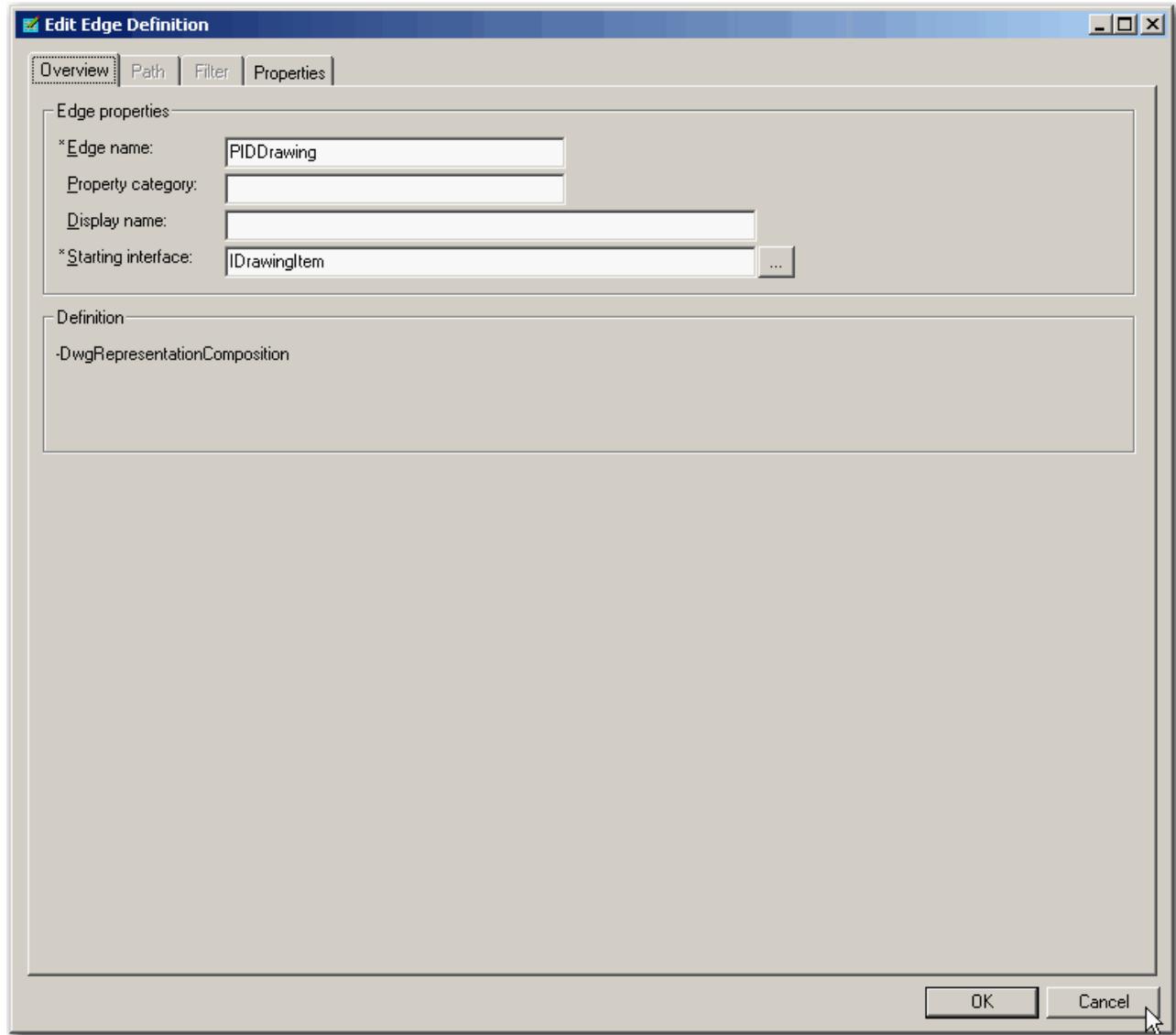
EdgeDefs can specify discrimination criteria, including position, to help you filter the objects at the end of the path defined by the EdgeDef.



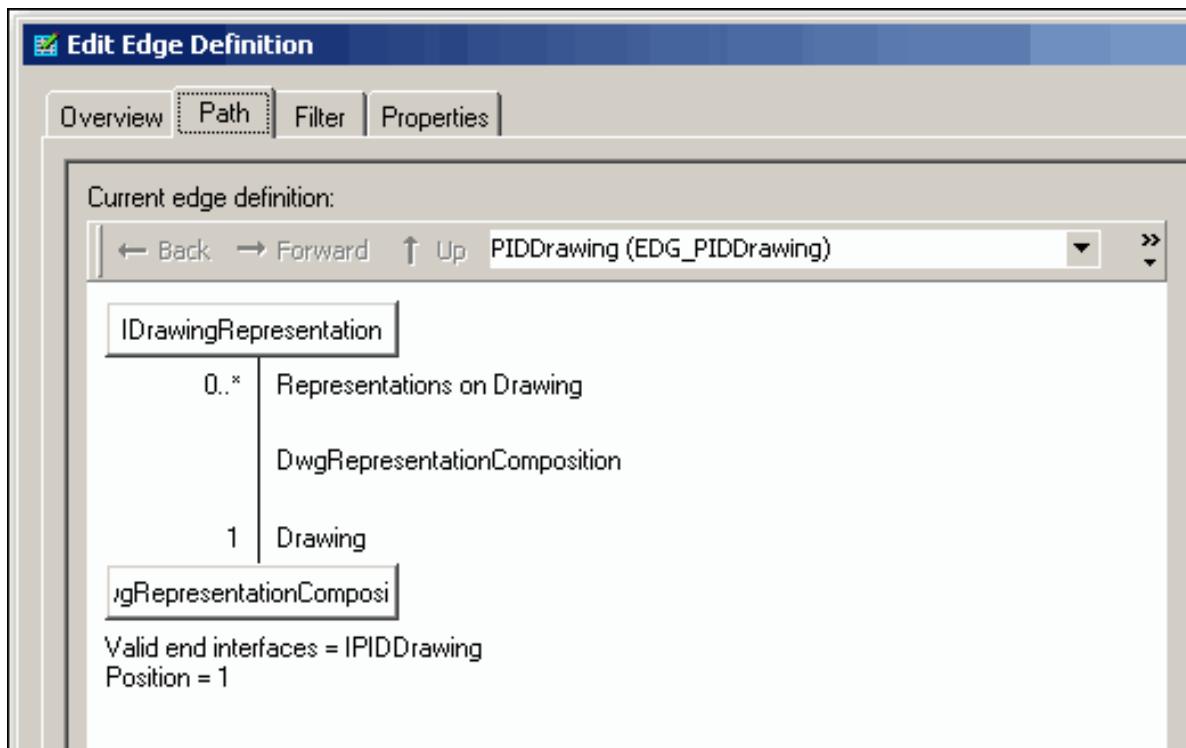
5.1.1 Properties of an Edge Definition

Edge definitions have the following properties:

- ❑ **Edge name** – Specifies the name of the edge definition.
- ❑ **Property category** - Defines the property category for properties contained in this edge definition. The property category helps organize properties in the **Properties** window in SmartPlant Foundation and SmartPlant P&ID.
- ❑ **Display name** – Specifies the name that you want the user interface to use when displaying the edge definition.

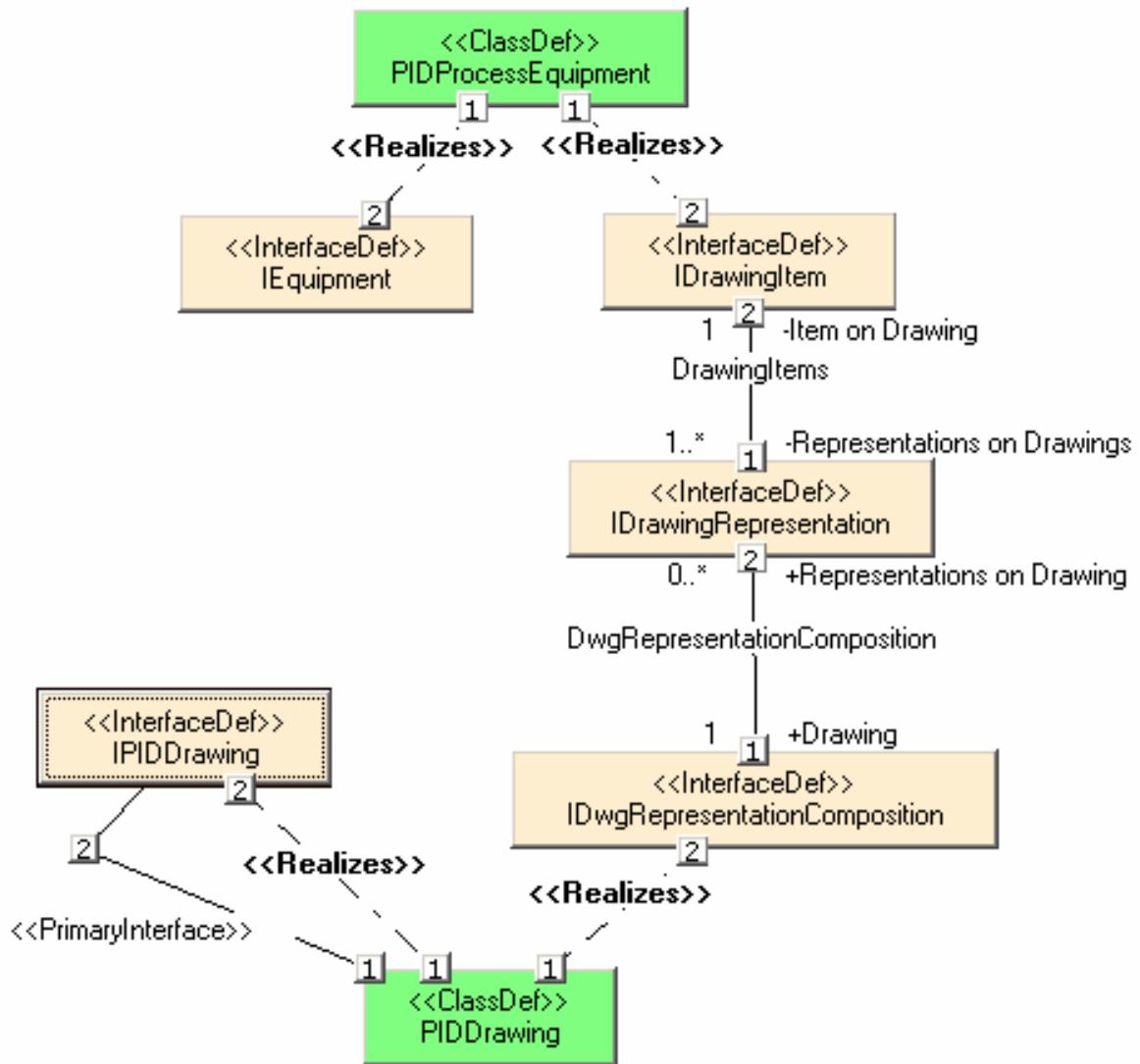


- ❑ **Starting interface** – Allows you to select the starting interface when you create a new edge definition.



- ❑ **Path** – Allows you to select the relationship edges that you want to include in the edge definition. The set of edge definitions that appear here are the set of interface definitions that apply to the end interface definition. As you click nodes in the tree, the path for the edge definition updates beside the **Path** label at the bottom of the dialog box. The UI allows you to include interface definitions that are implied by the end interface definition in your edge definition. Every edge definition starts at an interface definition and traverses through one or more relationship definitions to another interface definition at the other end. When you define a path for an edge definition, the set of edge definitions that appear in the **Current edge definition** window are the set of interface definitions that apply to the end interface definition. For example, if you traverse the relationship definition from equipment to equipment components, you will see the edge definitions that apply to the role of equipment components. However, if you want to continue the path using nozzles, which are a specialization of equipment components, then you can select INozzleOcc from this list. Selecting INozzle for example, will allow you to continue the path using the more specific nozzle edge definitions.

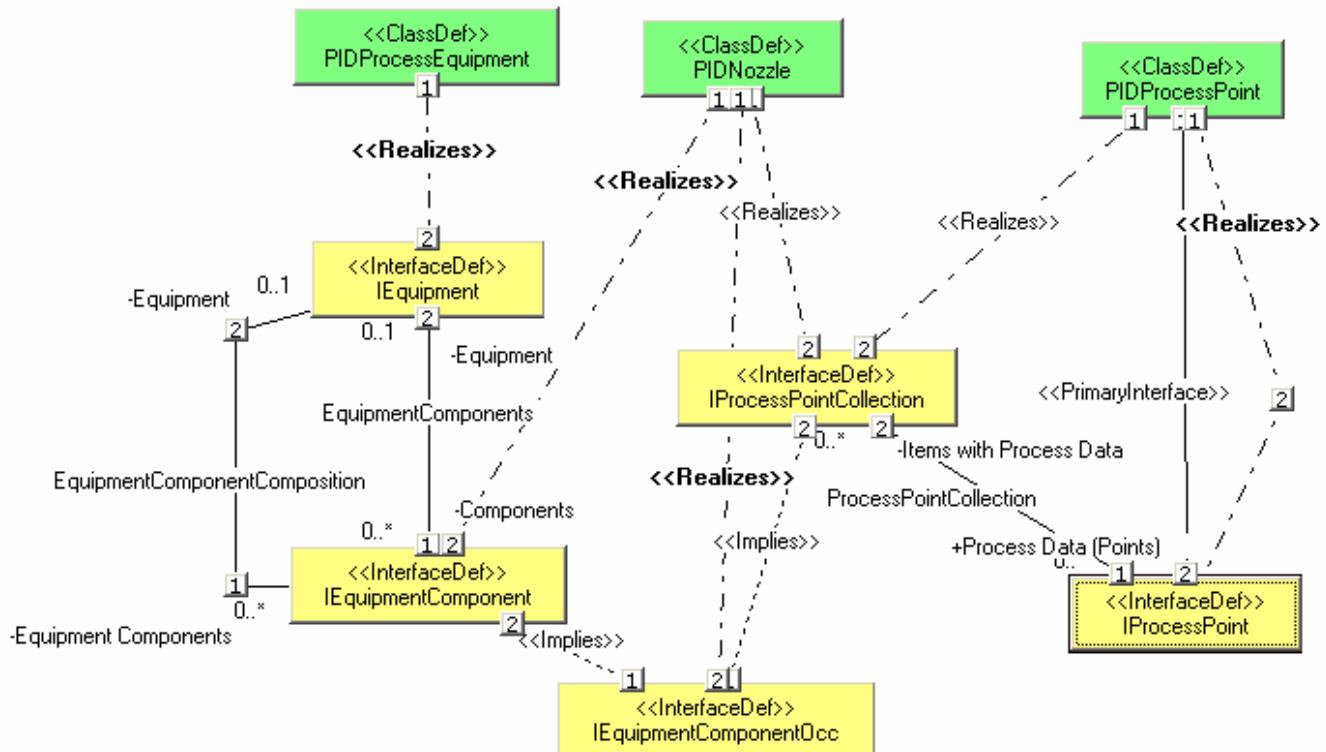
The following UML diagram shows an EdgeDef that navigates from a piece of equipment to the P&ID on which the piece of equipment appears:



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.

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.

The following UML diagram can be used to show what an edge definition might look like:



Edge Definitions

- Select the starting interface for the new EdgeDef (**IEquipment**) and click the Path tab.

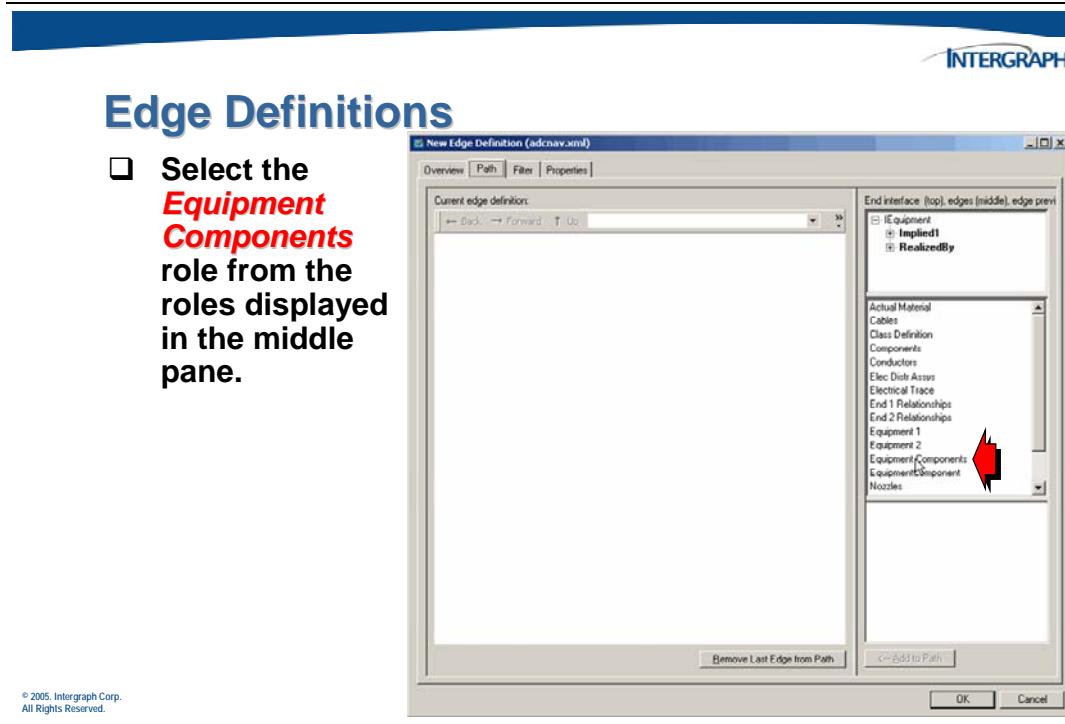
New Edge Definition (adcnav.xml)

Path (highlighted)

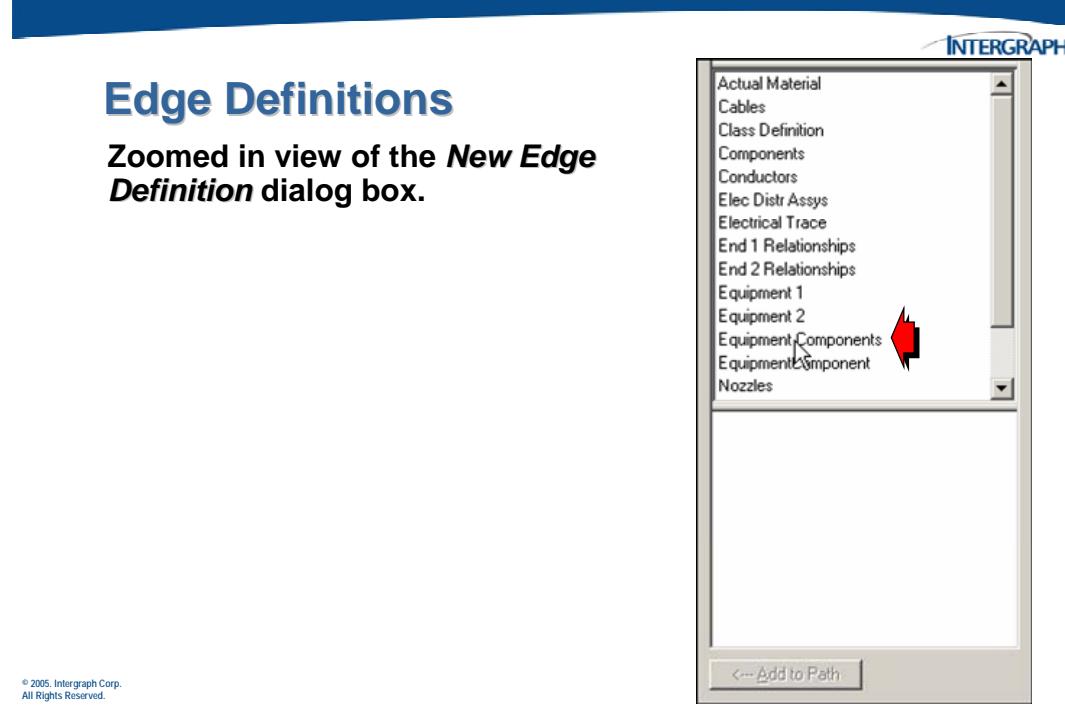
Edge name: [*]	Equipment2ProcessPoint
Property category:	
Display name:	Process Points
Starting interface: [*]	IEquipment

Definition

© 2005, Intergraph Corp.
All Rights Reserved.



© 2005. Intergraph Corp.
All Rights Reserved.

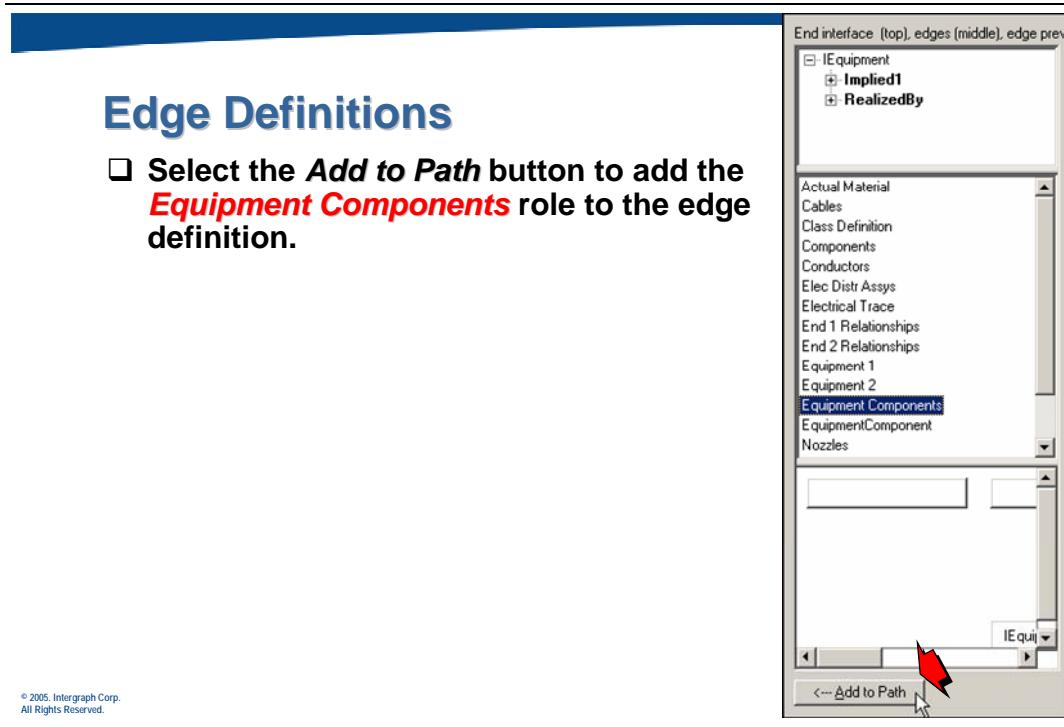


© 2005. Intergraph Corp.
All Rights Reserved.

Edge Definitions

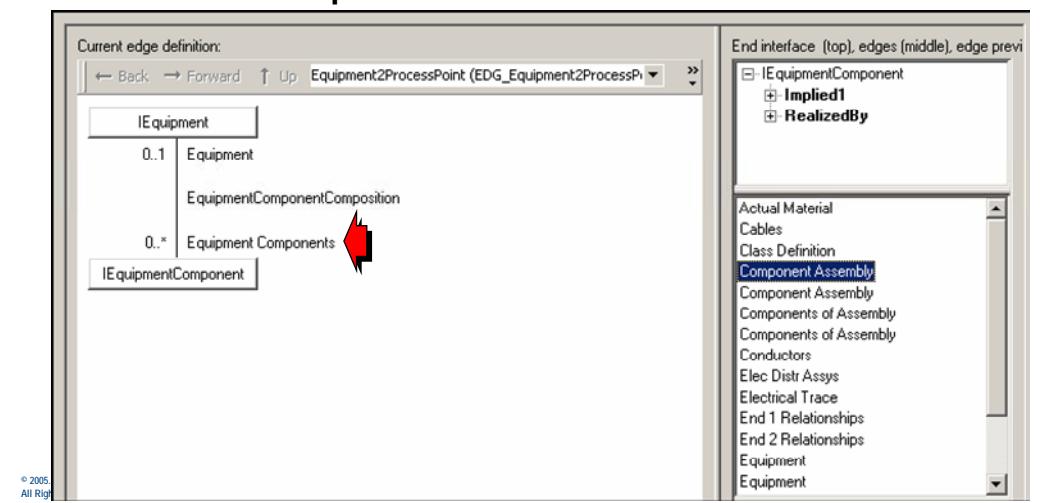
- Select the **Equipment Components** role from the roles displayed in the middle pane.

Click on the *Equipment Components* role in the middle pane.



Edge Definitions

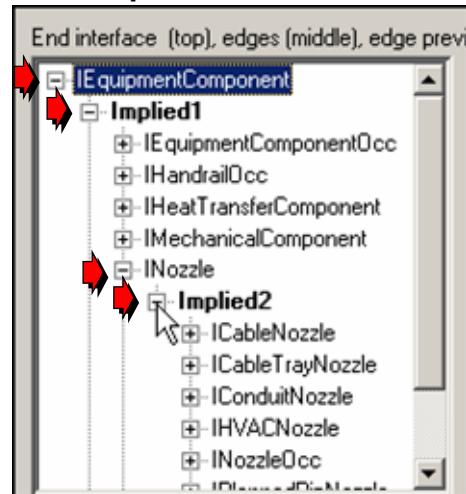
The *Equipment Components* role will be displayed in a UML view in the left pane.





Edge Definitions

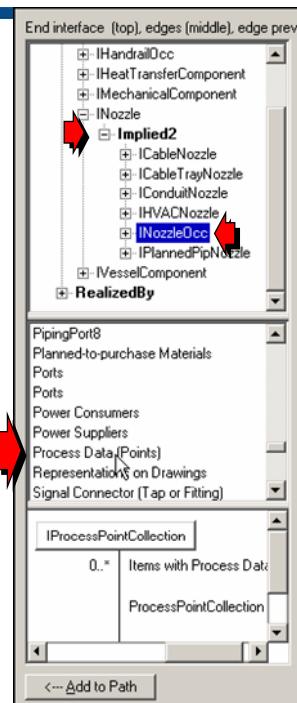
- Expand the **IEquipmentComponent** tree and display the implied relationships.



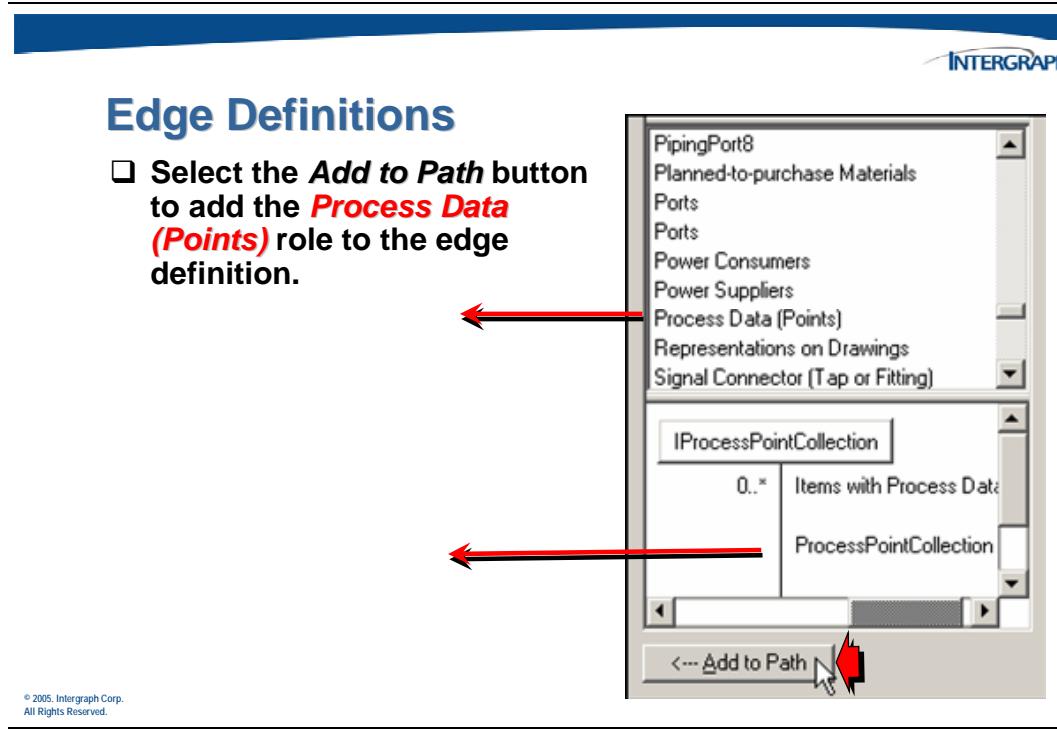
Drill down in the tree to locate the *INozzleOcc* implied relationship.

Edge Definitions

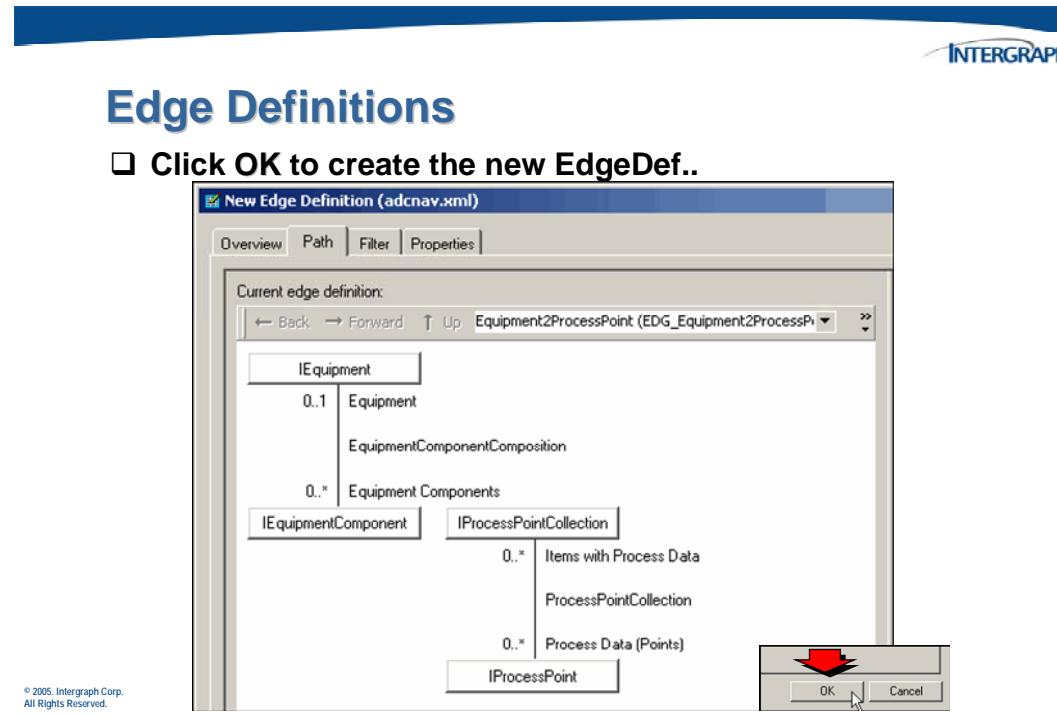
- Select the implied interface and then the ending role for this new EdgeDef (**Process Data (Points)**).



© 2005, Intergraph Corp.
All Rights Reserved.

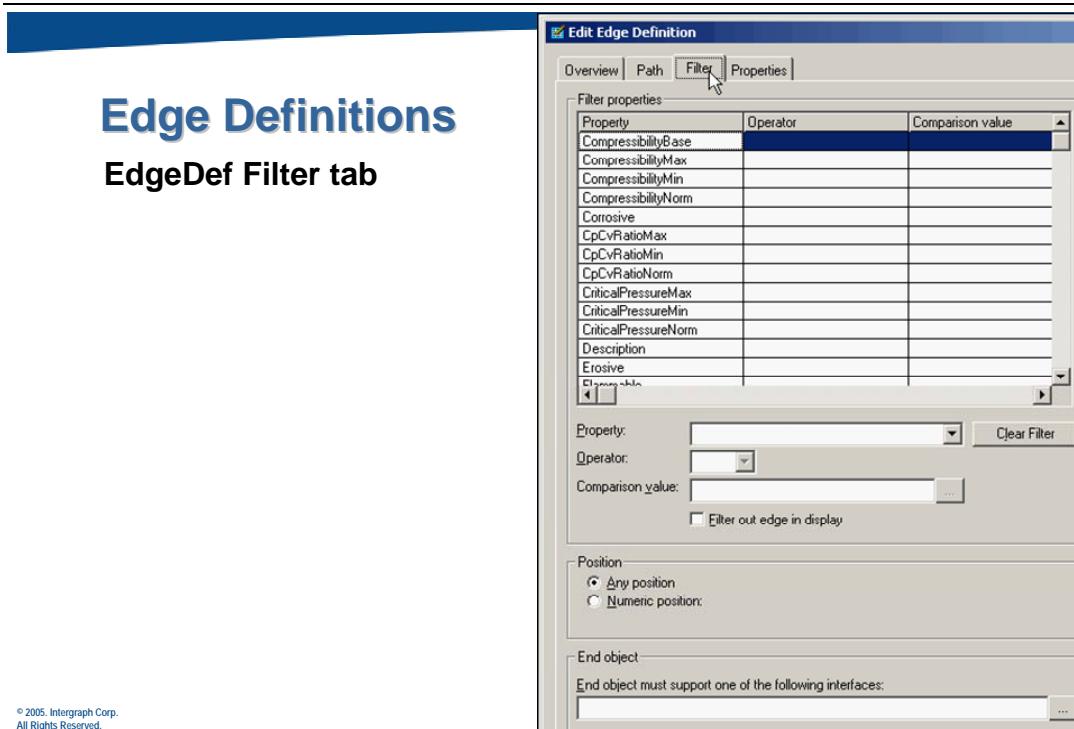


© 2005, Intergraph Corp.
All Rights Reserved.



© 2005, Intergraph Corp.
All Rights Reserved.

You can also define discrimination criteria for edge definitions using the options on the **Filter** tab.

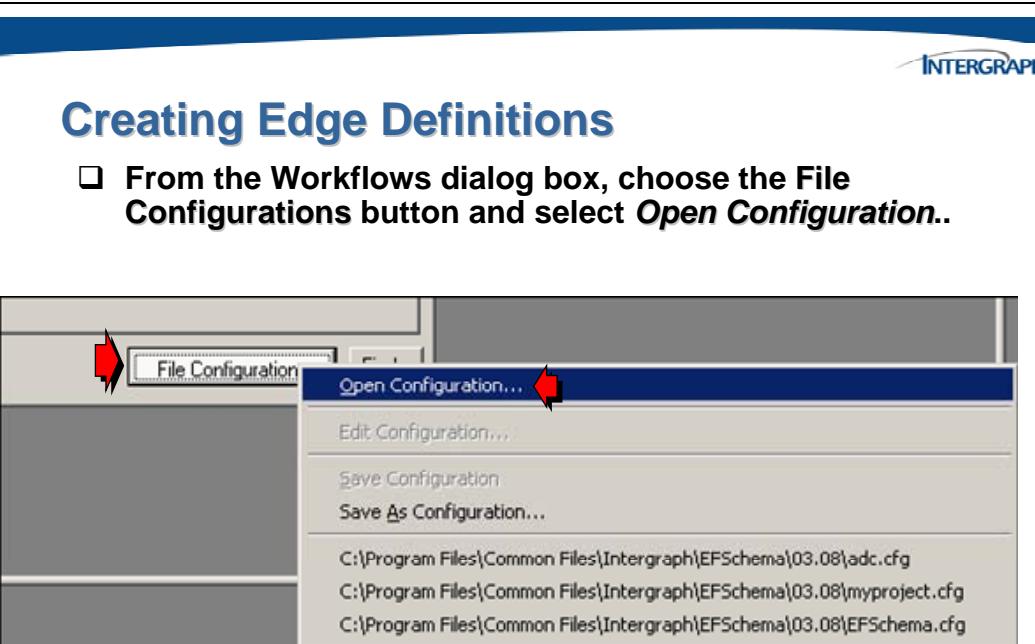


- Property** – Allows you to select properties exposed by the starting interface to use as criteria to limit the objects at the end of the path for the edge definition.
- Operator** – Specifies whether you want objects at the end of the path for the EdgeDef to be equal to (=), less than (<), greater than (>), or not equal to (<>) the value in the **Comparison value** box.
- Comparison value** – Specifies the value that you want to use with the specified comparator to limit objects at the end of the edge definition. If you select a property in the **Property** list that has an enumerated list associated with it, you can click the ? button here to select a value from the enumerated list to use as the comparison value.
- Filter properties** – Displays the criteria you selected for the edge definition.
- Clear Filter** – Removes all specified criteria from the table.
- Any position** – Specifies that objects at the end of the path for the edge definition can be in any position.
- Numeric position** – Specifies that objects at the end of the path for the edge definition must be in a particular numeric position. For example, if you want to return the fifth nozzle on a piece of equipment, the numeric position would be 5.
- Named position** – Specifies that you want to see objects at the end of the path that are in a named position. For example, named positions can include first and last.

- Filter out edge in display** – Specifies whether you want to see the edge definition in the Schema Editor user interface. If this option is checked (✓) which is True, user access for this edge definition can be controlled in the SPF client by user group security.
- End object must support the following interfaces** – Allows you to further define the objects at the end of the path for the edge definition by specifying an interface that end objects must support. For example, if you want to create an EdgeDef 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 DwgRepresentationCompostion relationship are returned by the EdgeDef.

5.2 Interactive Activity – Creating Edge Definitions

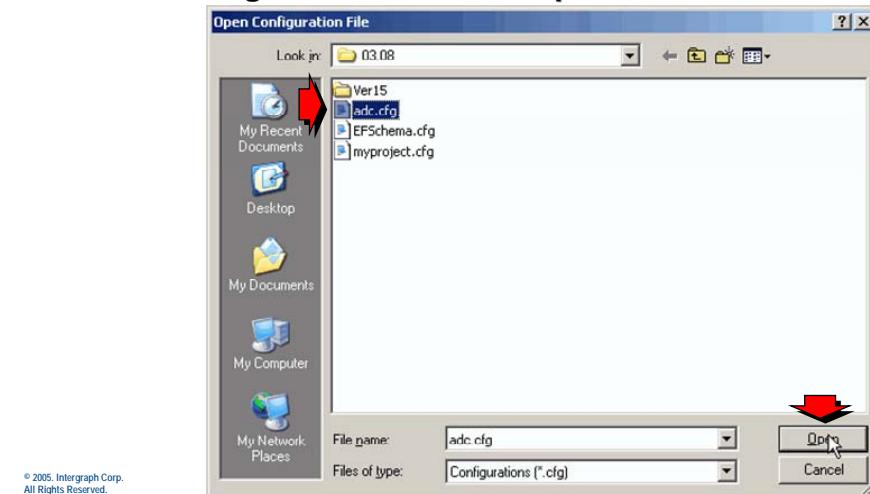
The following is an example of an **EdgeDef** with property criteria. Add the EdgeDef **NavProjectToOpenDesignChanges**. We cannot just rely on the RelDef because we only want open design changes, so we create an EdgeDef to test the status.





Creating Edge Definitions

- In the **Open Configuration File** dialog, select the adc configuration then click Open.

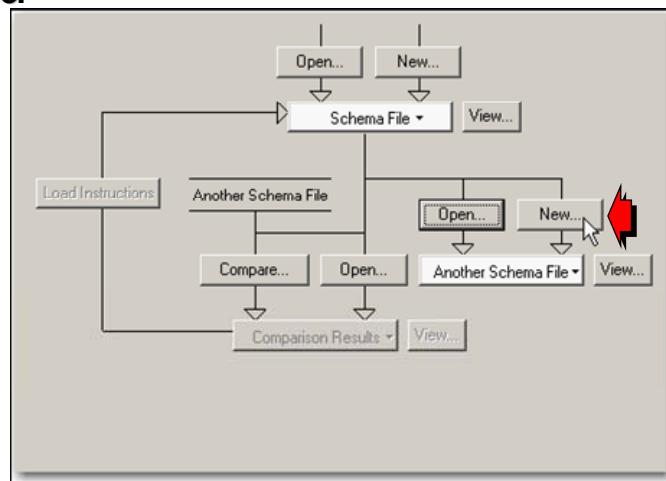


EdgeDef, *GraphDef* and *ViewDef* extensions are defined in a separate xml file. The **EdgeGraphView.xml** file will be delivered to the same folder as the EFSchema.

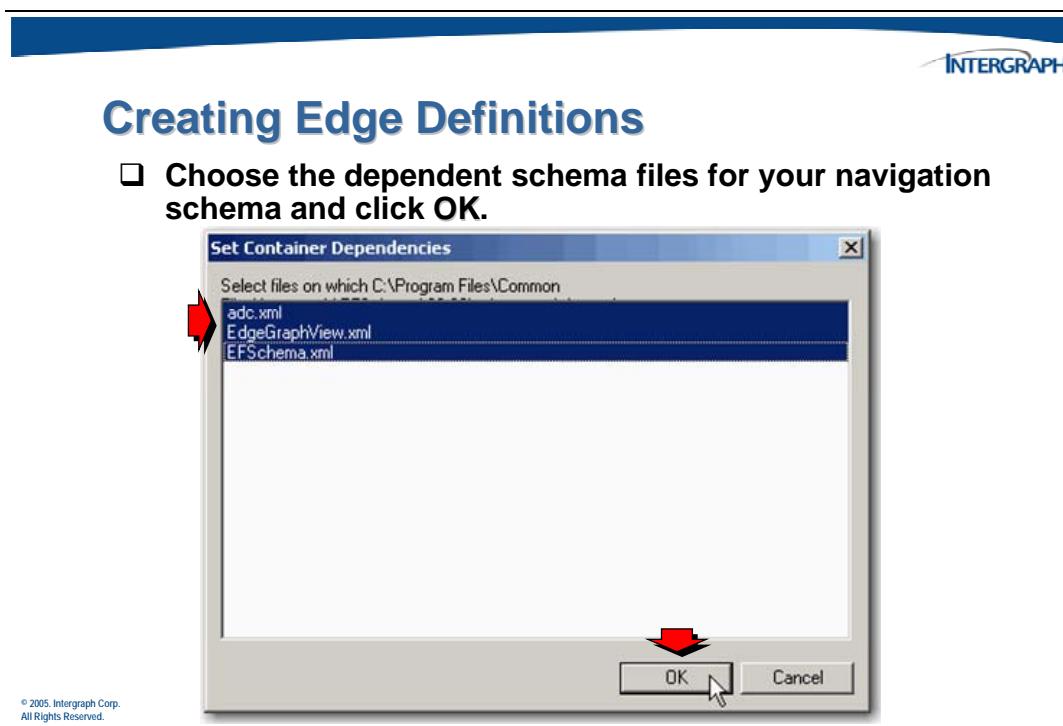
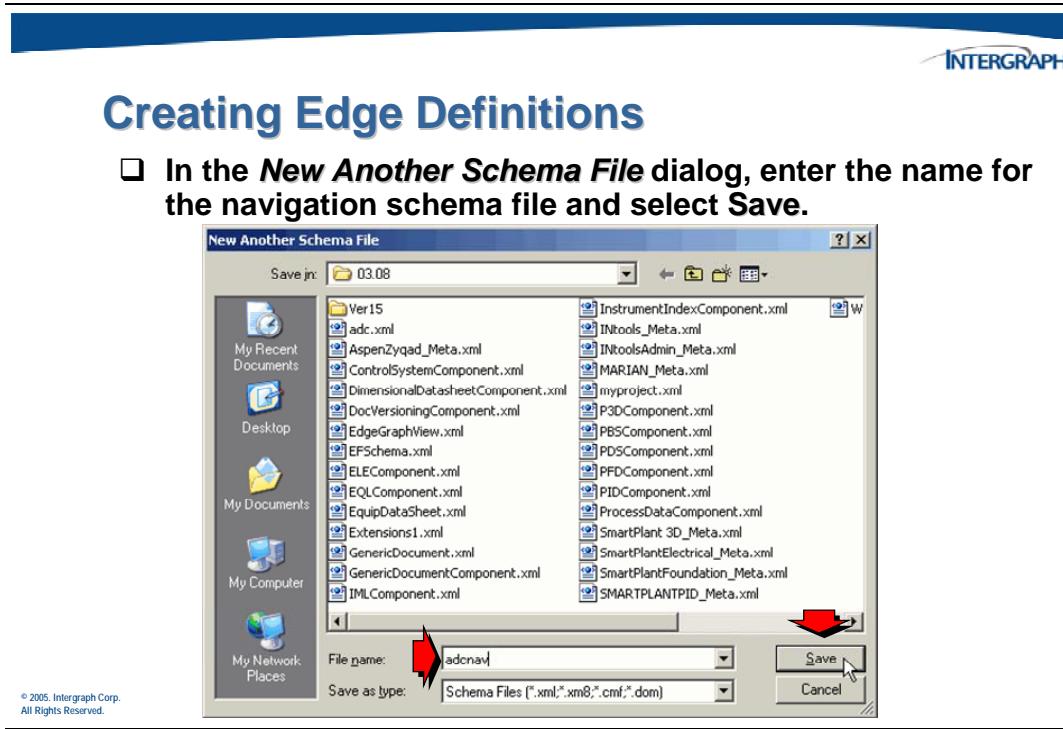


Creating Edge Definitions

- Create a new container schema for the navigation object types by selecting the New button above **Another Schema File**.



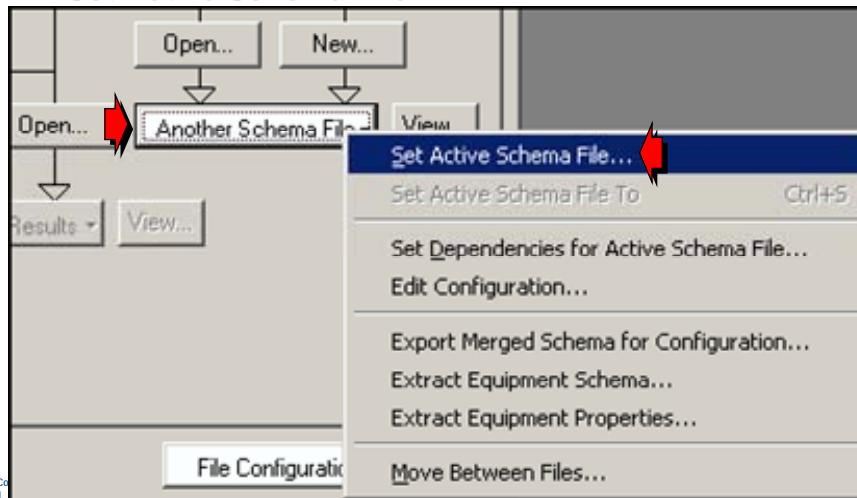
Since this schema will contain navigation object types, name the file **adcnav.xml**.





Creating Edge Definitions

- Click on the bolded **Another Schema File** and then select **Set Active Schema File...**

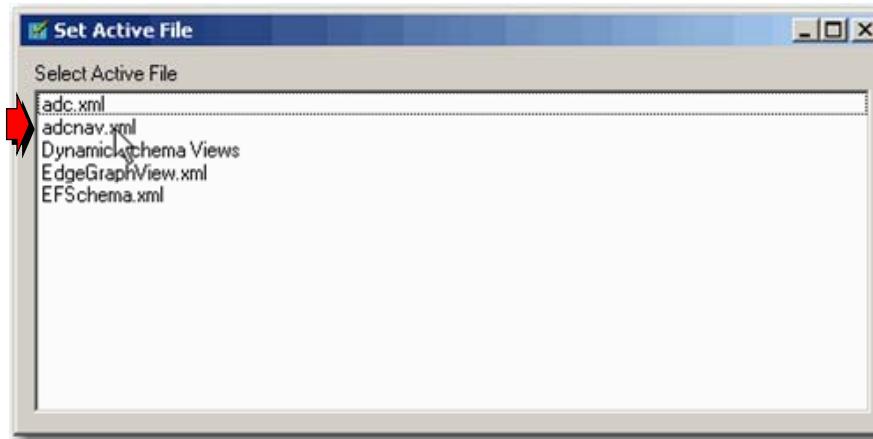


Make sure the active schema is set correctly.



Creating Edge Definitions

- Choose the navigation schema to set it as the Active File..



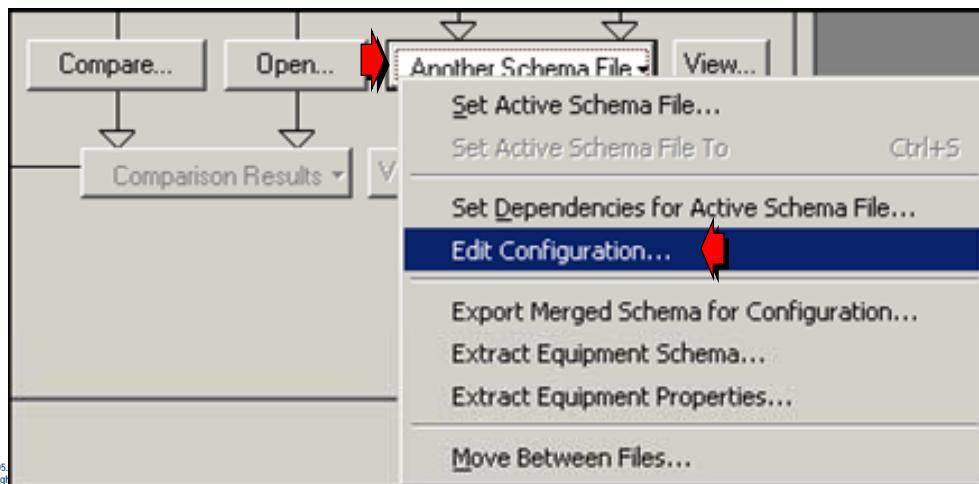
© 2005, Intergraph Corp.
All Rights Reserved.

Verify the configuration for all the schema files and make any necessary changes.



Creating Edge Definitions

- Click on the bolded **Another Schema File** button, then select *Edit Configuration...*

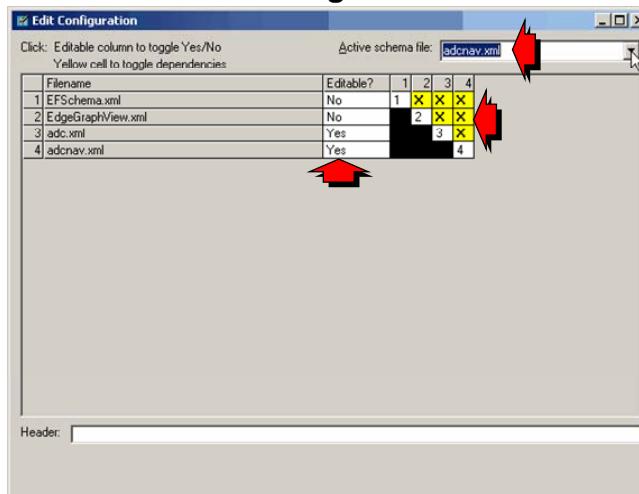


Make sure the dependencies (an X in the yellow boxes) are correct.



Creating Edge Definitions

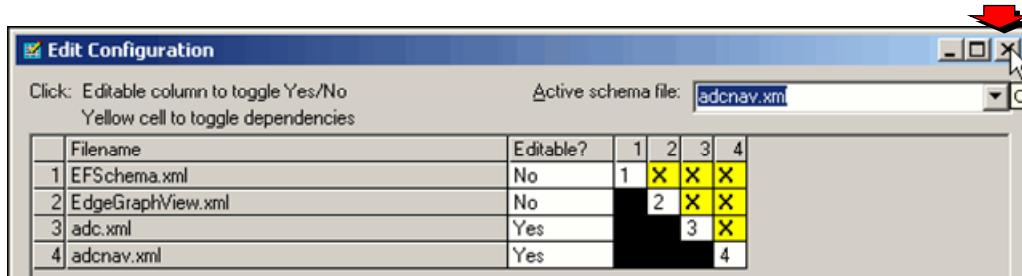
- Click in the **Editable?** Field to set the permissions on the schema files in this configuration.





Creating Edge Definitions

- Click the X to close the *Edit Configuration* dialog.



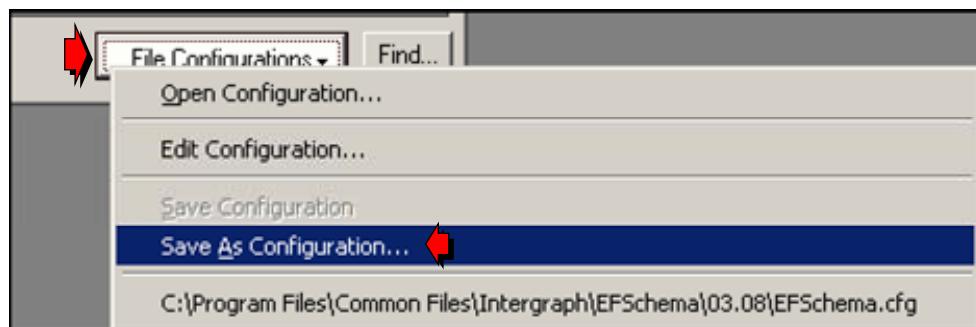
© 2005, Intergraph Corp.
All Rights Reserved.

Since other xml files have been added to the configuration, it can be saved as the same configuration file or as a different configuration.



Creating Edge Definitions

- Click the File Configuration button in the lower right corner of the window and choose *Save As Configuration*.

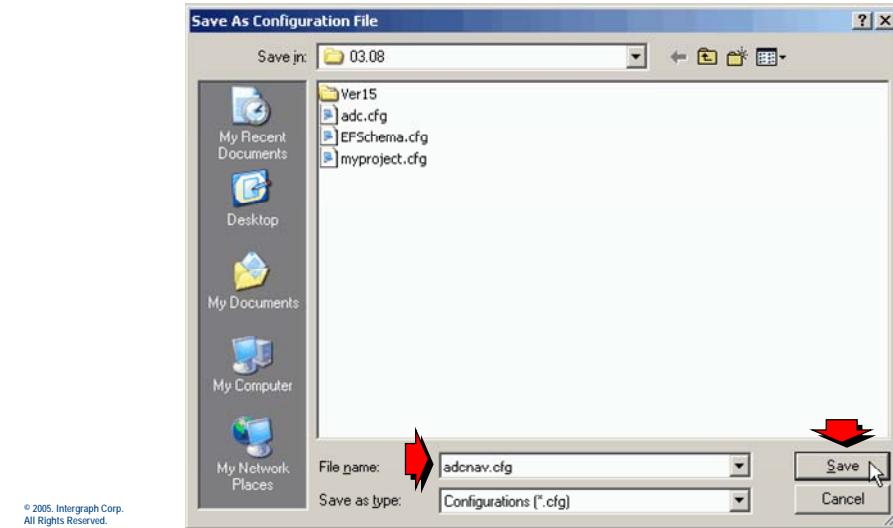


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

- Enter a name for this configuration and click Save.

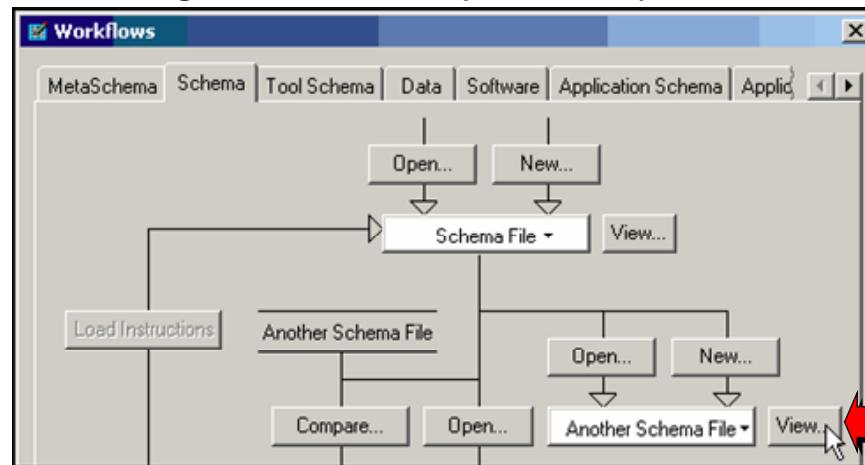


In this example, the new configuration will be named **adnav.cfg**.



Creating Edge Definitions

- In the **Workflows** dialog box, click the **View** button to view the navigation schema file (**adnav.xml**).



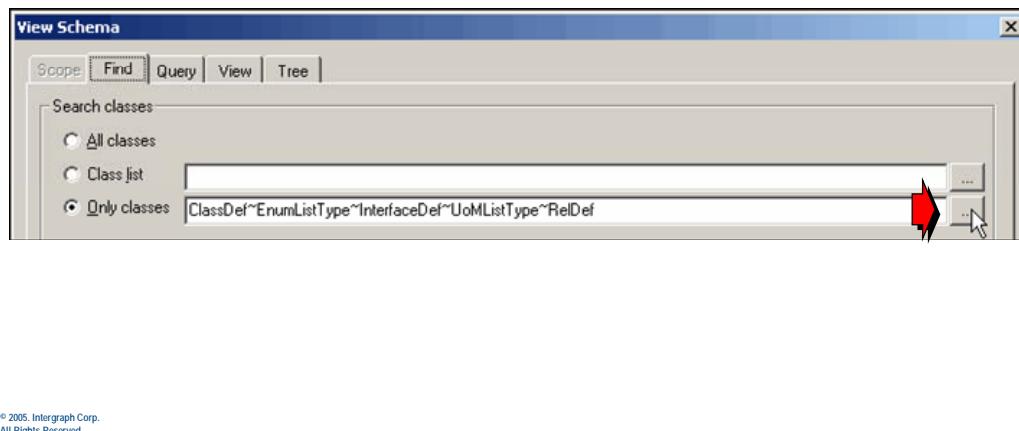
© 2005, Intergraph Corp.
All Rights Reserved.

This will allow you to view *EFSchema.xml*, *adc.xml*, *EdgeGraphView.xml* and edit *adcnav.xml*.



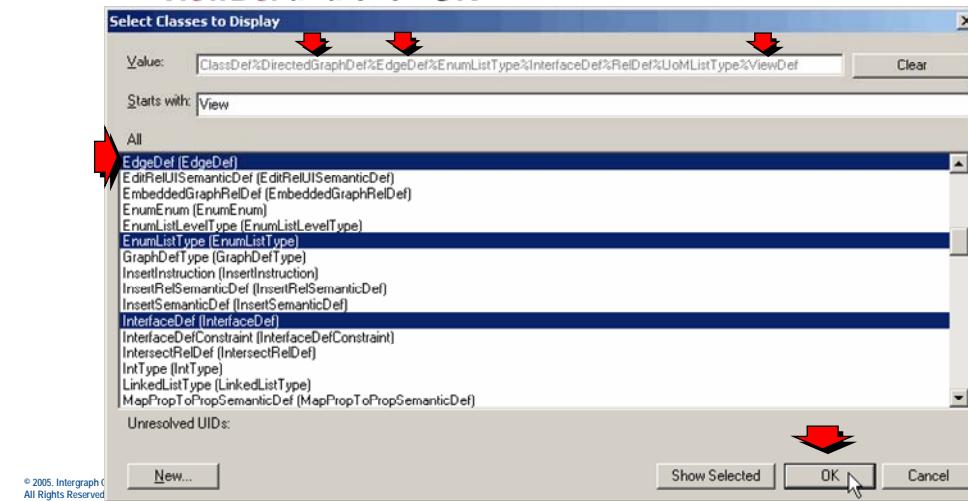
Creating Edge Definitions

- Use the **Only classes** browse (...) button to add the navigation classes.

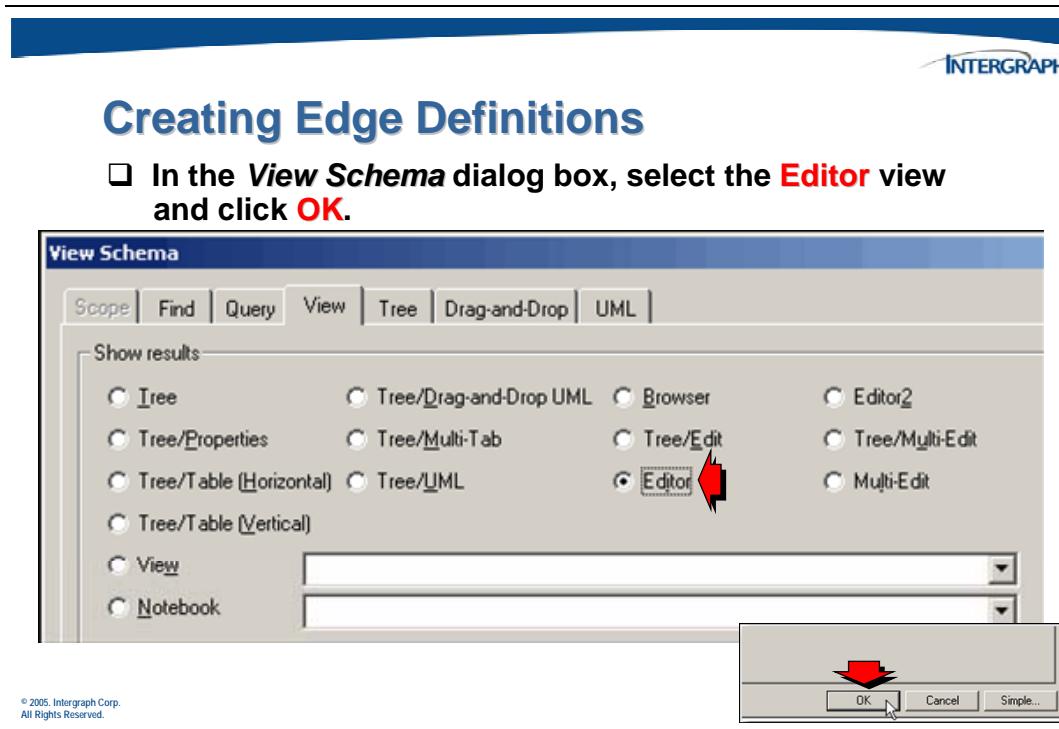
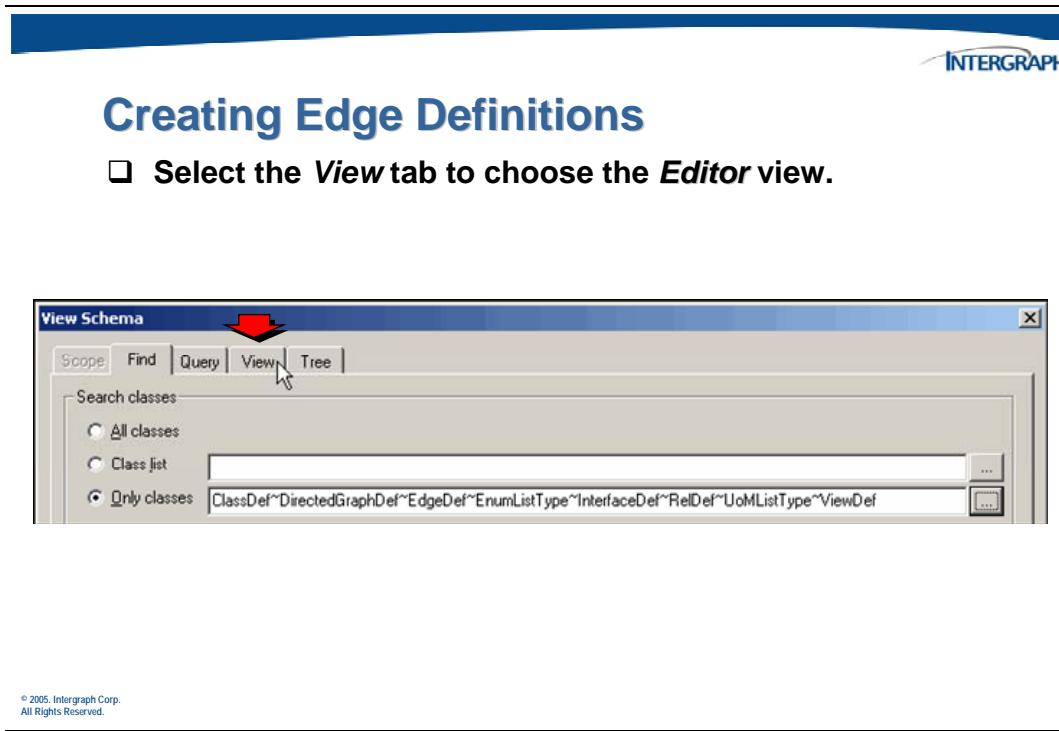


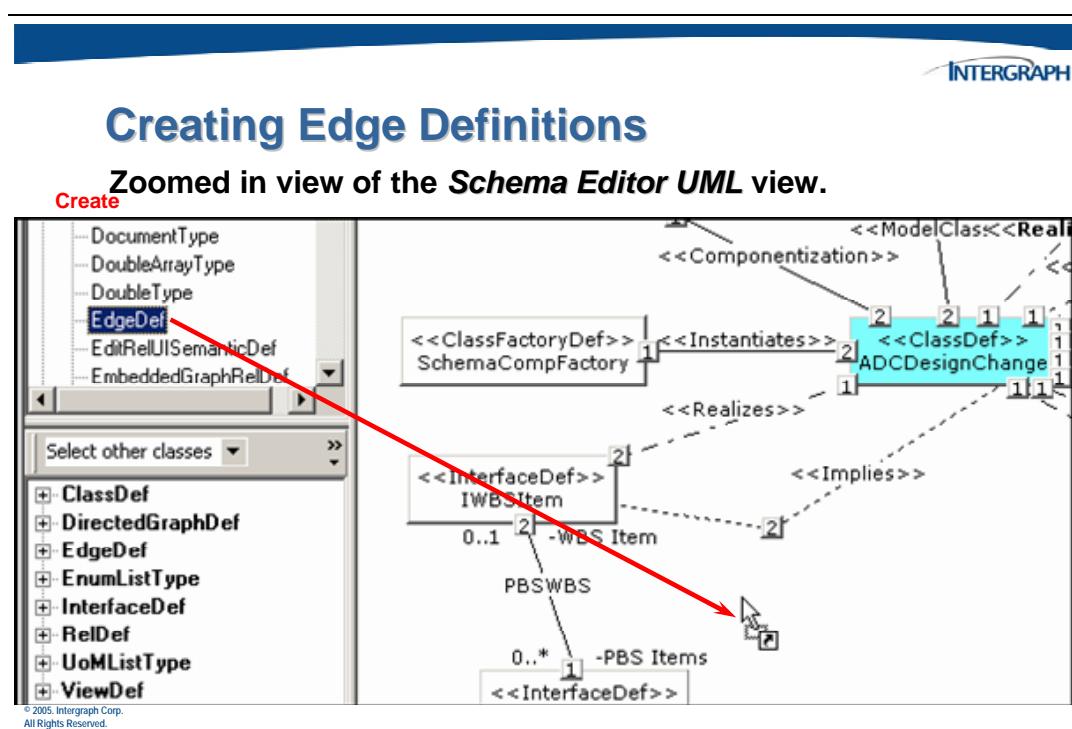
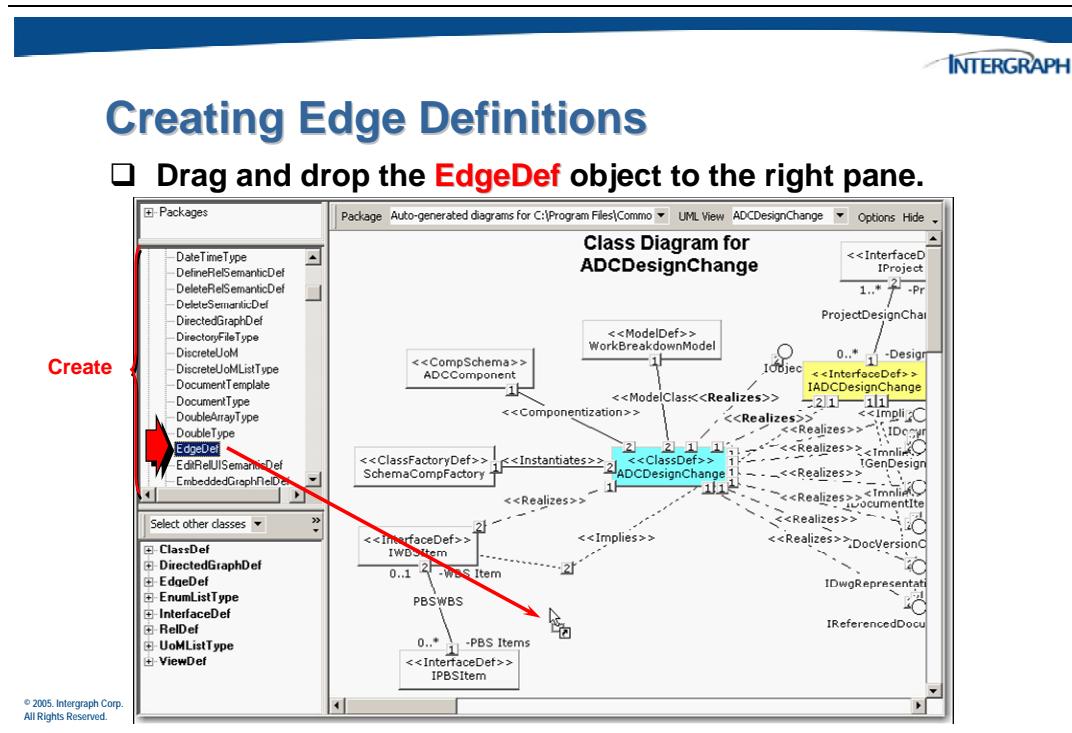
Creating Edge Definitions

- Select the classes **DirectedGraphDef**, **EdgeDef**, and **ViewDef** and click OK.



This will add the navigation object types to the view tree.

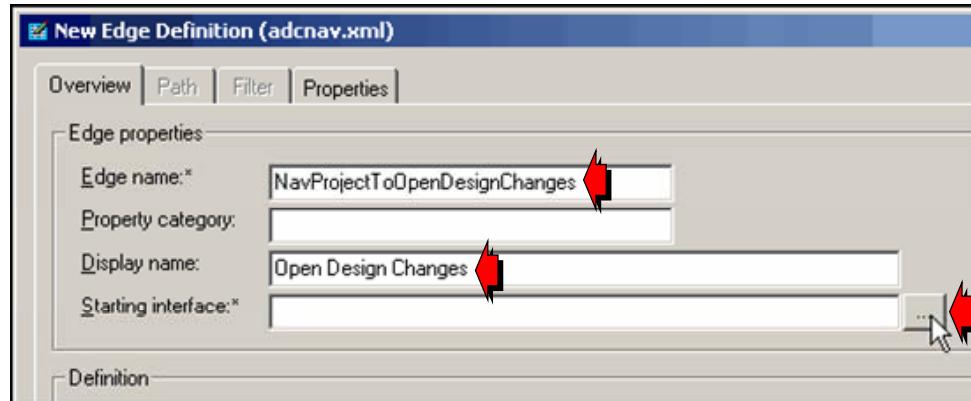






Creating Edge Definitions

- Enter an **Edge name** and **Display name** then click the **Starting interface** browse (...) button.



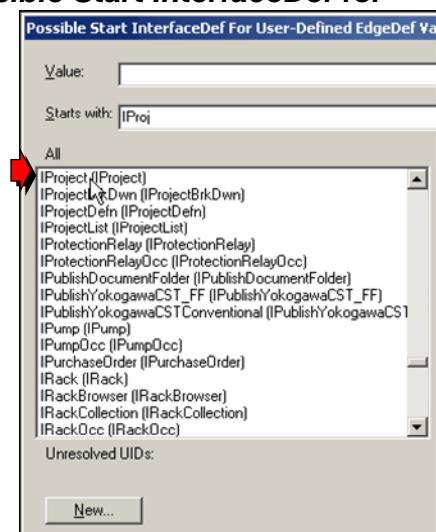
© 2005. Intergraph Corp.
All Rights Reserved.

A Possible Start InterfaceDef for User-Defined EdgeDef dialog will display.



Creating Edge Definitions

- Click on **IProject** from the **Possible Start InterfaceDef for User-Defined EdgeDef** dialog.

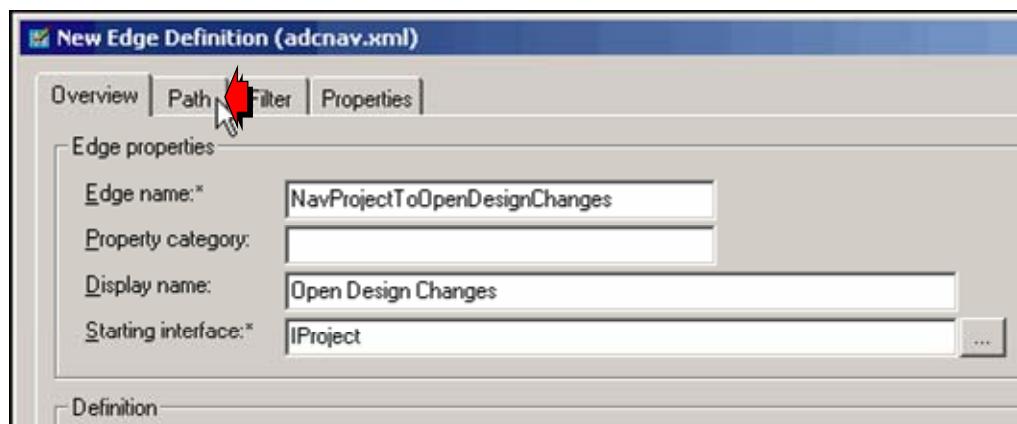


© 2005. Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

- Select the **Path** tab to specify the relationship edges for this new EdgeDef.

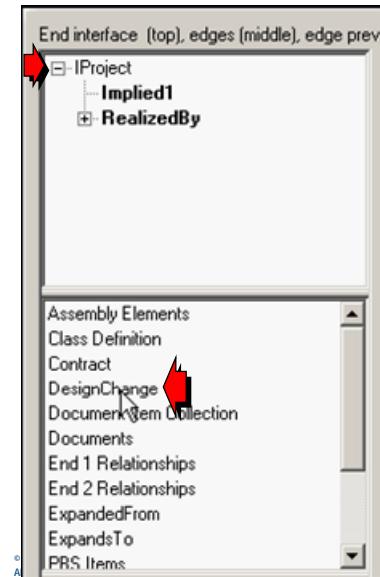


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

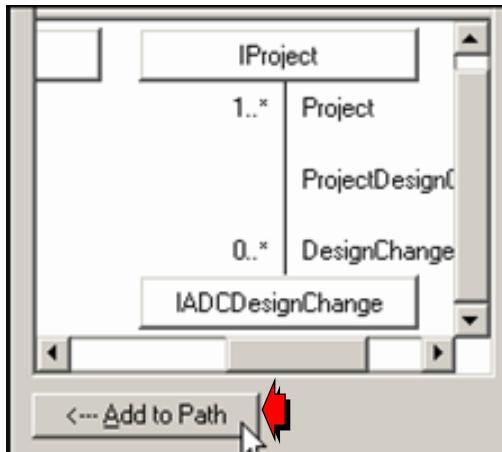
- Click to expand **IProject** in the top right pane and select the **DesignChange** role from the middle pane.





Creating Edge Definitions

- Select the **Add to Path** button to add the **ProjectDesignChange** role to the edge definition.

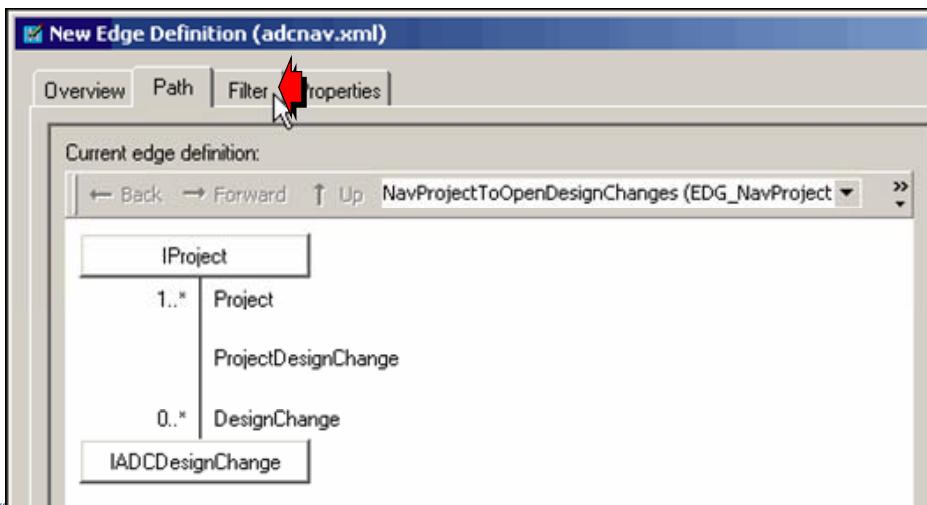


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

- Select the **Filter** tab to specify edge criteria.



© 2005, Intergraph Corp.
All Rights Reserved.

Select the Property list button and choose a property name.

Use the *GenDesignedItemStatus* property to define EdgeDef criteria.

Choose an Operator and then a Comparison value.

UID	Short Description	Long Description
Created	CR	Created
InProgress	IP	InProgress
Required	REQ	Required
Approved	AP	Approved
Completed	COM	Completed

Creating Edge Definitions

Select OK to define the new EdgeDef.

New Edge Definition (adcnav.xml)

Overview Path Filter Properties

Filter properties

Property	Operator	Comparison value
ADCApprovalSignature		
ADCDetail		
ADCImpact		
ADCMeas		
ADCProposalSignature		
Description		
DocCategory		
DocSubtype		
DocTitle		
DocType		
ExpansionCount		
GenDesignedItemStatus	=	@InProgress
Name		

Property: GenDesignedItemStatus
Operator: =
Comparison value: @InProgress
 Filter out edge in display

OK Cancel

© 2005, Intergraph Corp.
All Rights Reserved.

Add the EdgeDefs **NavDesignChangeToEquipment**, **NavDesignChangeToInstruments**, **NavDesignChangeToLines**. The following are examples of EdgeDefs using the ending interface criteria. We rely on a very generic relationship between DesignChange and Tags: PBSItemCollection. Navigating that would return Equipment, Instruments, & Lines (and basically any PBSItem related to it.) We specify ending interface criteria for each of these that selects only what we want.

Creating Edge Definitions

Repeat the same sequence to define another edge by dragging and dropping EdgeDef from the Create window into the UML view window.

DoubleType
EdgeDef
EditRelUISemanticDef
EmbeddedGraphRelDef

Select other classes <>

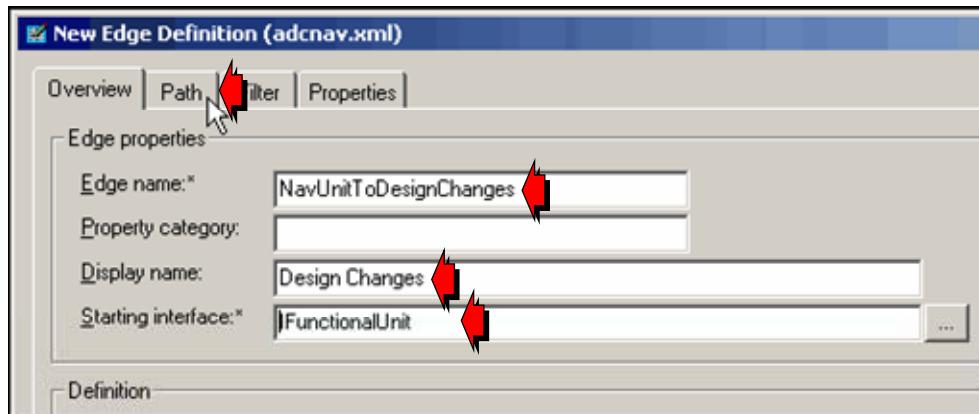
ClassDef
DirectedGraphDef

© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

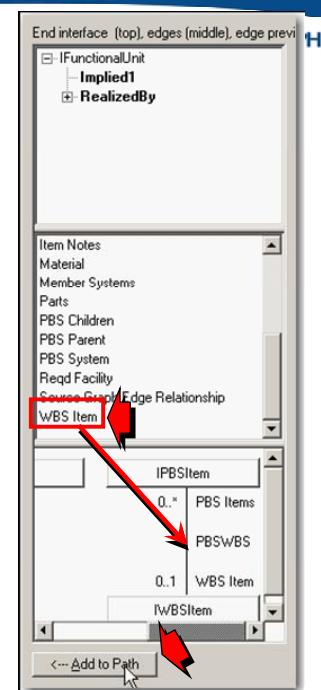
- Enter a name and display name for the next edge then select the *Path* tab.



© 2005, Intergraph Corp.
All Rights Reserved.

Edge Definitions

- Select the WBS Item role and then click the *Add to Path* button to add the **WBS Item** role to the edge definition.

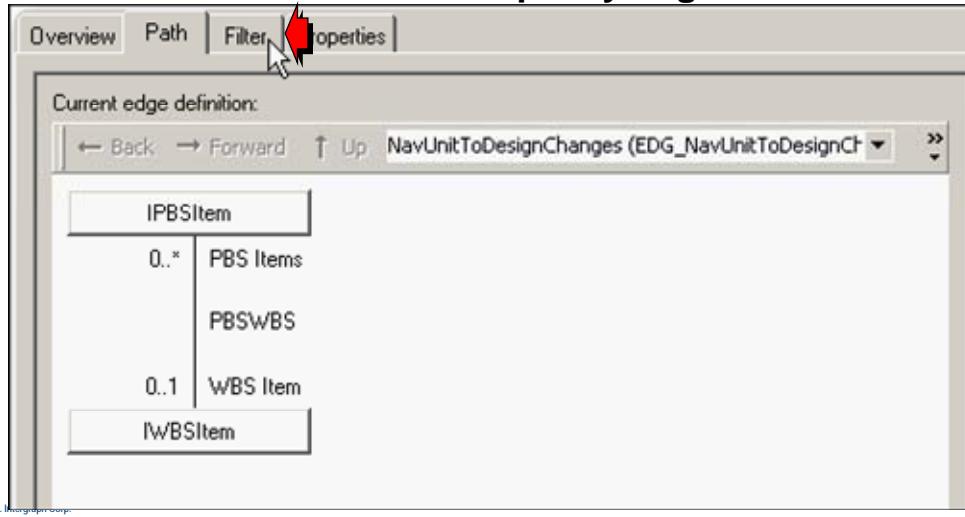


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

- Select the **Filter** tab to specify edge criteria.

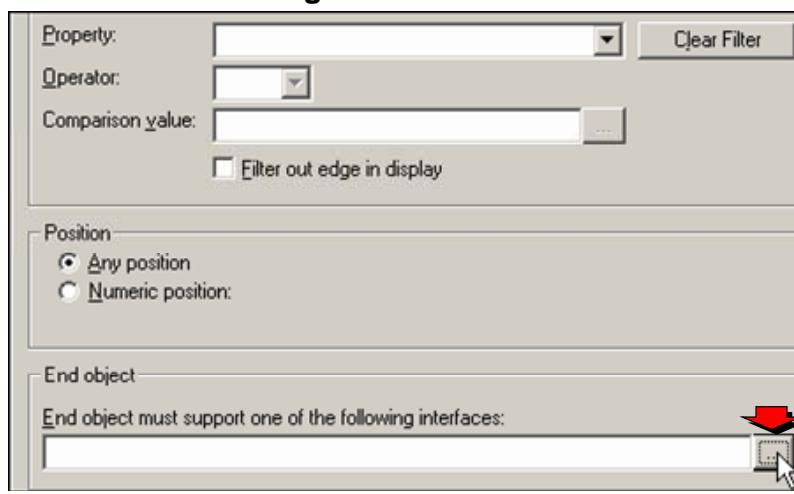


In this case, the criteria will be a specific ending interface.



Creating Edge Definitions

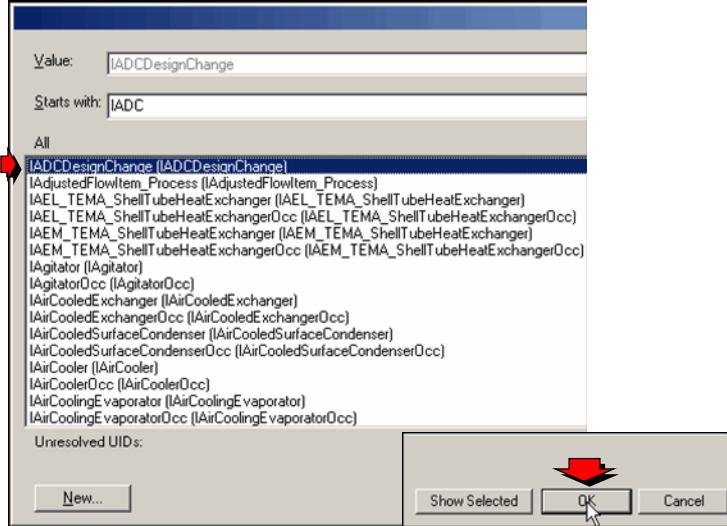
- Select the End object browse button and choose a ending interface for the edge.





Creating Edge Definitions

- Select the **IADCDesignChange** interface from the displayed list.

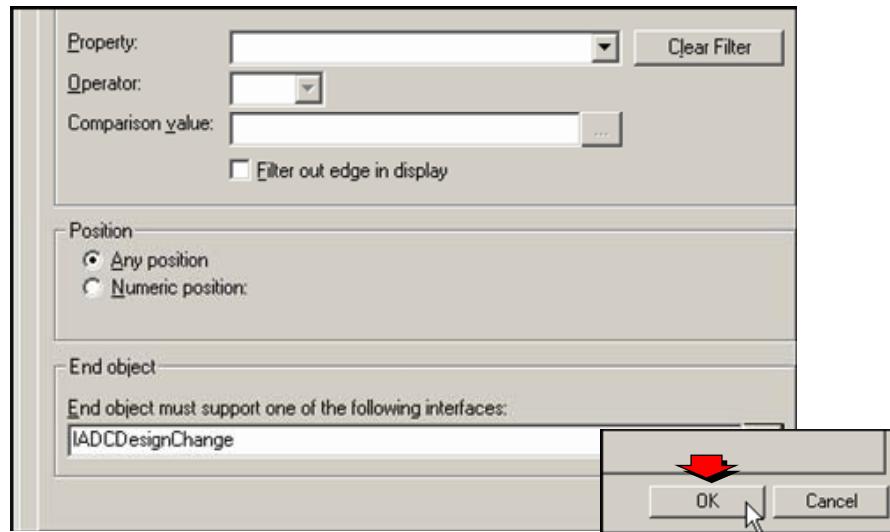


Click **OK** to choose this as the ending interface for the EdgeDef.



Creating Edge Definitions

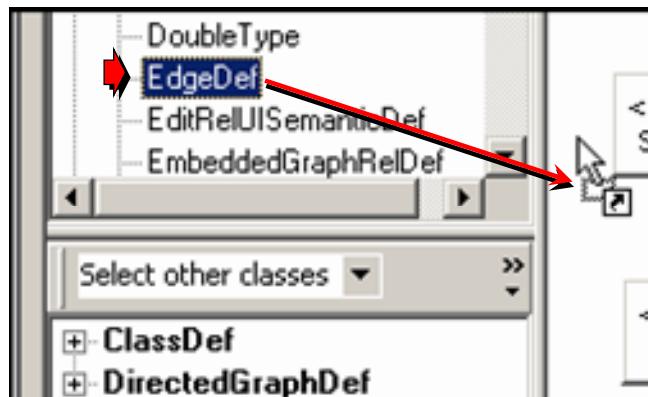
- Select **OK** to define the new edge.





Creating Edge Definitions

- ❑ Repeat the same sequence to define another edge by dragging and dropping EdgeDef from the Create window into the UML view window.

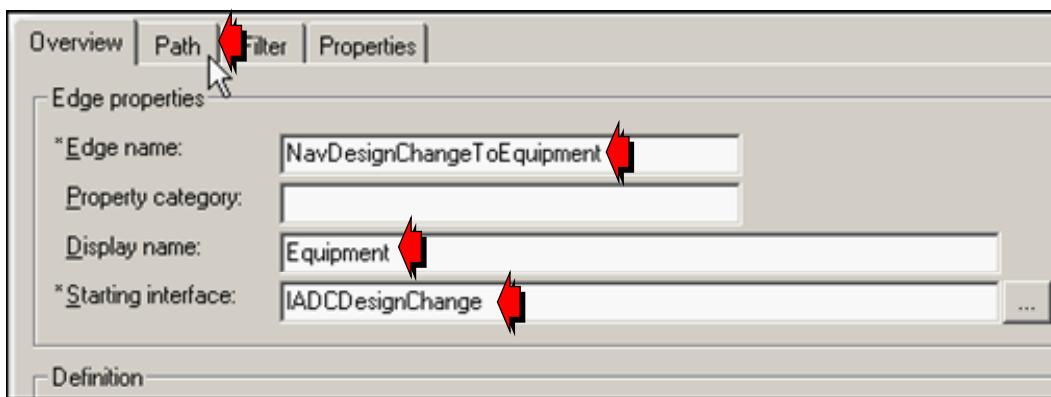


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

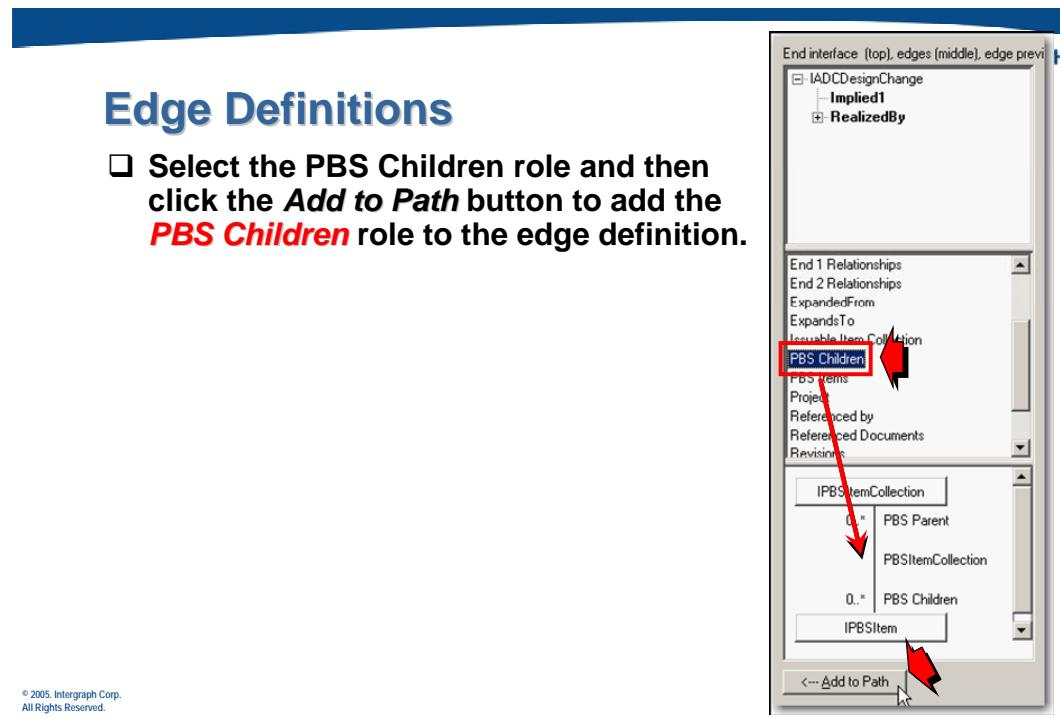
- ❑ Enter a name and display name for the next edge then select the *Path* tab.



© 2005, Intergraph Corp.
All Rights Reserved.

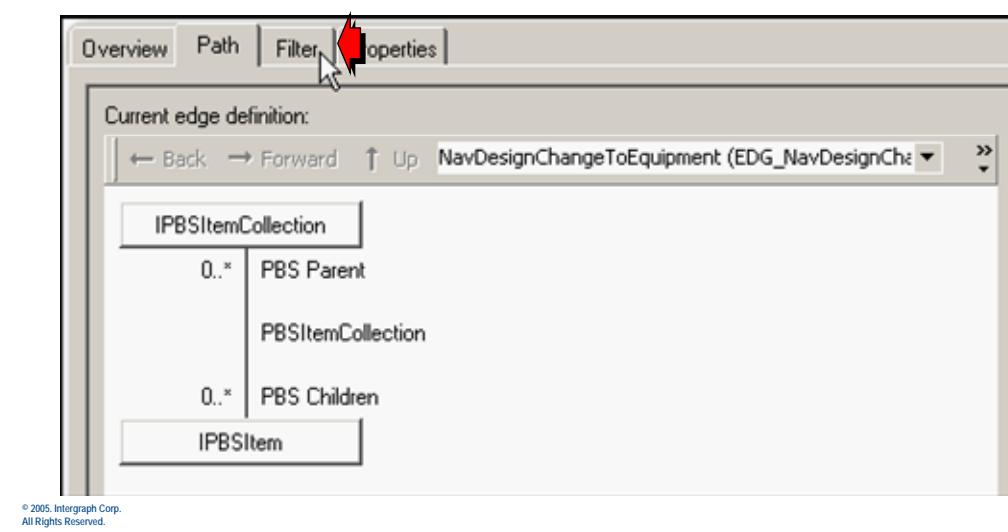
Edge Definitions

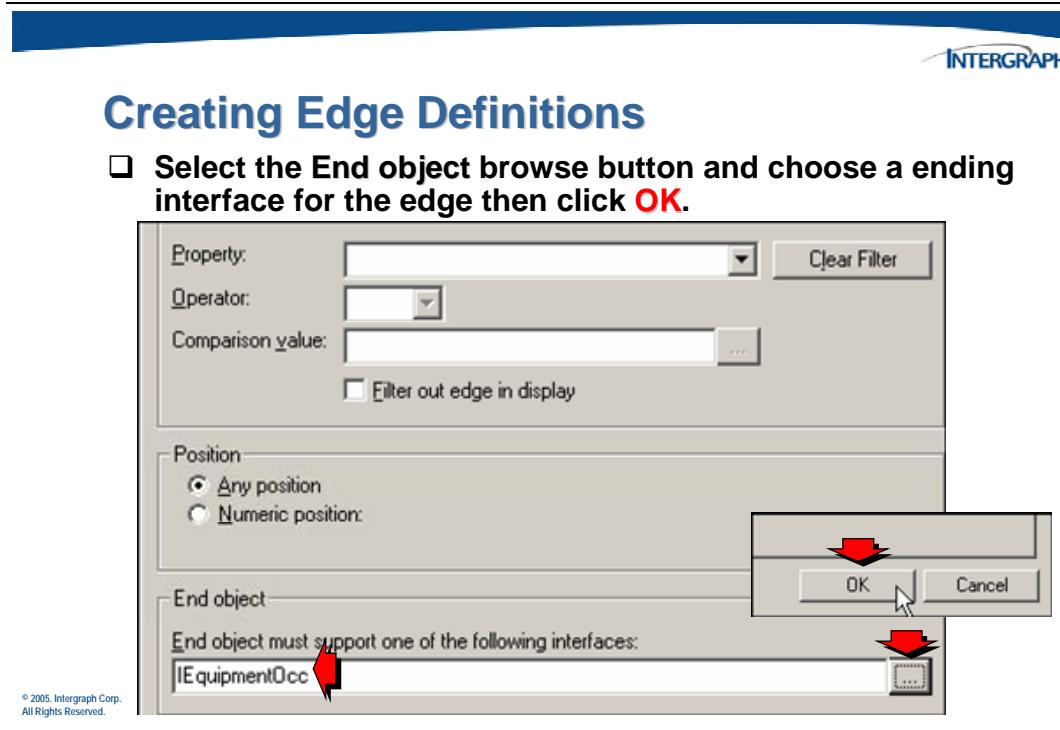
- Select the PBS Children role and then click the *Add to Path* button to add the **PBS Children** role to the edge definition.



Creating Edge Definitions

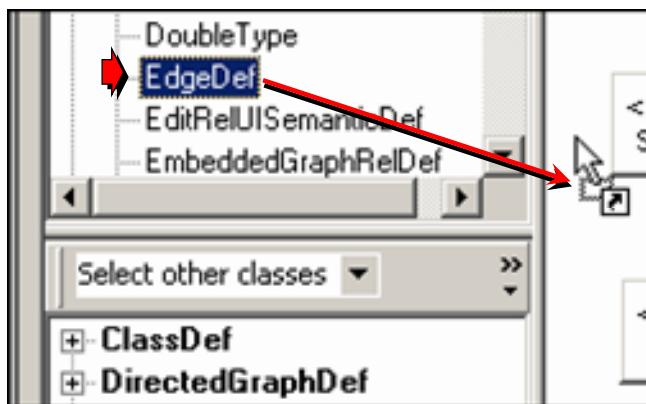
- Select the *Filter* tab to specify edge criteria.





Creating Edge Definitions

- Repeat the same sequence to define another edge by dragging and dropping EdgeDef from the Create window into the UML view window.





Creating Edge Definitions

- Enter a name and display name for the next edge then select the *Path* tab.

Overview | Path | **Filter** | Properties |

Edge properties

*Edge name: NavDesignChangeToInstrument

Property category:

Display name: Instrument

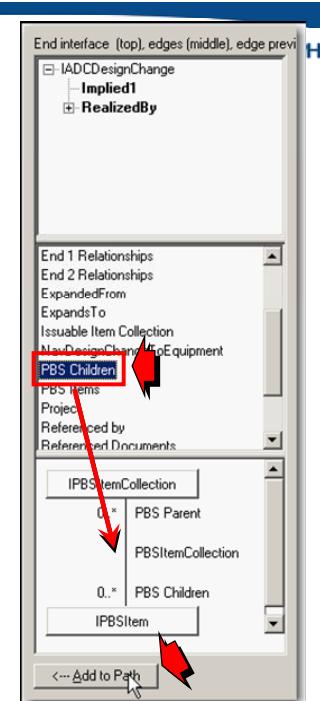
*Starting interface: IADCDesignChange

Definition

© 2005, Intergraph Corp.
All Rights Reserved.

Edge Definitions

- Select the PBS Children role and then click the *Add to Path* button to add the **PBS Children** role to the edge definition.

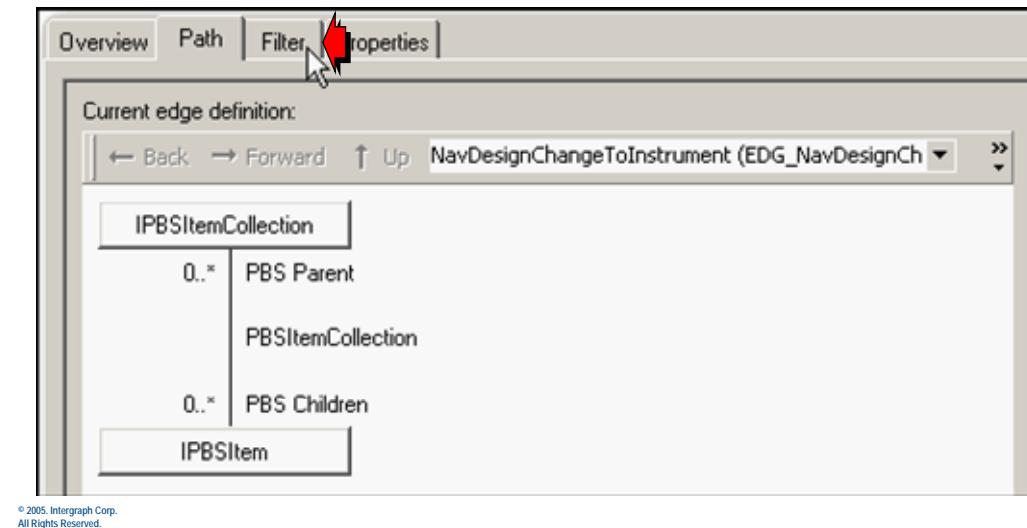


© 2005, Intergraph Corp.
All Rights Reserved.



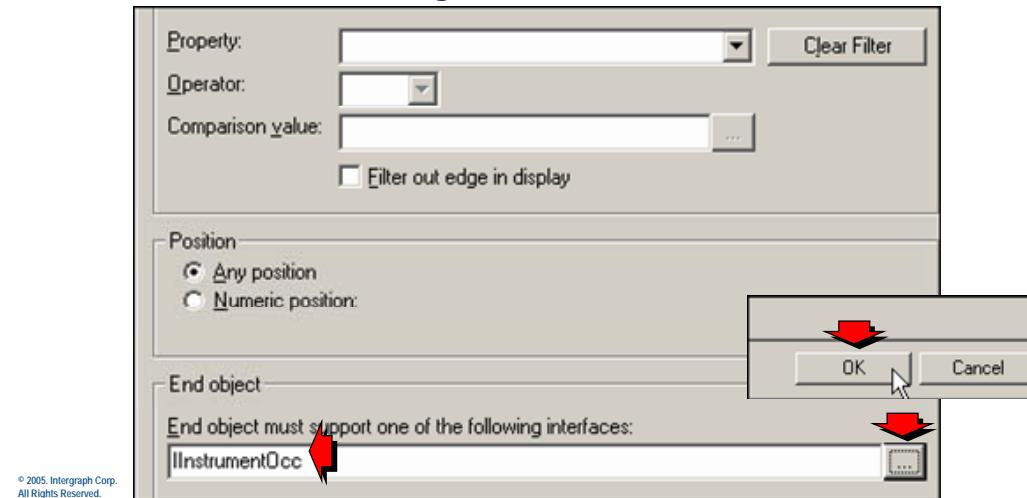
Creating Edge Definitions

- Select the **Filter** tab to specify edge criteria.



Creating Edge Definitions

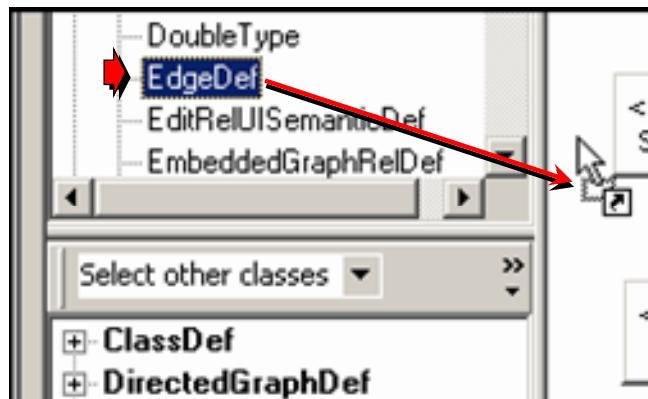
- Select the End object browse button and choose a ending interface for the edge then click **OK**.





Creating Edge Definitions

- Repeat the same sequence to define another edge by dragging and dropping EdgeDef from the Create window into the UML view window.

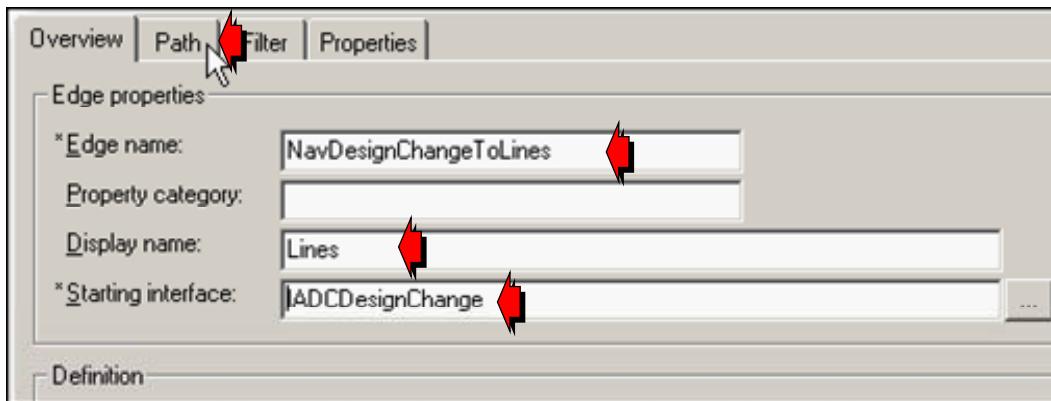


© 2005, Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

- Enter a name and display name for the next edge then select the *Path* tab.



© 2005, Intergraph Corp.
All Rights Reserved.

Edge Definitions

Select the PBS Children role and then click the Add to Path button to add the **PBS Children** role to the edge definition.

The screenshot shows the 'Edge Definitions' dialog box. In the center, there's a diagram of an association between 'IPBSItemCollection' and 'IPBSItem'. The 'IPBSItemCollection' side has a multiplicity of '0..*' and the 'IPBSItem' side has a multiplicity of '0..*'. Below the diagram, the 'PBS Parent' role is highlighted. To the right of the diagram, the 'PBS Children' role is also highlighted. At the bottom right of the dialog, there is a button labeled '... Add to Path' with a red arrow pointing to it. The top right corner of the dialog has a small note: 'End interface (top), edges (middle), edge previous (bottom)'.

Creating Edge Definitions

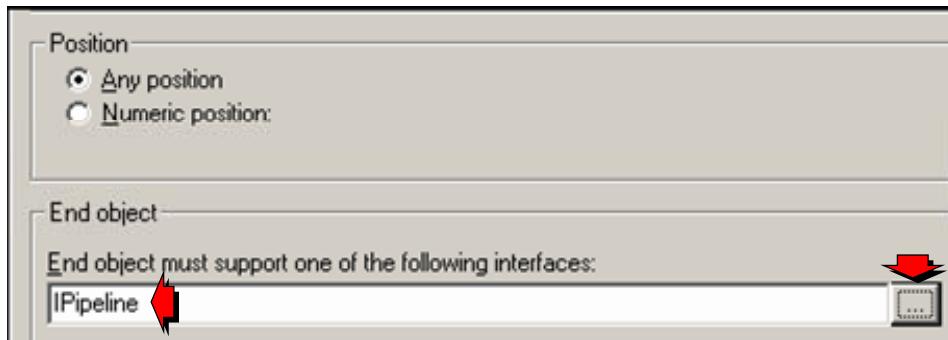
Select the **Filter** tab to specify edge criteria.

The screenshot shows the 'Edge Definitions' dialog box with the 'Filter' tab selected. On the left, under 'Current edge definition', the same association diagram is shown. On the right, the 'Class Definition' list includes 'PBS Parent' and other items like 'Implied1', 'RealizedBy', 'ExpandedFrom', etc. A red arrow points to the 'PBS Parent' item in the list. The top right corner of the dialog has a small note: 'End interface (top), edges (middle), edge previous (bottom)'.



Creating Edge Definitions

- Select the End object browse button and choose a ending interface for the edge then click **OK**.



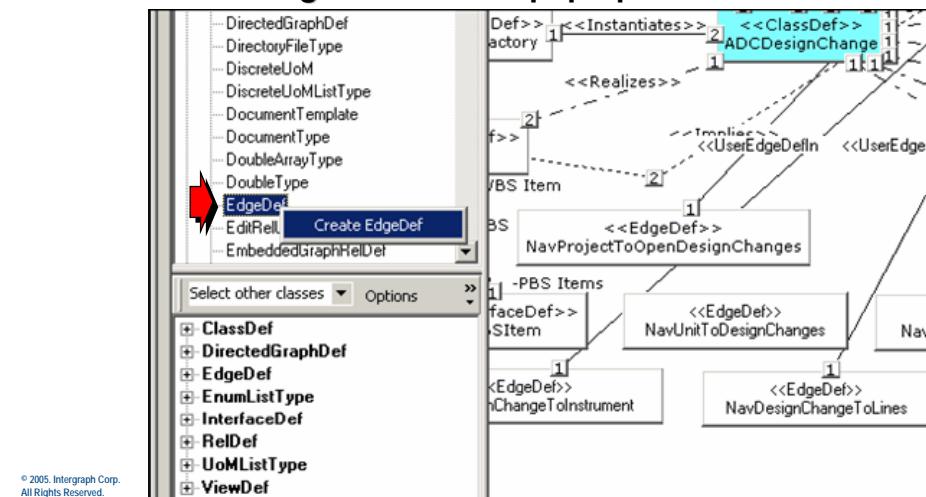
© 2005. Intergraph Corp.
All Rights Reserved.

Add the EdgeDef **DesignChangeToUnit** (needed only for the ViewDef). Once again, the relation to Unit is very generic and we need to know for sure that we are going to have a unit for our ViewDef.



Creating Edge Definitions

- Right-click on **EdgeDef** from the **Create** window and select **Create EdgeDef** from the pop-up menu.

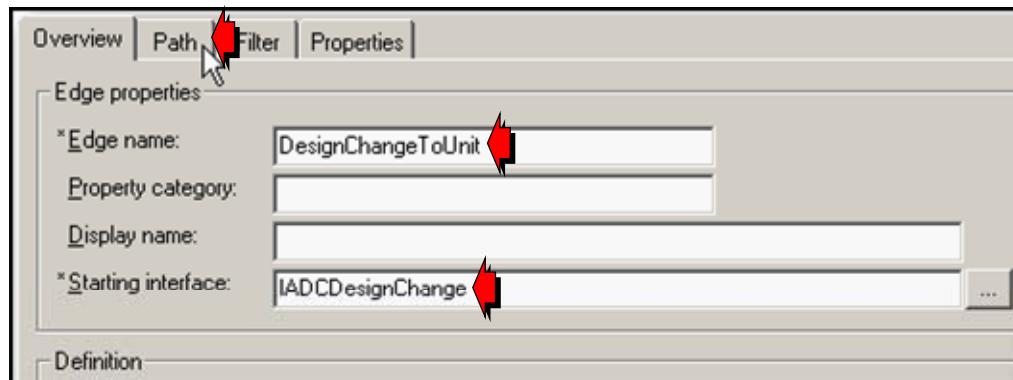


© 2005. Intergraph Corp.
All Rights Reserved.



Creating Edge Definitions

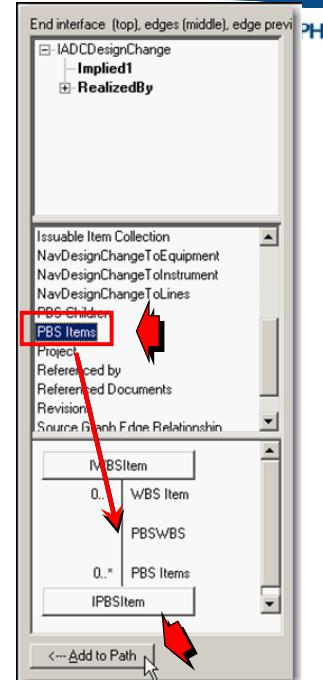
- Enter a name and starting interface for the edge then select the **Path** tab.



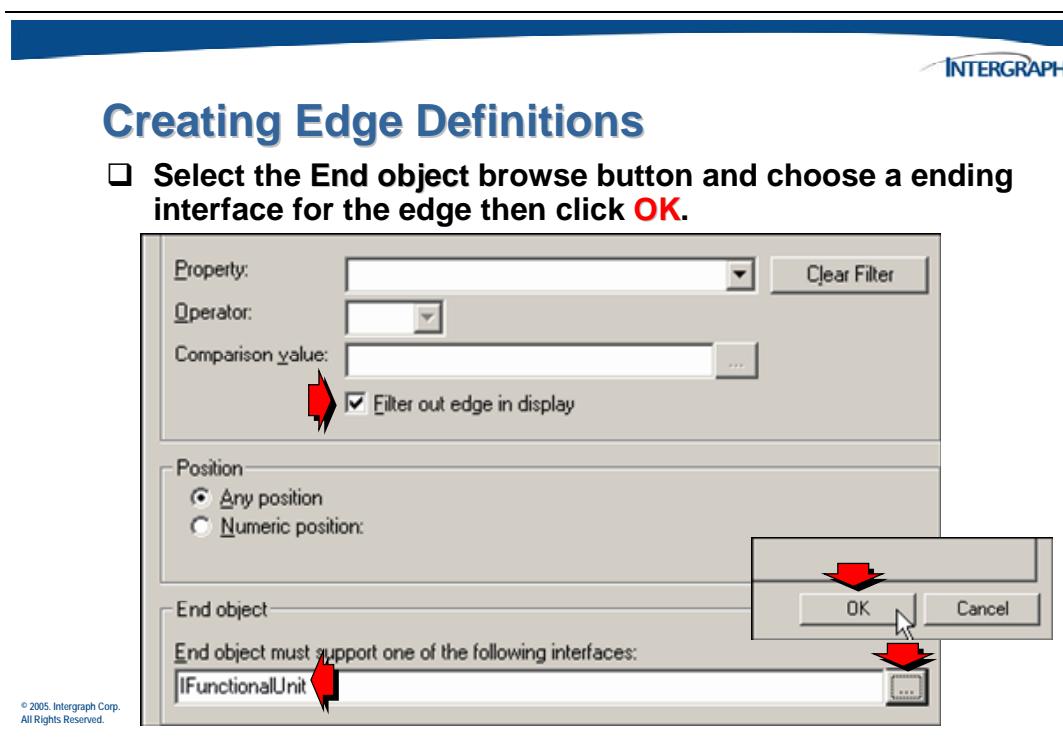
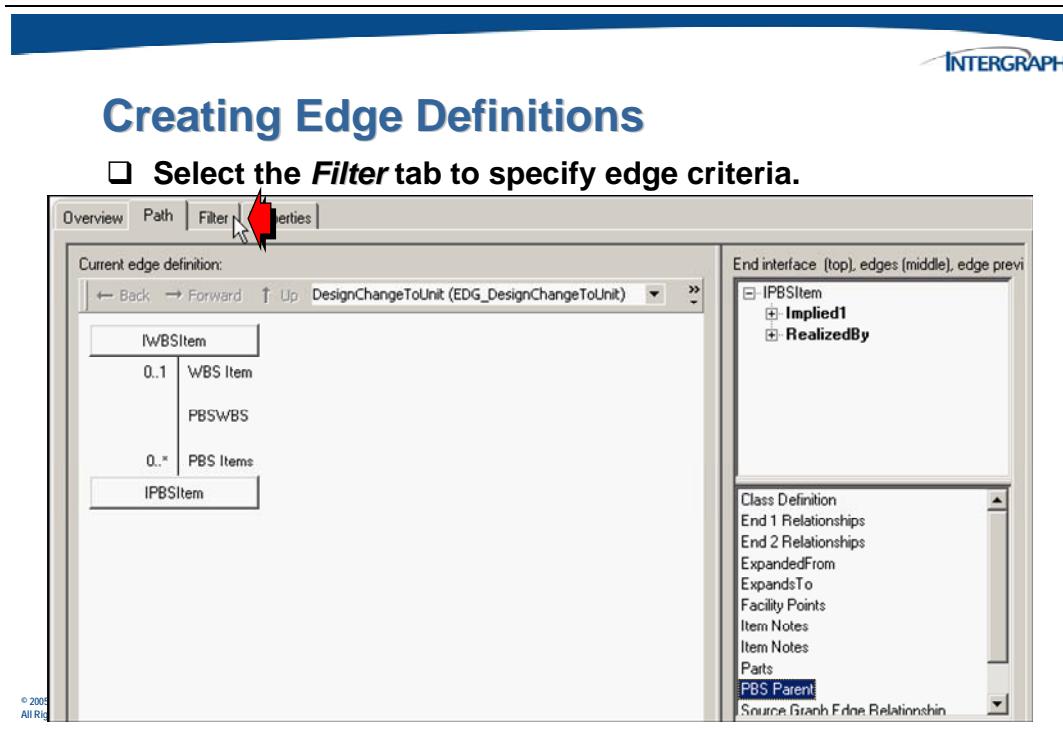
© 2005, Intergraph Corp.
All Rights Reserved.

Edge Definitions

- Select the PBS Items role and then click the **Add to Path** button to add the **PBS Items** role to the edge definition.



© 2005, Intergraph Corp.
All Rights Reserved.



5.3 Graph Definitions

Graph definitions (GraphDef), also called Directed Graph Definitions in the schema, are a connected network of edge definitions starting at a particular interface definition.

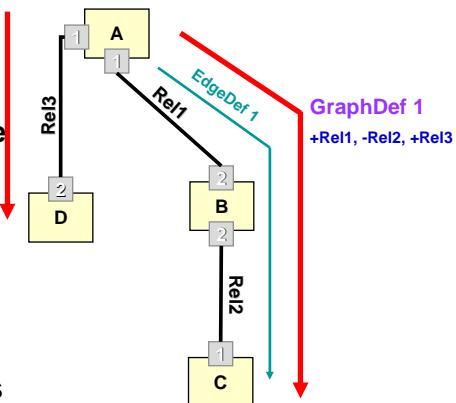
Graph Definitions

Graph definitions (GraphDef) are a set of edge definitions with structure.

Each GraphDef starts at an interface definition and branches out from that interface definition to related interface definitions.

Any edge that is tied to the starting interface definition or to any interface definition implied by that interface definition can be used as part of the GraphDef.

It is not necessary to create EdgeDefs for GraphDefs unless you need to apply criteria.



© 2005, Intergraph Corp.
All Rights Reserved.

Graph definitions include the following:

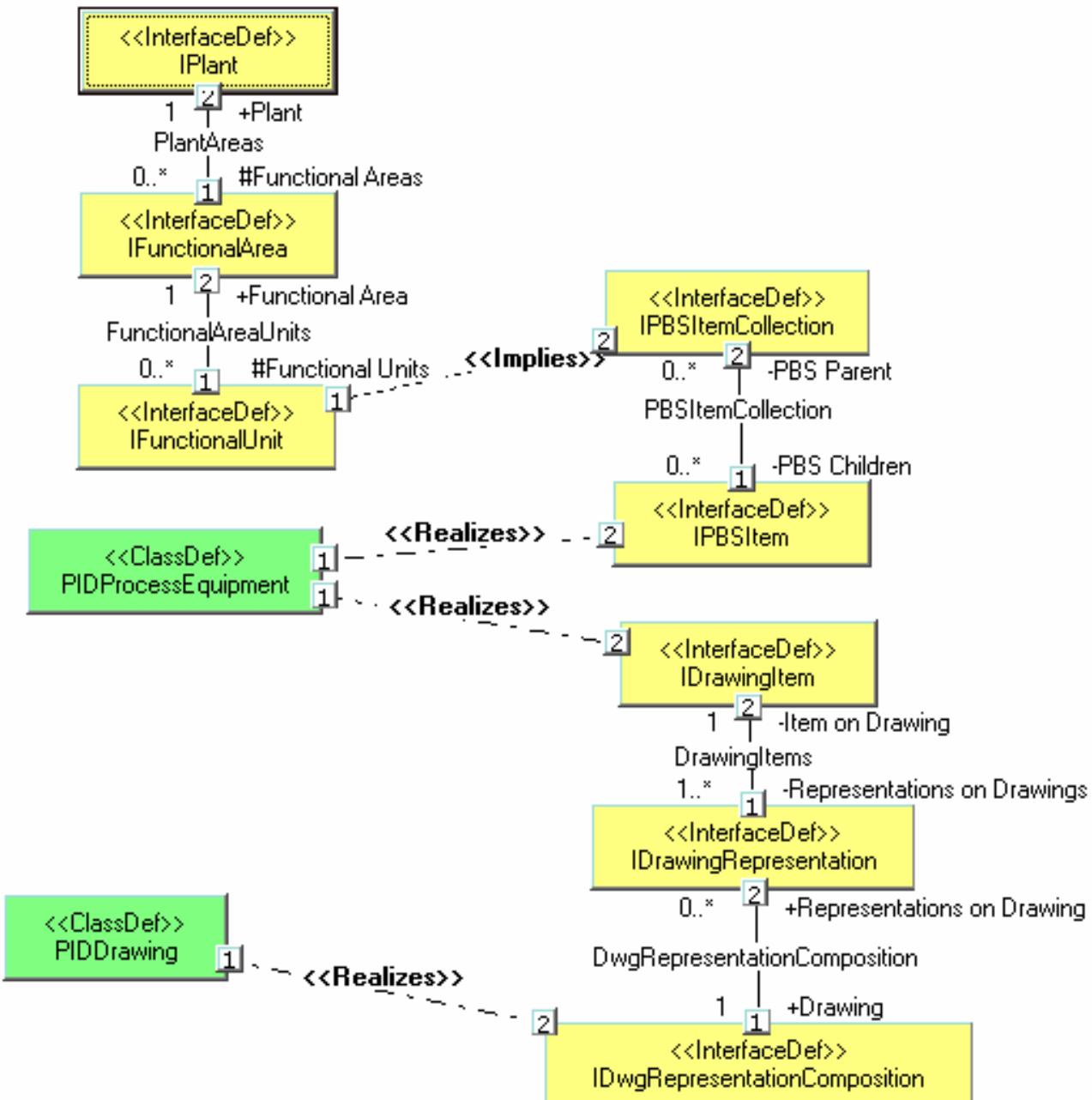
- A relationship to its starting interface definition
- A definition of the directed graph

When you create a GraphDef, explicit EdgeDefs are not required unless you need to apply criteria. One example is the Equipment to Nozzle relationship. You must turn the one-to-many relation into many one-to-one relationships using the position criteria. Another example is the Equipment to Drawing relationship. If you want to put include the P&ID in the GraphDef, you need to navigate that relationship, but you must apply criteria to get only P&IDs instead of all drawings.

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

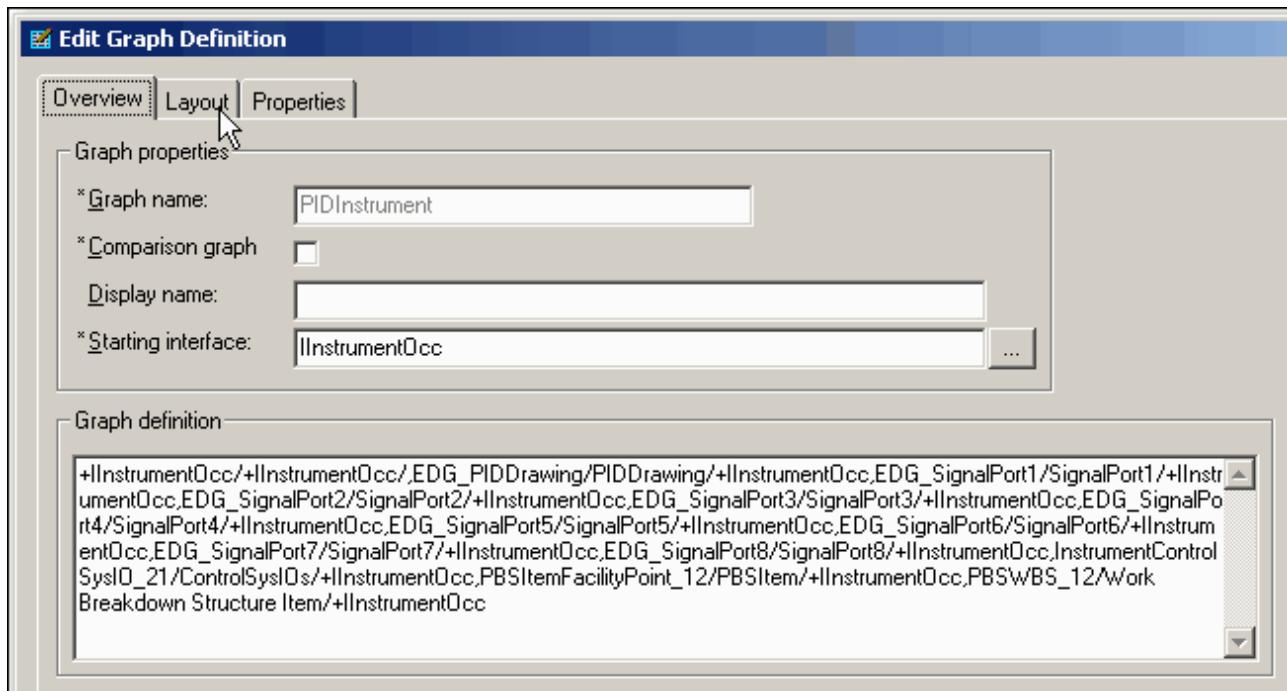
The following UML diagram shows a GraphDef that goes through the plant, area, and unit to a piece of equipment and then to the P&ID on which the equipment appears.

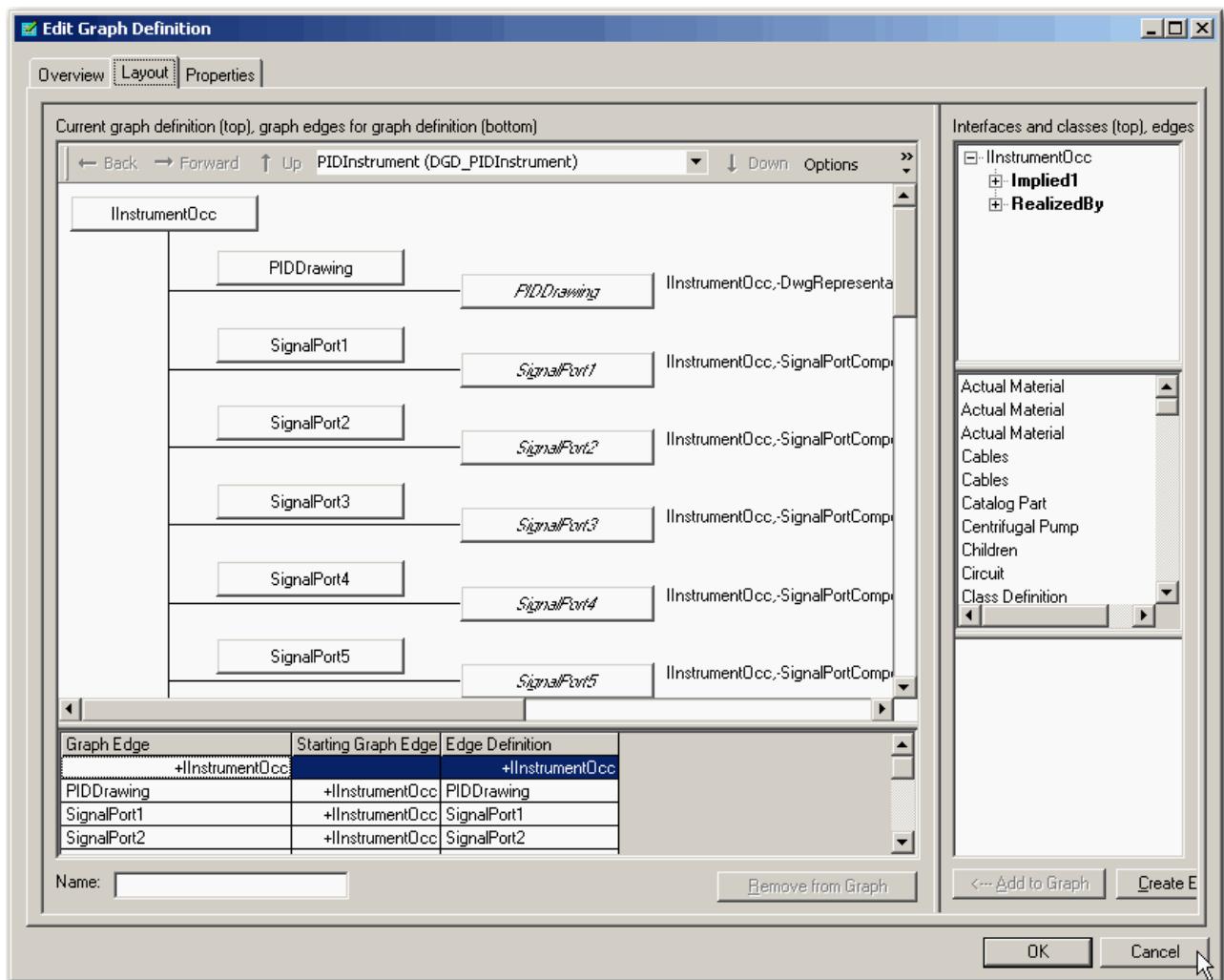


5.3.1 Properties of a Graph Definition

The following options are available when you create or edit a graph definition:

- ❑ **Graph name** – Specifies the name of the graph definition.
- ❑ **Comparison graph** – Specifies whether this graph definition is used for comparison. 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.
- ❑ **Display name** – Specifies the name that you want the user interface to use when displaying the graph definition.
- ❑ **Starting interface** – Defines the starting interface for the graph definition. The starting interface determines which edge definitions are available to include in the graph definition. When you are editing a graph definition, you cannot modify the starting interface definition.

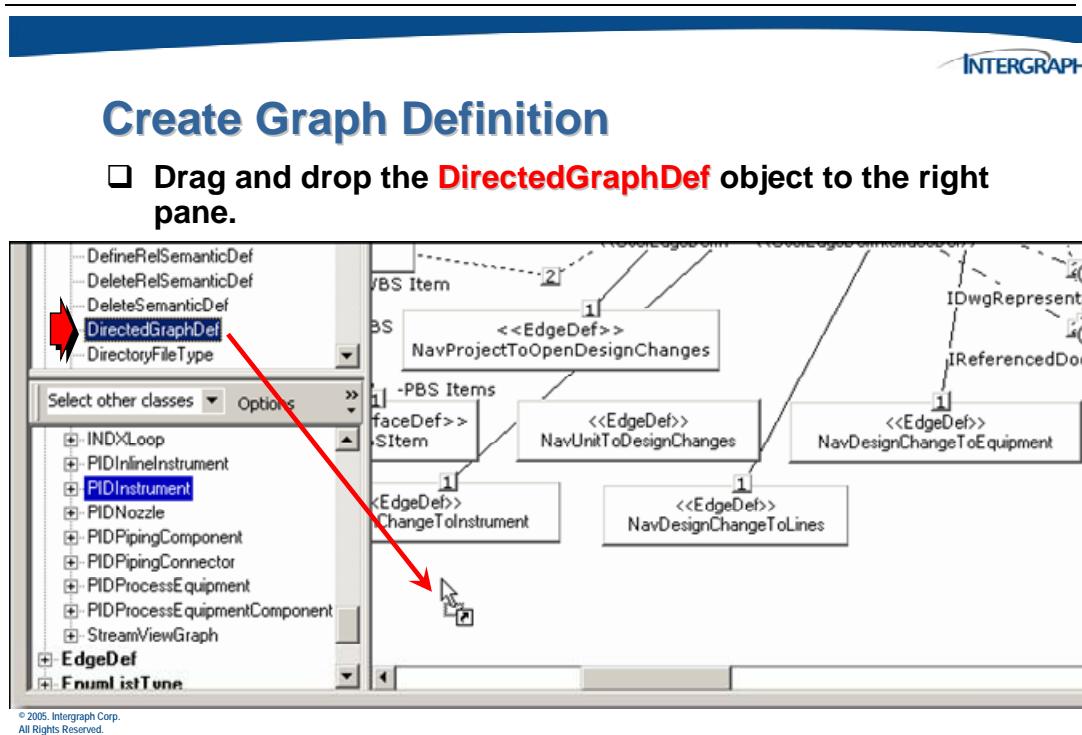




- Layout** – Displays a tree containing all the possible relationship edges that can be expanded from the selected interface definition and all the interfaces that it implies. To add an edge definition to the graph definition, you can click it in the tree view, and then click **Add to Graph**.
- Add to Graph** – Adds the selected edge to the graph definition.
- Current graph definition** – Displays the current graph definition in a tree view.
- Graph definition** – Displays the current graph definition in UML format. Relationships have a + or – in front of them to indicate direction, followed by the relationship name. Commas separate relationships traversed in the GraphDef.

5.4 Interactive Activity – Creating Graph Definitions

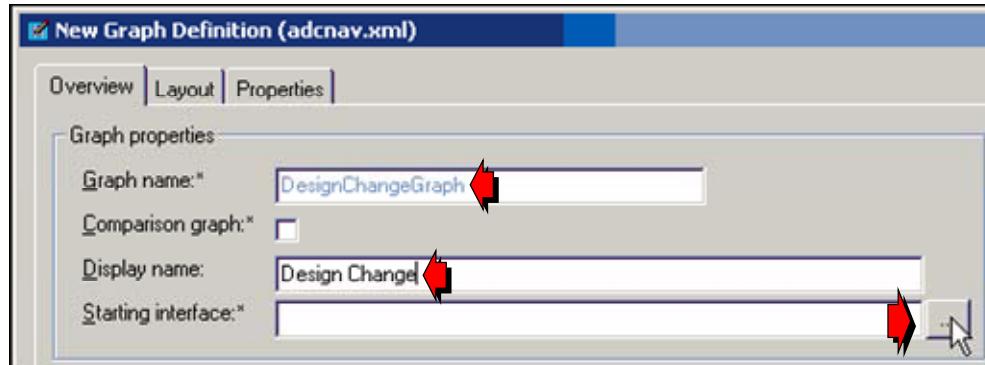
Add the DirectedGraphDef **DesignChangeGraph** to the adc model. This is a Graph of *DesignChange*, *Project*, and *Unit* to be use in a ViewDef.





Create Graph Definition

- Enter a **Graph Name**, a **Display name** and choose a **Starting interface**.

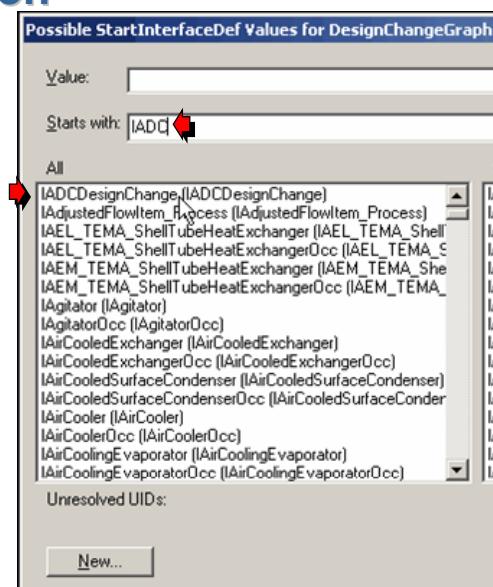


© 2005, Intergraph Corp.
All Rights Reserved.



Create Graph Definition

- Click on **IADCDesignChange** from the **Possible Start InterfaceDef for DesignChangeGraph** dialog.

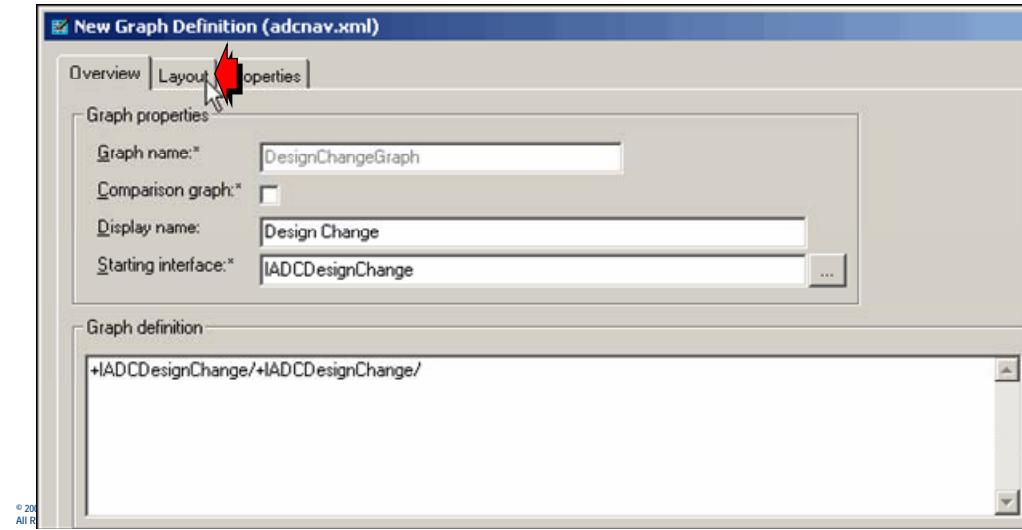


© 2005, Intergraph Corp.
All Rights Reserved.



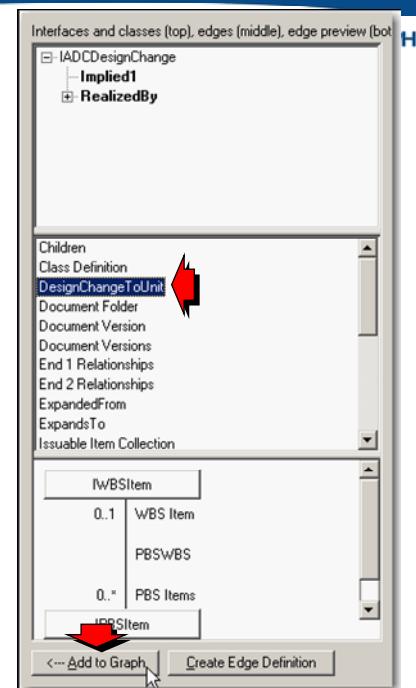
Create Graph Definition

- Select the *Layout* tab to add the edges to this GraphDef.



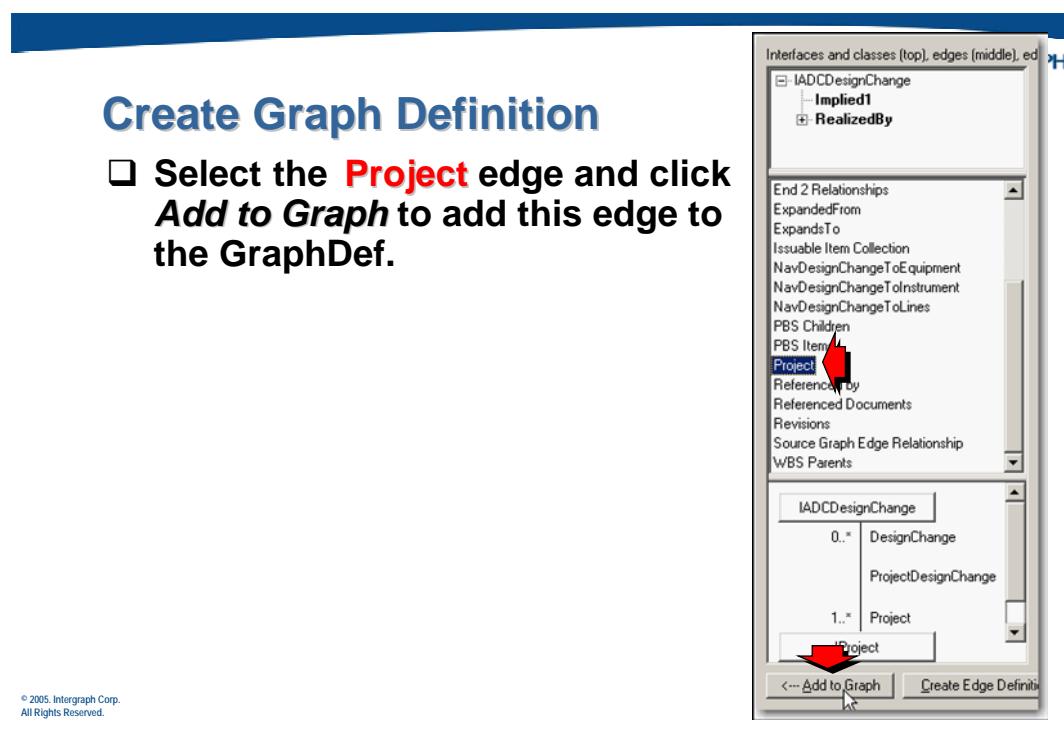
Create Graph Definition

- In the middle pane on the right side of the window, select the **DesignChangeTo Unit** edge and click **Add to Graph** to add this edge to the GraphDef.



Create Graph Definition

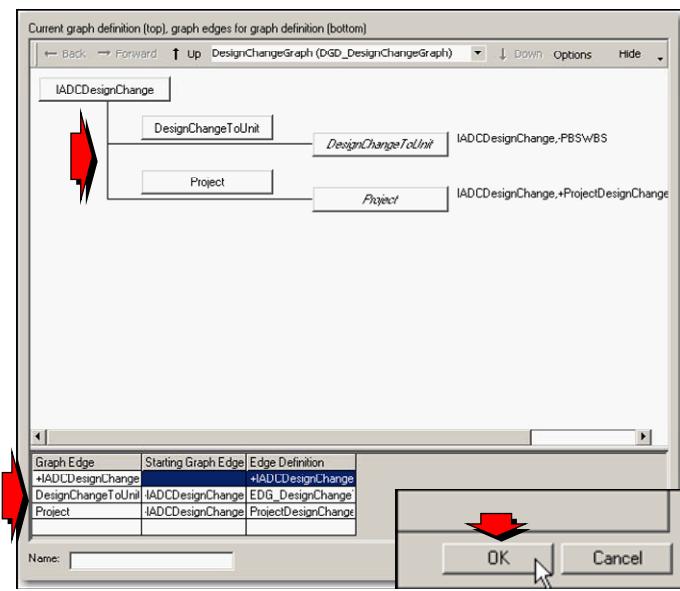
- ❑ Select the **Project** edge and click **Add to Graph** to add this edge to the GraphDef.



© 2005. Intergraph Corp.
All Rights Reserved.

Create Graph Definition

- ❑ Click **OK** to create the new GraphDef.



© 2005. Intergraph Corp.
All Rights Reserved.

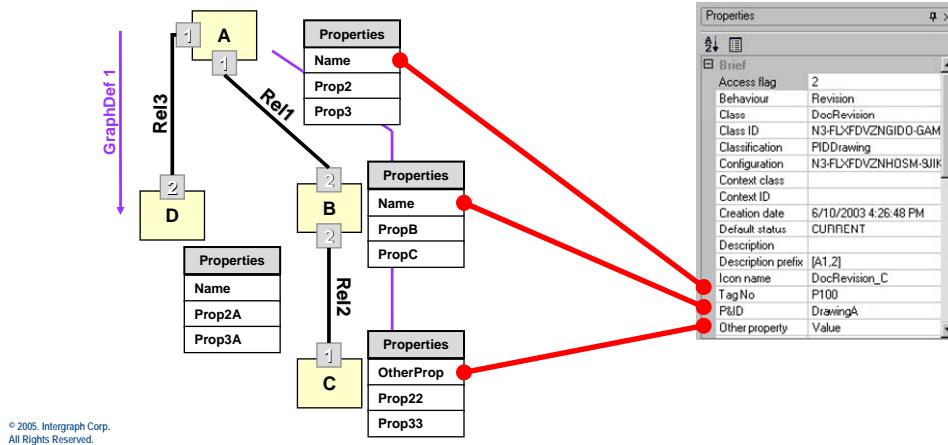
5.5 View Definitions

While schema-based depictions of the data are useful to users who are familiar with the underlying schema, other users need more user-oriented views of the data for it to be meaningful.



View Definitions

View definitions (ViewDef) are properties extracted from possible properties that a graph definition exposes. A view definition is used to provide a different view of data from that provided by the underlying schema.





View Definitions

ViewDefs 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.

© 2005. Intergraph Corp.
All Rights Reserved.

SmartPlant view definitions consist of the following:

- A relationship to its starting interface definition
- An identification of the directed graph definition that applies
- A definition of the projection of property definitions from the directed graph definition

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. In actuality, this interface definition is always the same interface definition as that for its directed graph definition. The directed graph definition for the view definition defines the set of edge definitions that will be traversed when the view corresponding to the view definition is created.

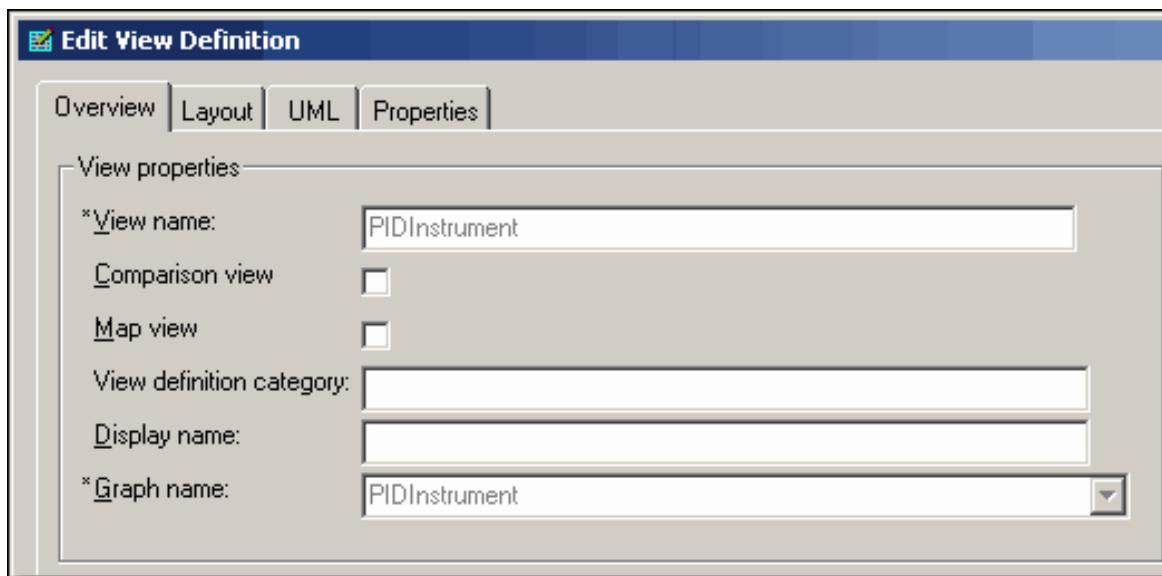
In SmartPlant Foundation, view definitions are used to define what users see in the **Properties** window when they select an object in the client. When you create ad-hoc reports in the SmartPlant Foundation client, you can also select the view definition that you want to use as the basis for the report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

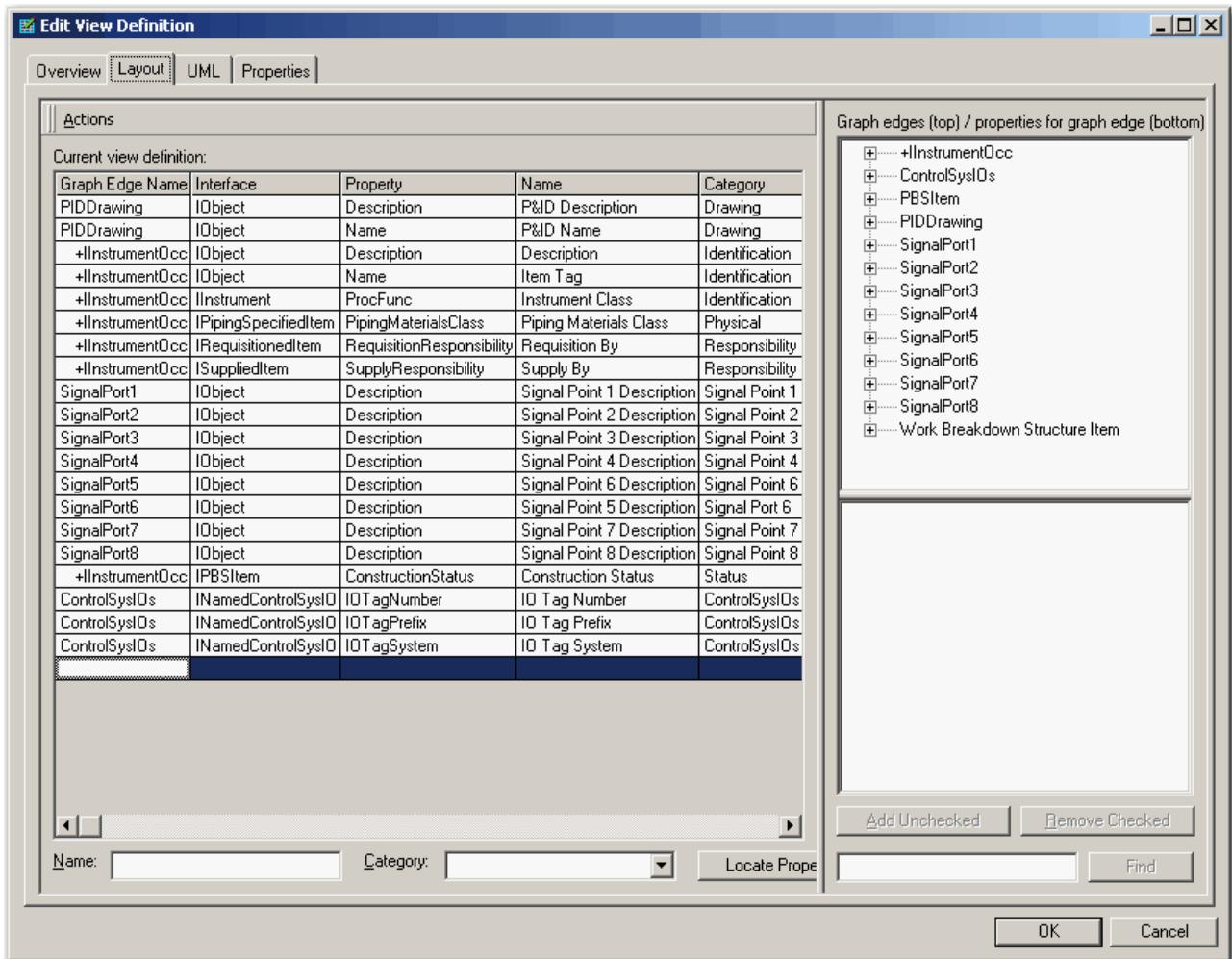
5.5.1 Properties of a View Definition

The following options are available when you create or edit a view definition:

- ❑ **View name** – Specifies the name of the view definition.
- ❑ **Comparison view** - Specifies whether this view definition is used for comparison. A comparison graph 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.
- ❑ **Map view** – Specifies whether or not the view definition is to be used for defining mapping between applications and the SmartPlant schema.
- ❑ **View definition category** – Currently not used
- ❑ **Display name** - Specifies the name that you want the user interface to use when displaying the view definition.
- ❑ **Graph name** – Specifies the name of the graph definition that this view definition contains.

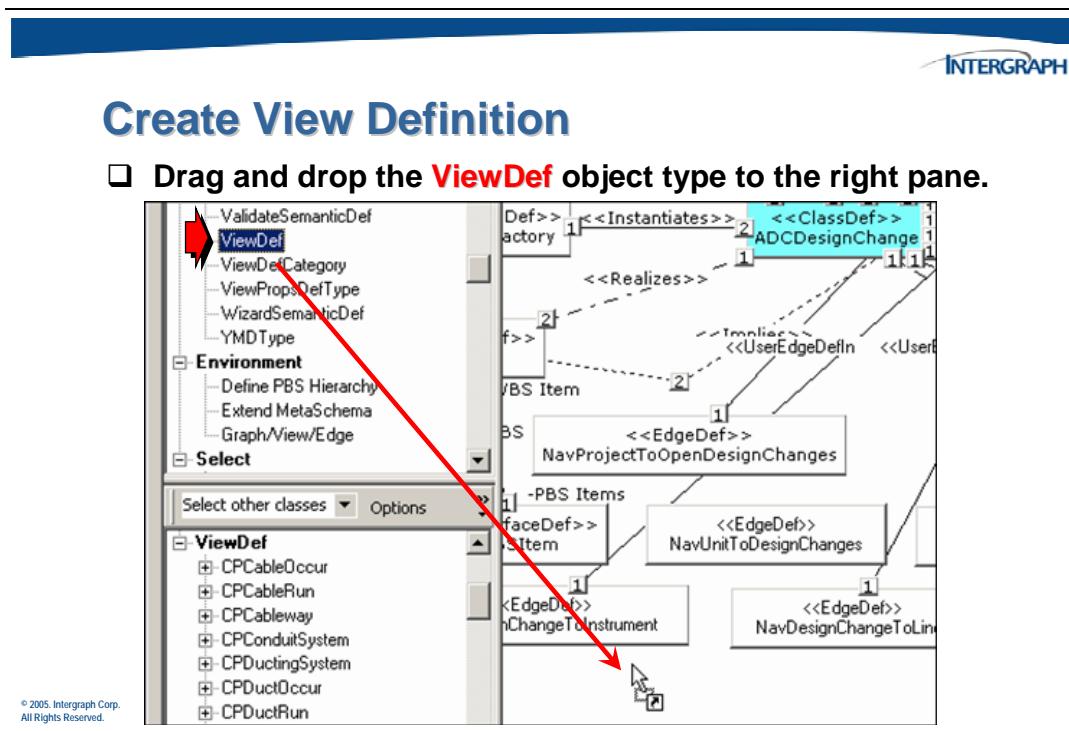


- ❑ **Current view definition** - Displays the current view definition in text format. You can edit the view definition manually in this box.
- ❑ **Name** – Specifies the name of the selected property when it appears in the view definition. After you select the property that you want to add you can type a new display name here.



5.6 Interactive Activity – Creating View Definitions

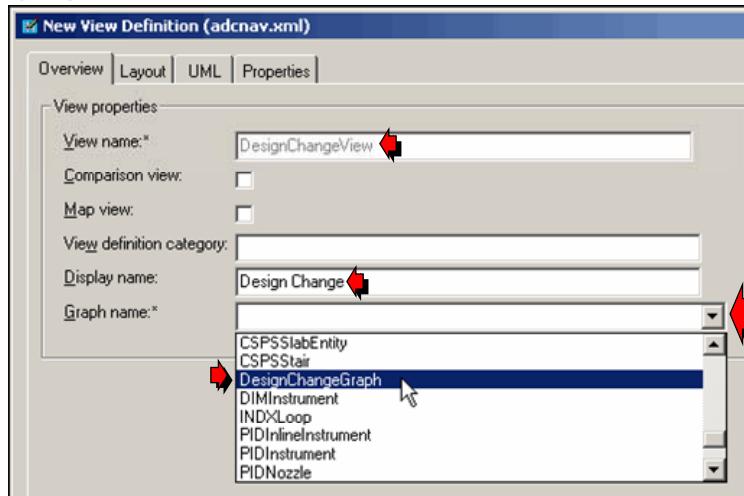
Add the ViewDef **DesignChangeView** to the adc model. This is the view that can be displayed in the Desktop Client property grid when a user clicks on a Design Change object instance.





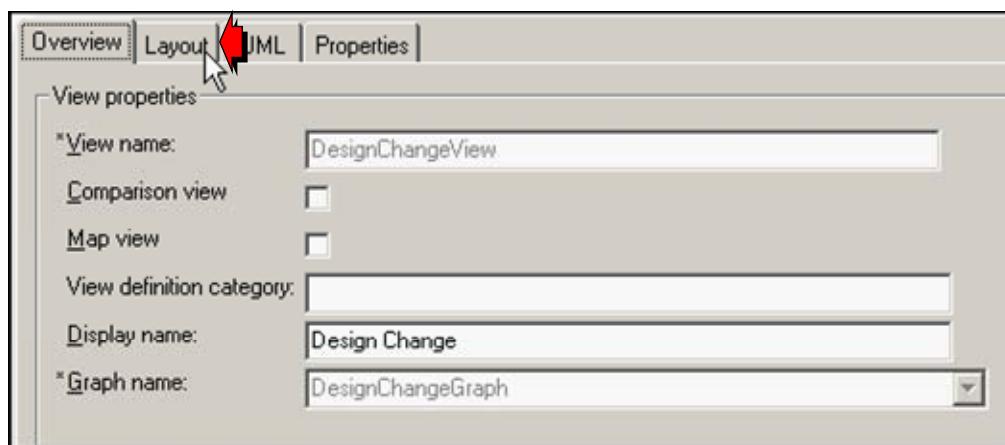
Create View Definition

- Enter a **View Name**, a **Display name** and choose a **Graph Name**.



Create View Definition

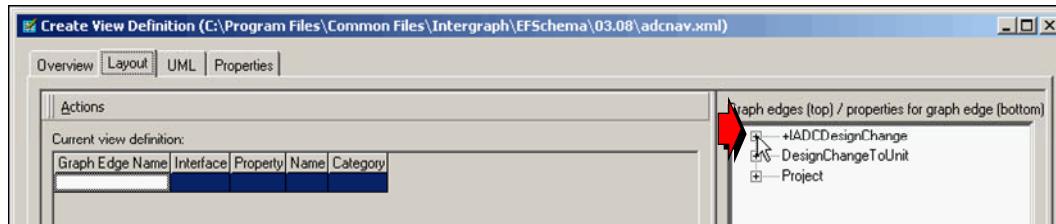
- Select the **Layout** tab to choose the properties for this ViewDef.





Create View Definition

- Click to expand **IADCDesignChange** in the *Graph edges* field.

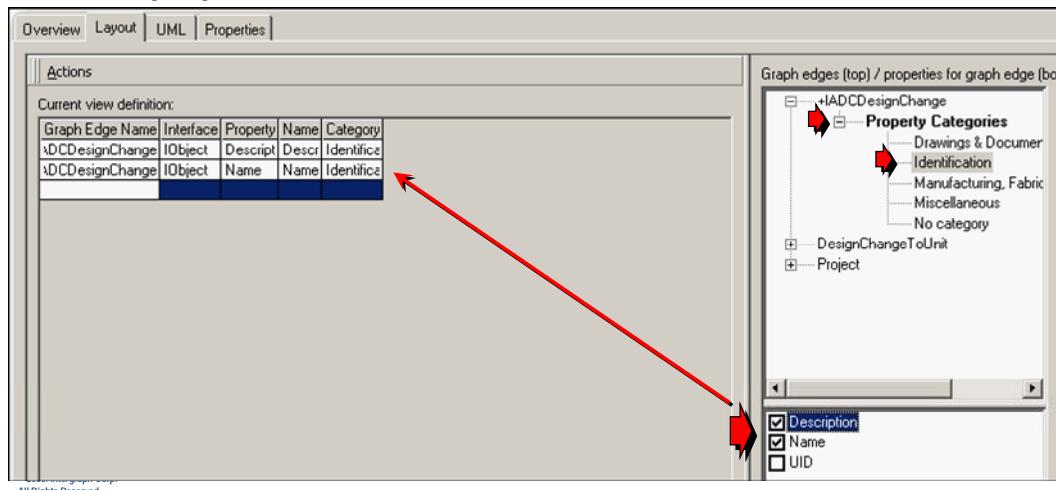


© 2005, Intergraph Corp.
All Rights Reserved.



Create View Definition

- Click the **Identification Property Category** and select the properties to be included in the view.



© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Select the **Name** property in order to rename it for this view.

Graph Edge Name	Interface	Property	Name	Category
\DCDesignChange	IObject	Name	Name	Identification
\DCDesignChange	IObject	Description	Description	Identification

Name: Category:

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Enter a new display name and either press the *Tab* key or click the Locate Property button.

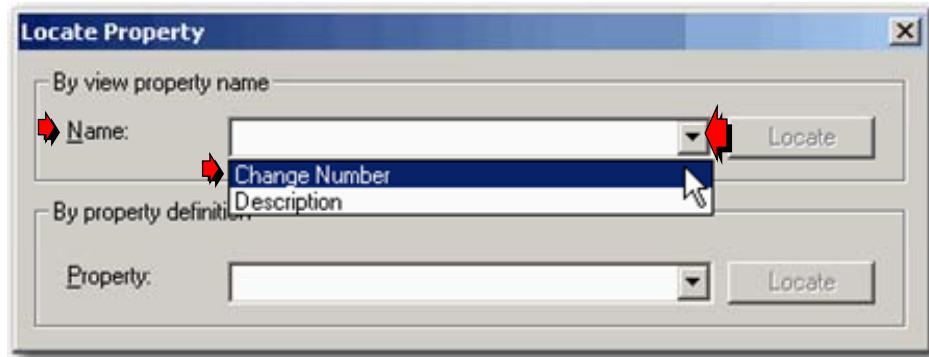
Graph Edge Name	Interface	Property	Name	Category
\DCDesignChange	IObject	Name	Name	Identification
\DCDesignChange	IObject	Description	Description	Identification

Name: Change Number:



Create View Definition

- Use the **By view property name** list button to verify the new view property name.



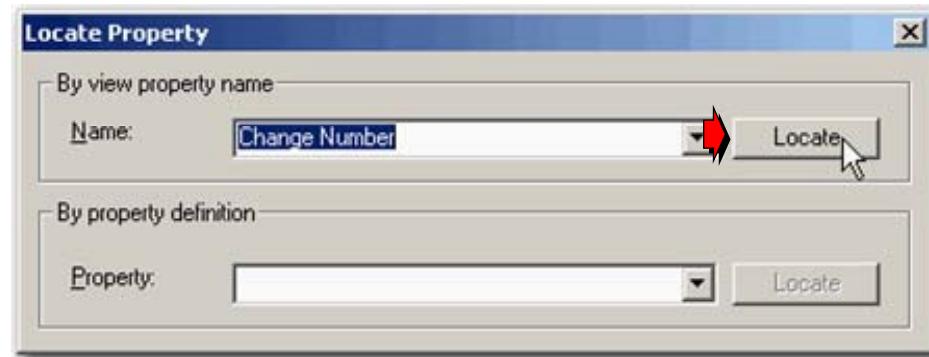
© 2005, Intergraph Corp.
All Rights Reserved.

This is an alternative way to change the view property name.



Create View Definition

- Click the Locate button.



© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

The property name in the view will show the modification.

Graph Edge Name	Interface	Property	Name	Category
xDCTDesignChange	IObject	Name	Change Number	Identification
xDCTDesignChange	IObject	Description	Description	Identification

Name: Change Number Category: Identification Locate Prop...

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

- Click the Project Identification Property Category and select the properties to be included in the view.

Graph Edge Name	Interface	Property	Name	Category
Project	IObject	Name	Project Number	Identification
xDCTDesignChange	IObject	Name	Change Number	Identification
xDCTDesignChange	IObject	Description	Description	Identification

Name: Project Number Category: Identification Locate Prop...

Graph edges (top) / properties for graph edge (bottom)

Property Categories

- + → IADCDDesignChange
 - Property Categories
 - Drawings & Documents
 - Identification
 - Manufacturing, Fabrication
 - Miscellaneous
 - No category
- DesignChangeToUnit
- Project
 - Property Categories
 - Identification
 - Manufacturing, Fabrication
 - Miscellaneous

Add Unchecked Remove Checked Find

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Click the **DesignChangeToUnit Identification Property Category** and select the properties to be included.

The screenshot shows the 'Create View Definition' dialog. On the left, a table titled 'Current view definition' lists several graph edges with their interfaces, properties, names, and categories. One row, 'DesignChangeToUnit', has its 'Name' column set to 'Unit' and its 'Category' column set to 'Identification'. A red arrow points from this row to the 'Identification' node in the 'Property Categories' tree view on the right. The tree view also shows other categories like 'Drawings & Documents', 'Manufacturing, Fabricate', and 'Miscellaneous'. Below the tree view are checkboxes for various properties, with 'Name' and 'UID' checked. Buttons for 'Add Unchecked' and 'Remove Checked' are at the bottom.

Create View Definition

Click the **IADCDesignChange No category Property Category** and select the additional properties to be included.

The screenshot shows the 'Create View Definition' dialog. On the left, a table titled 'Current view definition' lists several graph edges with their interfaces, properties, names, and categories. One row, 'IADCDesignChange', has its 'Name' column set to 'Change UOM' and its 'Category' column set to 'Identification'. A red arrow points from this row to the 'No category' node in the 'Property Categories' tree view on the right. The tree view also shows other categories like 'Drawings & Documents', 'Identification', 'Manufacturing, Fabricate', and 'Miscellaneous'. Below the tree view are checkboxes for various properties, all of which are checked: 'ADCDApprovalSignature', 'ADCDDetail', 'ADCDImpact', 'ADCMear', 'ADCPProposalSignature', and 'GenDesignedItemStatus'. Buttons for 'Add Unchecked' and 'Remove Checked' are at the bottom.

The order of the properties in the ViewDef can be changed by cutting and inserting rows in the *Current view definition* field.

Create View Definition

Highlight the row(s) to be moved, then right click and select Cut from the pop up menu.

Graph Edge Name	Interface	Property	Name	Category
Project	IObject	Name	Project Number	Identification
HADCDesignChar			Range Number	Identification
HADCDesignChar			Description	Identification
HADCDesignChar			Approval Signature	
HADCDesignChar			Range Detail	
HADCDesignChar			Range Impact	
HADCDesignChar			Range UOM	
HADCDesignChange	IADCDesignChar	GenDesign	Design Status	
DesignChangeToUn	IObject	Name	Unit	Identification

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Highlight the row where the new rows are to be inserted above, then right click and select Paste Insert from the pop up menu.

Graph Edge Name	Interface	Property	Name	Category
Project	IObject	Name	Project Number	Identification
HADCDesignChar			Delete	
HADCDesignChar			Cut	
HADCDesignChar			Paste Insert	
HADCDesignChar			Create Spreadsheet	
HADCDesignChar			Locate Property in Tree	
HADCDesignChar			Proposal Signature	
DesignChangeToUn	IObject	Name	Design Status	
			Unit	Identification

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Results of moving and pasting inserted rows

Actions					
Current view definition:					
Graph Edge Name	Interface	Property	Name	Category	
+IADCDesignChange	IObject	Name	Change Number	Identification	
Project	IObject	Name	Project Number	Identification	
DesignChangeToUnit	IObject	Name	Unit	Identification	
+IADCDesignChange	IObject	Description	Description	Identification	
+IADCDesignChange	IADCDesignChange	ADCDetail	Change Detail		
+IADCDesignChange	IADCDesignChange	ADCImpact	Change Impact		
+IADCDesignChange	IADCDesignChange	ADCMeas	Change UOM		
+IADCDesignChange	IADCDesignChange	ADCApprovalSignature	Approval Signature		
+IADCDesignChange	IADCDesignChange	ADCPProposalSignature	Proposal Signature		
+IADCDesignChange	IGenDesignedStatus	GenDesignedItemStatus	Design Status		

© 2005, Intergraph Corp.
All Rights Reserved.

Create View Definition

Click **OK** to create the new **ViewDef**.

UML Properties

DesignChangeView (VD_DesignChangeView)

IADCDesignChange	Approval Signature = IADCDesignChange.ADCApprovalSignature Change Detail = IADCDesignChange.ADCDetail Change Impact = IADCDesignChange.ADCImpact Change Number = IObject.Name Change UOM = IADCDesignChange.ADCMeas Description = IObject.Description Design Status = IGenDesignedStatus.GenDesignedItemStatus Proposal Signature = IADCDesignChange.ADCProposalSignature
DesignChangeToUnit	Unit = IObject.Name
Project	Project Number = IObject.Name

OK Cancel

© 2005, Intergraph Corp.
All Rights Reserved.

5.7 Class View Maps

In the schema, **class view maps** associate class definitions with view definitions. After the SmartPlant schema is loaded into the SmartPlant Foundation system administration database, class view maps can be used to define user access to different views in the SmartPlant Foundation client in SmartPlant Foundation SmartPlant Administration. The class view map specifies the default view definitions the software should use for a set of class definitions.



Class View Maps

A **Class View Map** (**ClassViewMap**) is a collection of **ClassDefs** and **ViewDefs** in the schema that has related user groups in SmartPlant Foundation.

ClassViewMaps define the default **ViewDefs** the software should use for a set of **ClassDefs**.

Relationships with SPF user groups specify which view the software uses in the *Properties* window or in an ad-hoc report when a particular user selects a **ClassDef** in the SPF client.

Associations between SPF user groups and ClassViewMaps are defined in SPF SmartPlant Administration.

© 2005, Intergraph Corp.
All Rights Reserved.

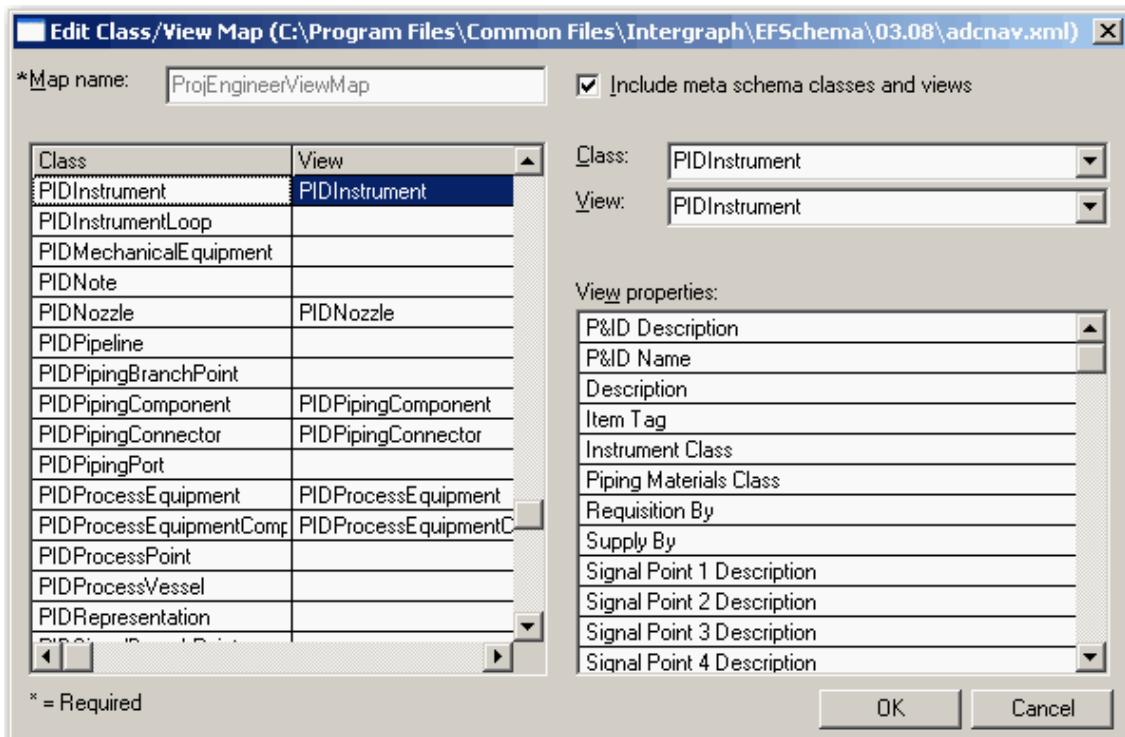
Defining the class view map is especially important for shared class definitions because it allows you to use the same view definition for multiple class definitions, including those objects that are shared across tools. By using the same view definition for multiple class definitions, users see the same presentation of information whether they are looking at the originally published class or a class from another tool for the shared object.

For example, if SmartPlant P&ID publishes an object of the PIDProcessEquipment class and Zyqad publishes an object of the EQDCentrifugalPump class, the shared object will have a different class in SmartPlant Foundation. To allow users to see the same view of the shared object regardless of which tool published it, the same view definition could be associated with both the class definitions.

5.7.1 Properties of a Class View Map

The following options are available when you create or edit a graph definition:

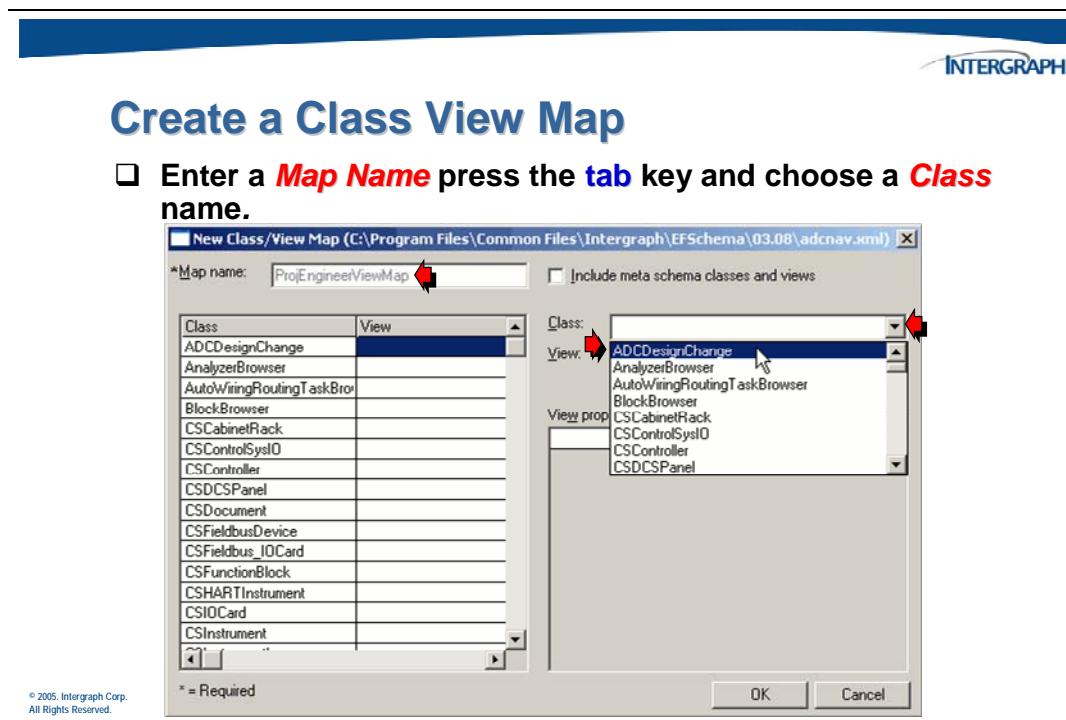
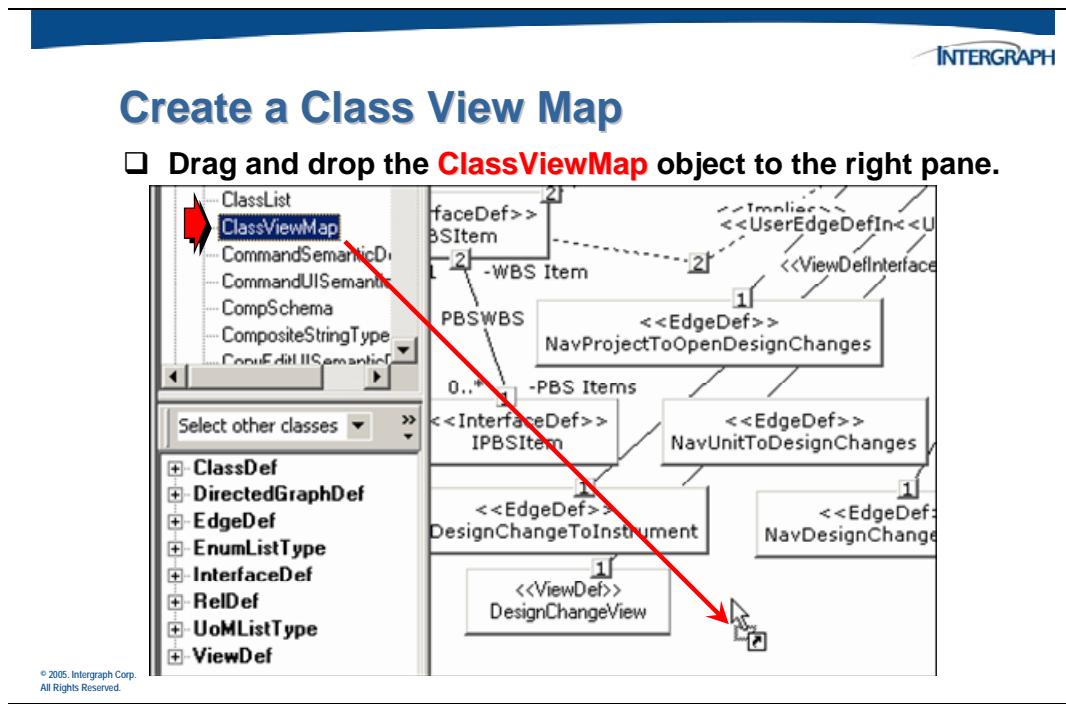
- Map name** – Specifies the name of the class view map.
- Class/View Table** - Displays a list of all class definitions and the view definitions associated with them. To select a class definition to map, you can click it in this table or in the **Class** box. You can also change the view mapping for a class definition by selecting the class definition in this table and making changes in the **Class** and **View** boxes.



- Include meta schema classes and views** – Indicates whether you want to include classes and views from the meta schema in your class view map. Including meta schema classes and views are only used inside the Schema Editor. They are useful when you want to use class view maps to define the set of comparison view definitions to use for the set of corresponding class definitions or to define the view that is displayed in the Table view in the Schema Editor for each class definition.
- Class** – Specifies the class definition to which you want to map a view definition. You can select a class definition in this list or click the class definition in the table on the left side of the dialog box to automatically fill in the **Class** box.
- View** – Specifies the view definition that you want to associate with the class definition selected in the **Class** box.
- View properties** – Displays a list of all properties associated with the view selected in the **View** list.

5.8 Interactive Activity – Creating Class View Maps

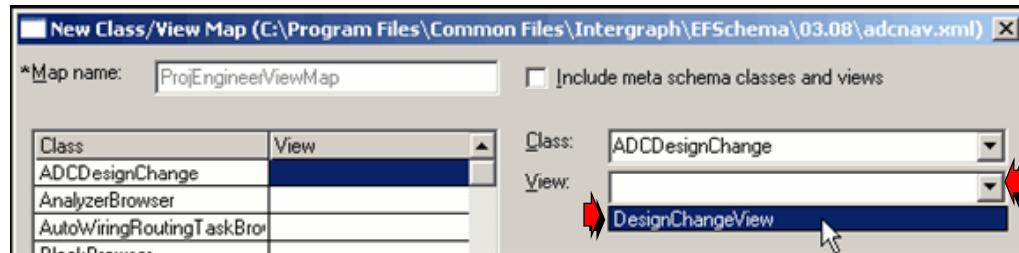
Add the ClassViewMap **ProjEngineerViewMap** to the adc model. A ClassViewMap is used to connect that view to a role.





Create a Class View Map

- Choose a **View** name to complete the mapping.

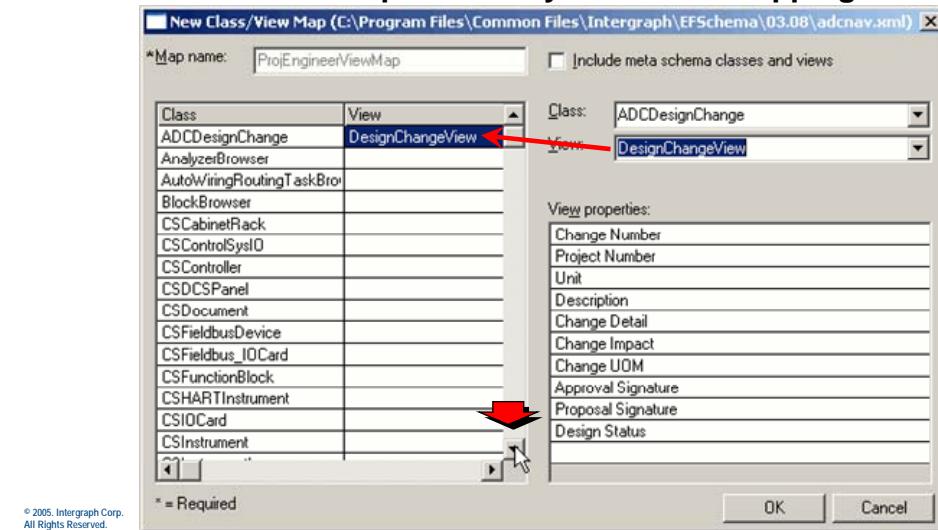


© 2005, Intergraph Corp.
All Rights Reserved.



Create a Class View Map

- Scroll down to perform any additional mapping.

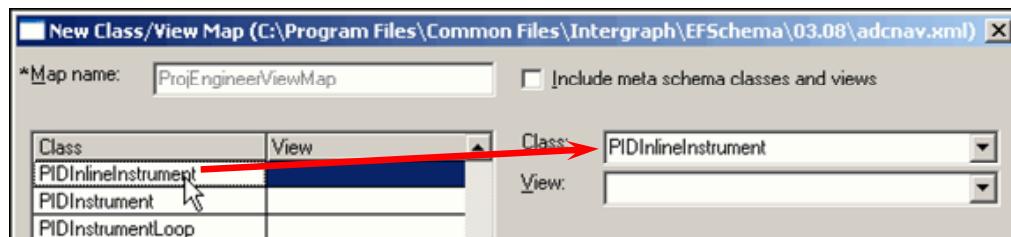


© 2005, Intergraph Corp.
All Rights Reserved.



Create a Class View Map

- Click on the class field to begin to define another Class -> View mapping.

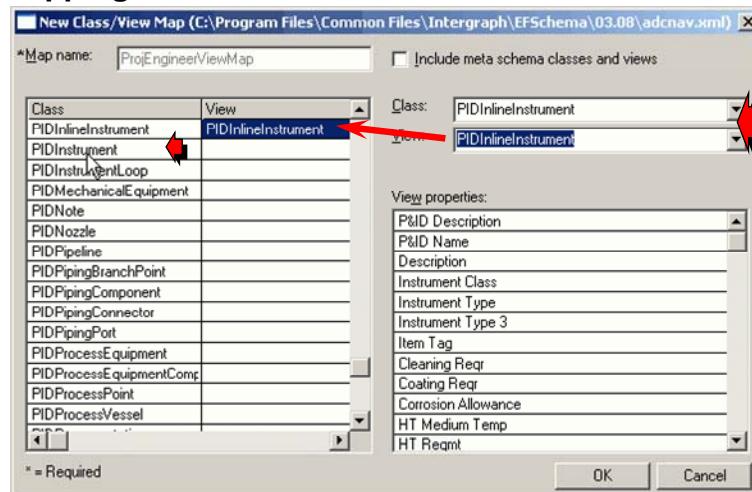


© 2005, Intergraph Corp.
All Rights Reserved.

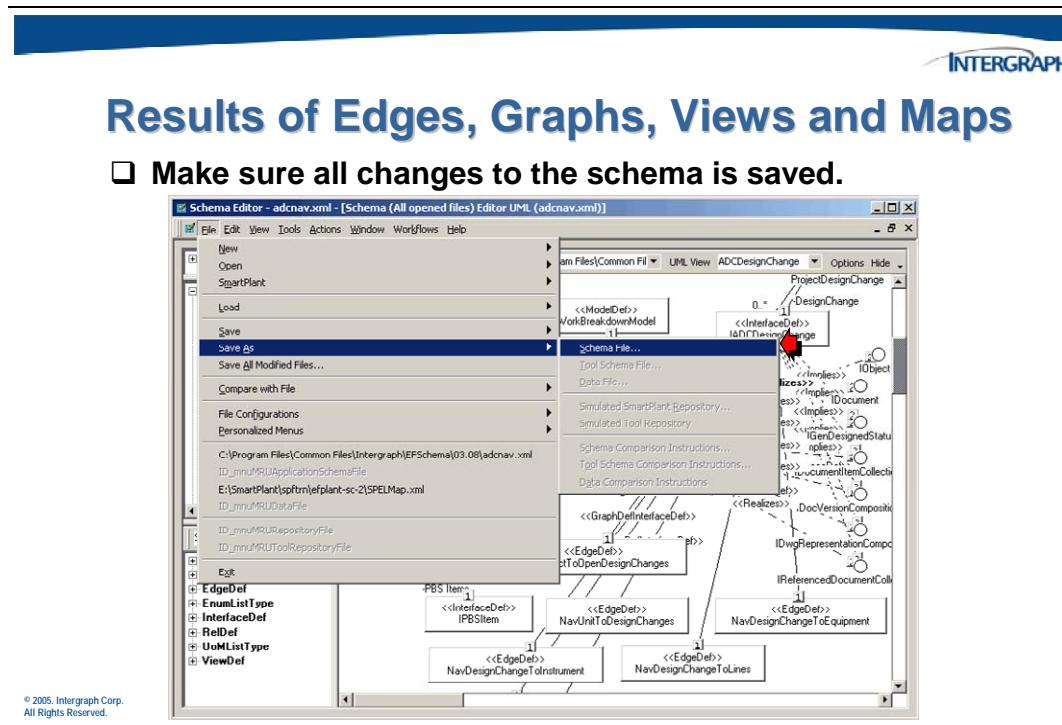
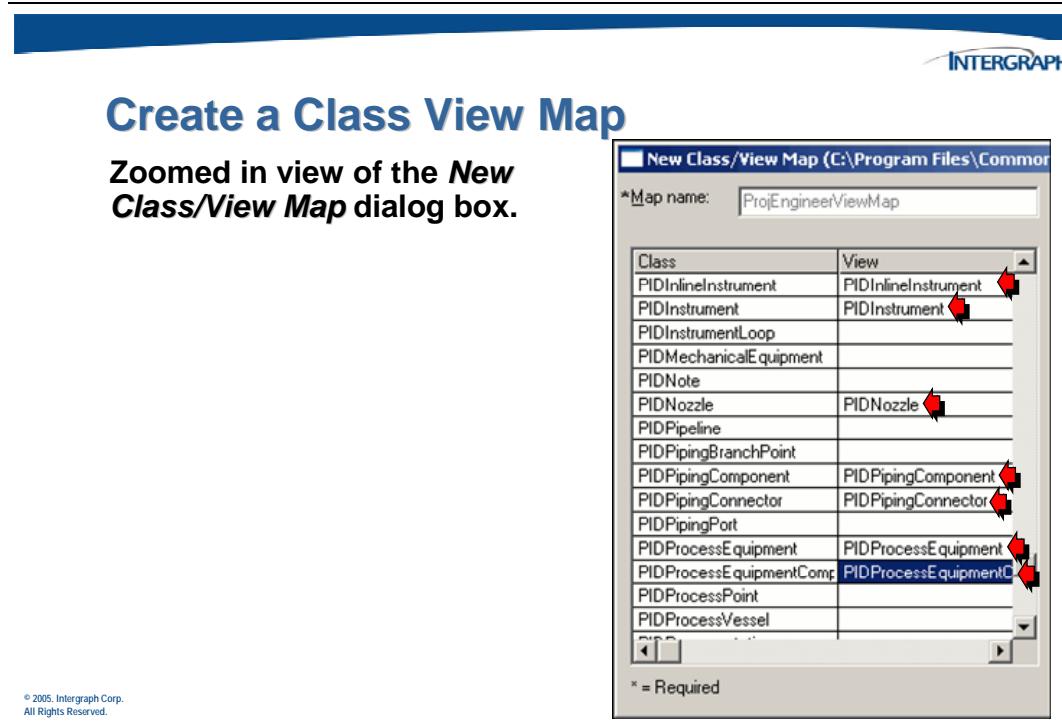


Create a Class View Map

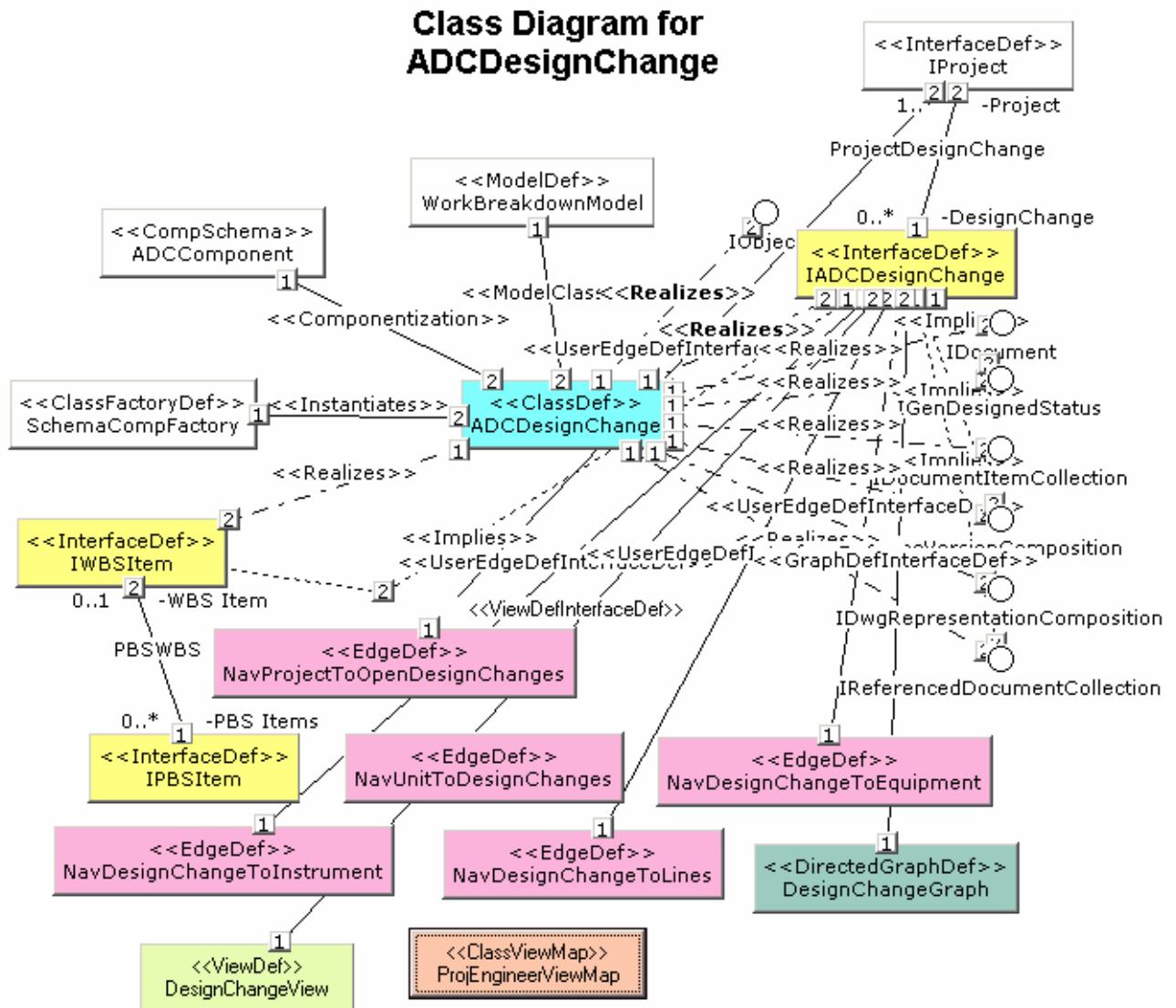
- Repeat the same sequence to define the remaining mappings.



© 2005, Intergraph Corp.
All Rights Reserved.



The UML diagram below shows all the changes that have been made to the schema.



6

C H A P T E R

Viewing and Finding Data

6. Viewing and Finding Data

Many sessions in the Schema Editor are typically performed to view the data files and not the schema files. When you want to view data files, you must open a schema file before you can open the data file. To view data organized by schema classes, it is often useful to view the data in terms of the schema.



Viewing a Data File

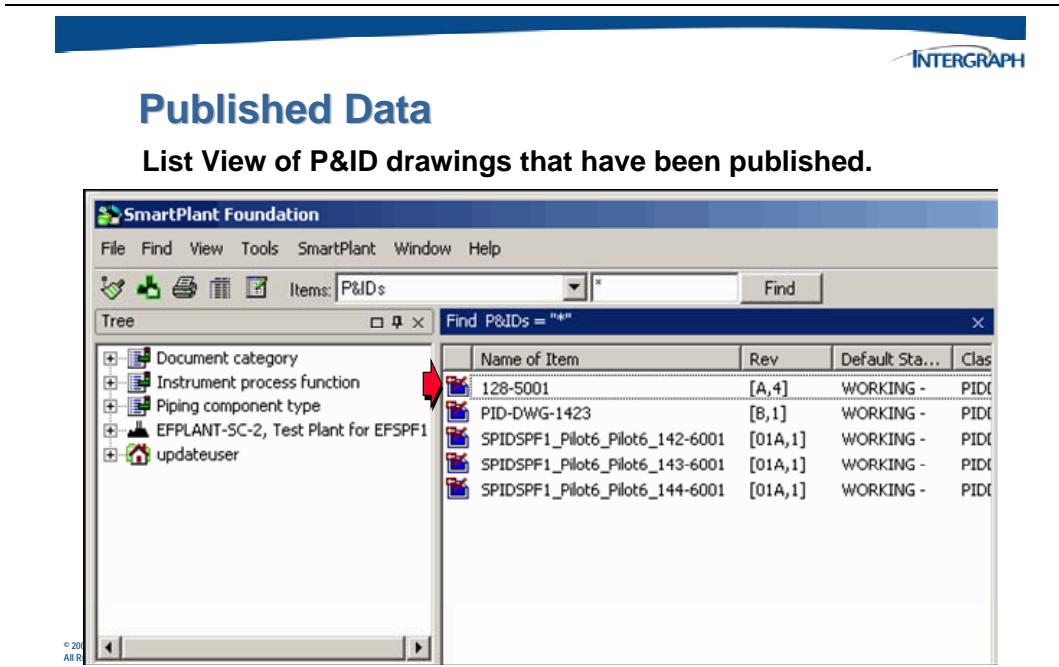
Many of the same views that are useful for viewing the schema are also useful for viewing data files.

Some of the most valuable data views include:

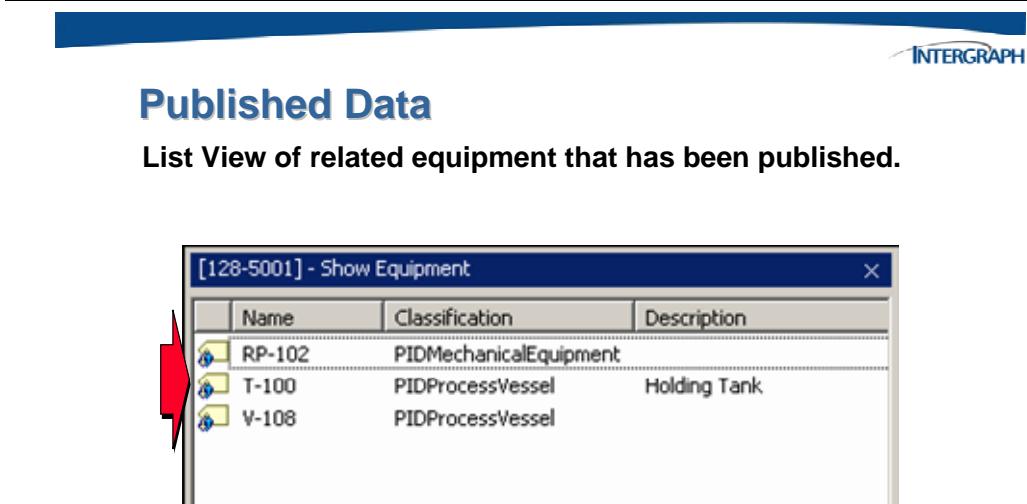
- Data Tree/UML View
- Data Tree/Table View
- Data Tree/Properties View
- Data Tree View

The data that can be viewed by the schema editor is the contents of the XML container that gets created as a result of performing a **Publish** operation from one of the engineering authoring tools.

For example, if drawing **128-5001** gets published from SmartPlant P&ID, the instantiation of the objects can be viewed in the SmartPlant Foundation *Desktop Client* browser.



Using the **Show Equipment** command will show the published equipment objects.



Using the **Show Instruments** command will show the published instrument objects.

Published Data

List View of related instruments that have been published.



Name	Classification	Description
ABV-128HV	PIDInlineInstrument	
SA-305	PIDInstrument	
SV-305	PIDInstrument	

© 2005, Intergraph Corp.
All Rights Reserved.

Using the **Show Lines** command will show the published pipeline objects.

Published Data

List View of related pipelines that have been published.



Name	Classification	Description
PH-101	PIDPipeline	
PH-104	PIDPipeline	

© 2005, Intergraph Corp.
All Rights Reserved.

6.1 XML Files Overview

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



XML Files

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.

Each publish includes at least data and metadata containers and may also contain an instructions (tombstones) container.

In XML files, 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 is sample data from a P&ID data container:

```

- <Container CompSchema="PIDComponent" Scope="Data" SoftwareVersion="03.08.00.12"
  SchemaVersion="03.08.00.04 Plus29922 MINUS Edge" Plant="EFPLANT-SC-2" Project=""
  DocUID="4CECEEA6B99D4490A557D9BFC4134435" DocName="128-5001" Revision="A"
  Version="4" ToolID="SMARTPLANTPID" ToolSignature="AABQ">
- <PIDDrawing>
  <IObject UID="4CECEEA6B99D4490A557D9BFC4134435" Name="128-5001"
    Description="" />
  <IDocument DocCategory="P&ID Documents" DocTitle="" DocType="P&ID" DocSubtype="" />
  <IDocVersionComposition />
  <IDwgRepresentationComposition />
  <IPIDDrawing />
  <ISchematicDwg />
</PIDDrawing>
- <PIDInlineInstrument>
  <IObject UID="04992AEB25CA489AA8BAA8242683B5A9" Name="ABV-128HV" />
  <IInlineInstrument />
  <IInstrument ProcFunc="@EE3F8" InstrumentType3="@EE425" InstrumentType="@EE3F8.1" />
  <IInstrumentOcc />
  <IPBSItem ConstructionStatus="@NewConstruction" ConstructionStatus2="@{78398AB4-
    9F3D-11D6-BDA7-00104BCC2B69}" />
  <IPipingPortComposition />
  <IPlannedMatl />
  <IDrawingItem />
  <IInlineInstrumentOcc />

<IPressureReliefItem />
<IProcessPointCollection />
<ISignalPortComposition />
<IDocumentItem />
<IElecPowerConsumer />
<IPart />
<INoteCollection />
<INamedInstrument InstrFuncModifier="ABV" InstrLoopSuffix="" InstrTagPrefix="Unit1"
  InstrTagSequenceNo="128" InstrTagSuffix="HV" MeasuredVariable="" />
<IPipeCrossSectionItem NominalDiameter="4 in" />
<IPipingSpecifiedItem PipingMaterialsClass="ABV" />
<IIInsulatedItem InsulCompositeMatl="@{12EB29A1-B69C-11D6-BDB9-00104BCC2B69}" />
<IReliefDevice OrificeLetter="@{96C5801B-9C2D-4E74-94CD-50890AAC614F}" />
<IInlineComponent IsFlowDirectional="False" />
</PIDInlineInstrument>
- <Rel>
  <IObject UID="PBS-04992AEB25CA489AA8BAA8242683B5A9-N3-FMCUHPQRGJCA-
    RDAWMMF3" />
  <IRel UID1="04992AEB25CA489AA8BAA8242683B5A9" UID2="N3-FMCUHPQRGJCA-
    RDAWMMF3" DefUID="PBSItemCollection" />
</Rel>
- <PIDInstrument>
  <IObject UID="9F1EFE60E26D47B6B4A8D0D98FF444E8" Name="SA-305" />
  <IInstrument />

```

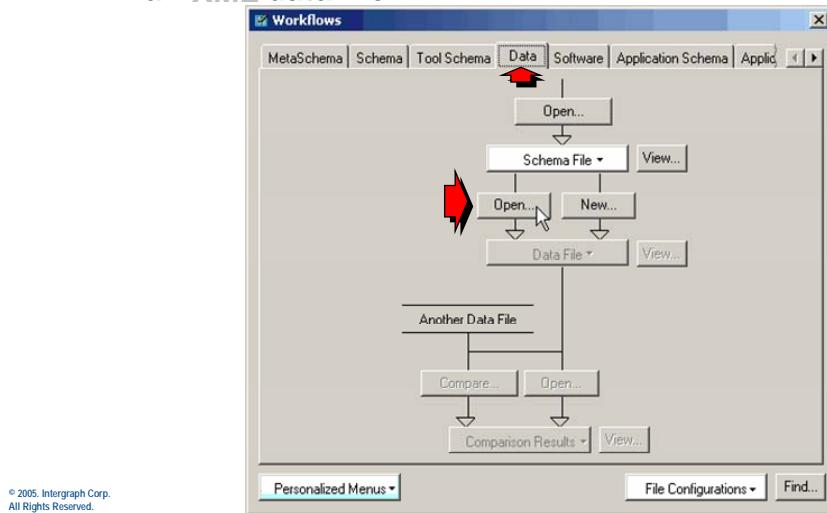
6.2 Opening and Viewing a Data File

After you open a schema file using a saved configuration, you can open an XML data file by selecting the *Data* tab and using the **Open** button. Before you can open a data file, **you must open a schema**.

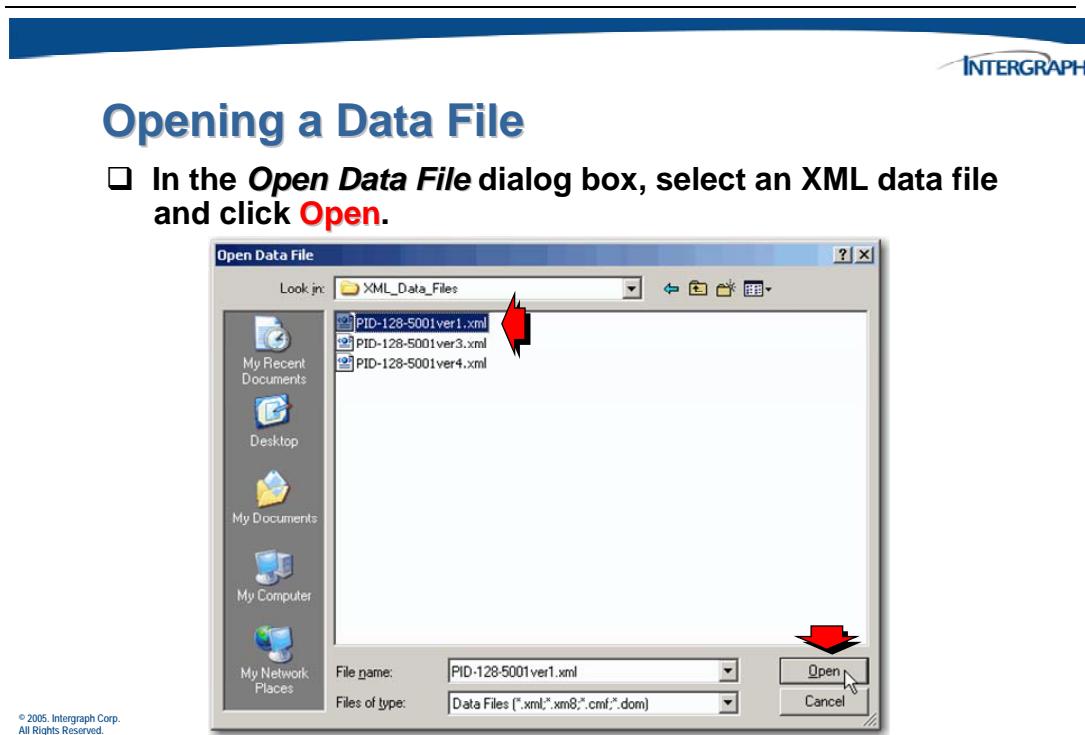


Opening a Data File

- ❑ In the *Workflows* dialog box, click the **Open** button to open an **XML** data file.



An *Open Data File* dialog box appears.



In the examples in this chapter, a published XML data file has been copied and renamed to make finding the file easier. In reality, the XML files to be viewed using the schema editor can be found in an SPF vault.

6.2.1 Data File Tree/UML View

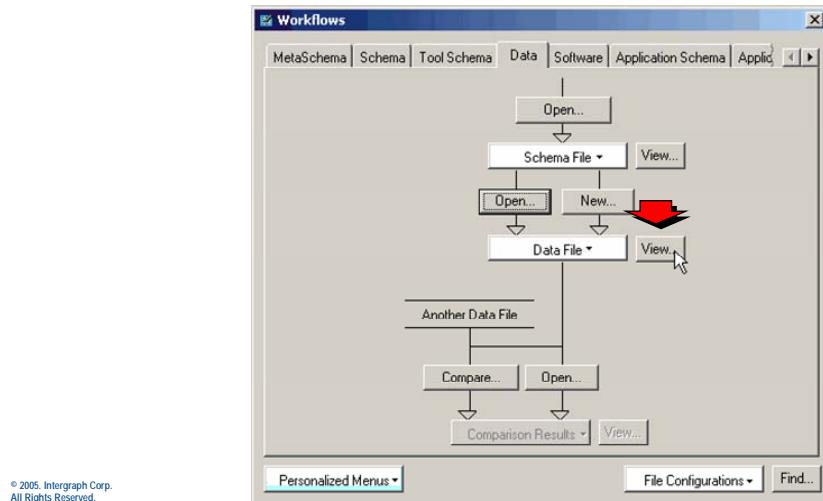
If you remember from the previous chapter, *Using the Schema Editor*, the **Tree/UML** view is a combination of the standard tree view with a UML view. The UML view provides a graphical representation of the relationships and relationship definitions involving the object selected in the tree view. In the UML view, you can click any object to center the UML around the selected object to view its relationships and relationship definitions.

First, make sure a schema file has been opened.



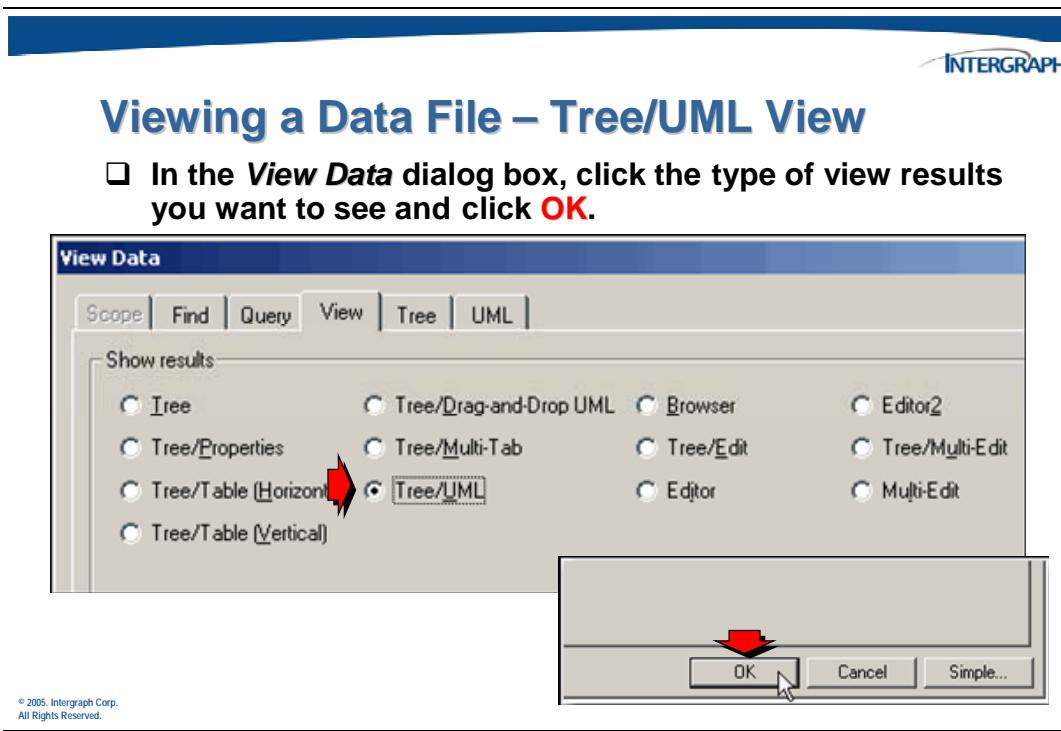
Viewing a Data File – Tree/UML View

- From the **Workflows** dialog box, click the **View** button to view the open data file.

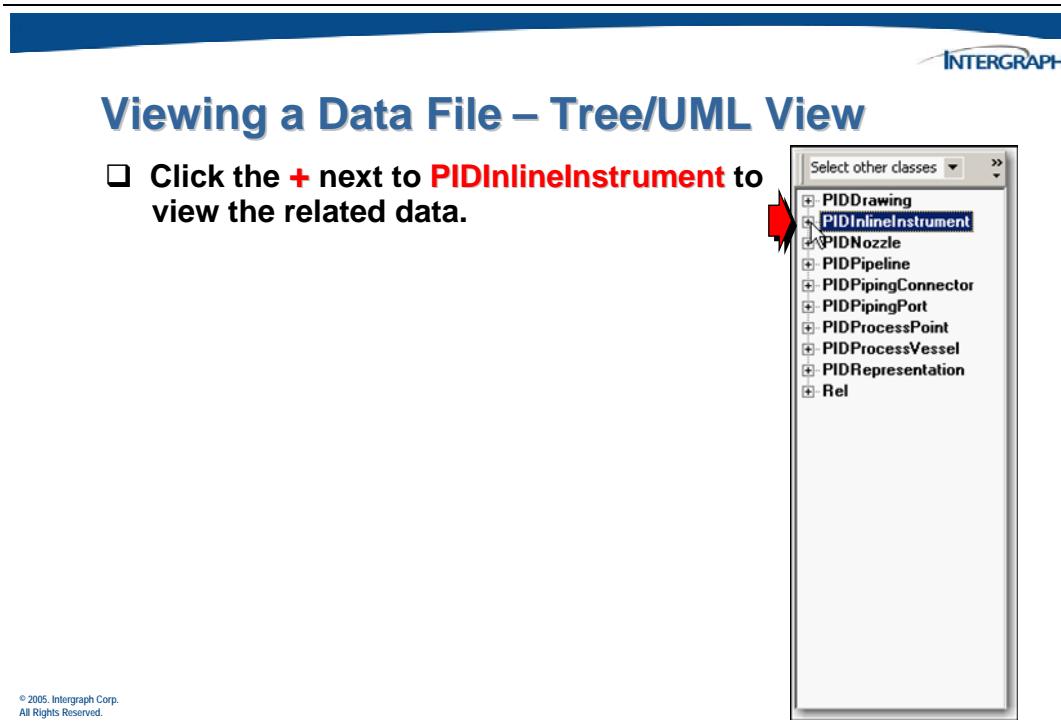


The UML view is useful for viewing relationships in the data file graphically.

The *View Data* dialog box appears.



The *Data Tree/UML* view window displays. Drill down and expand the tree to see related data.



Select the object to be viewed into the right pane.

Viewing a Data File – Tree/UML View

Click one of the instruments (**ABV-128HV**) to view the details in the right pane.

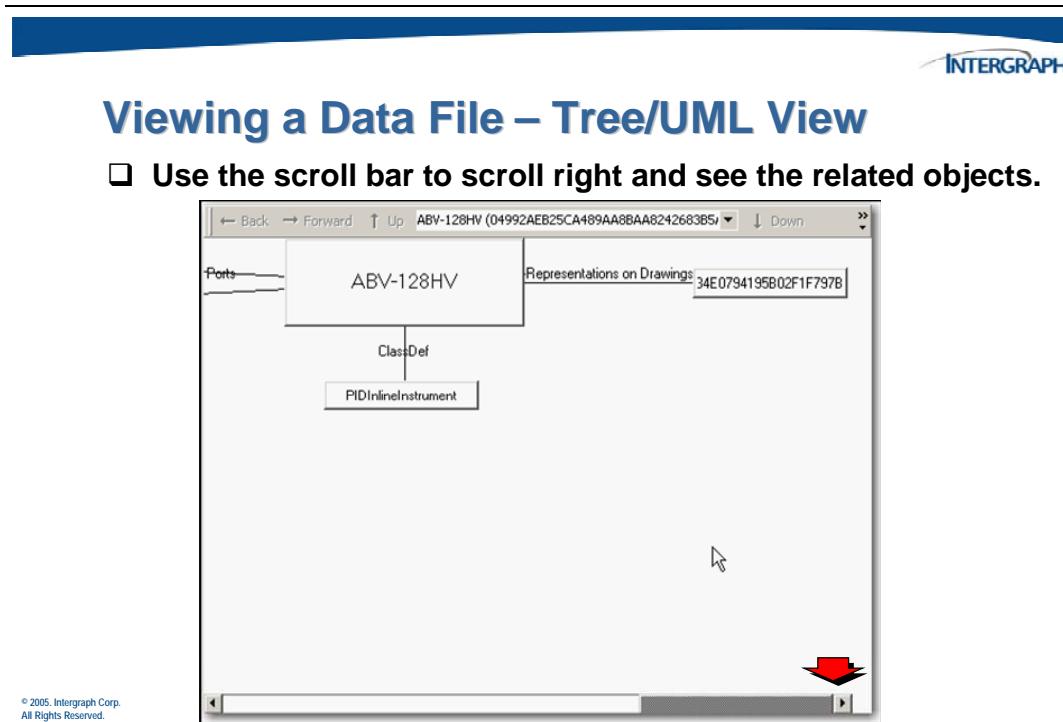
© 2005, Intergraph Corp.
All Rights Reserved.

Viewing a Data File – Tree/UML View

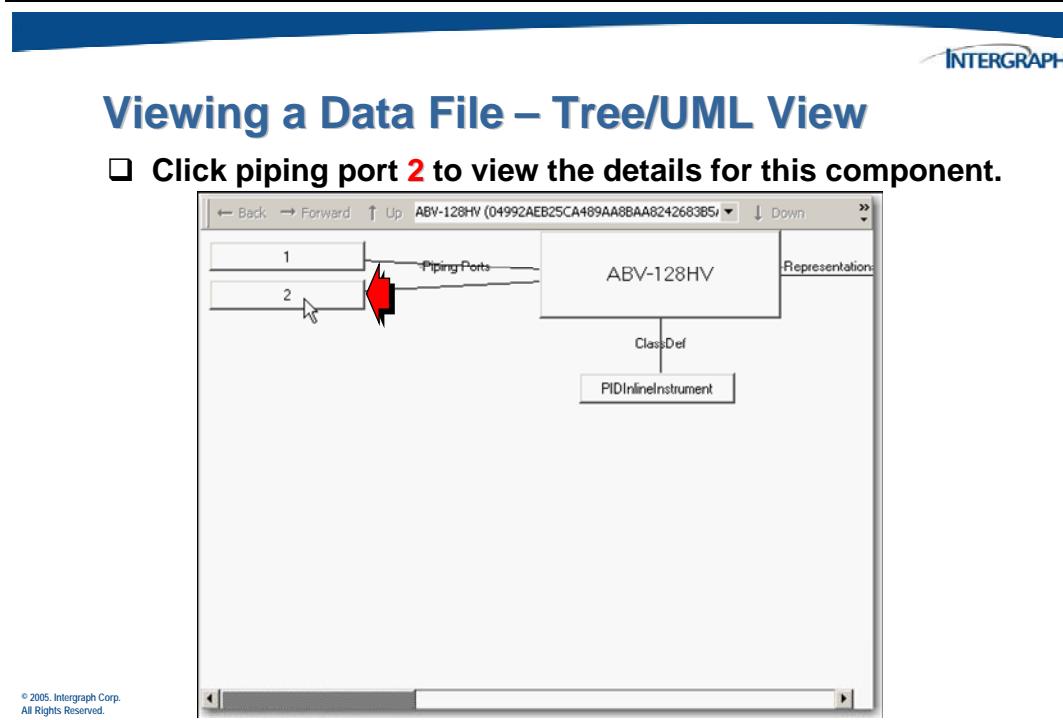
The view will change to show **ABV-128HV** and related components.

© 2005, Intergraph Corp.
All Rights Reserved.

Scroll right to see additional information associated with this object.



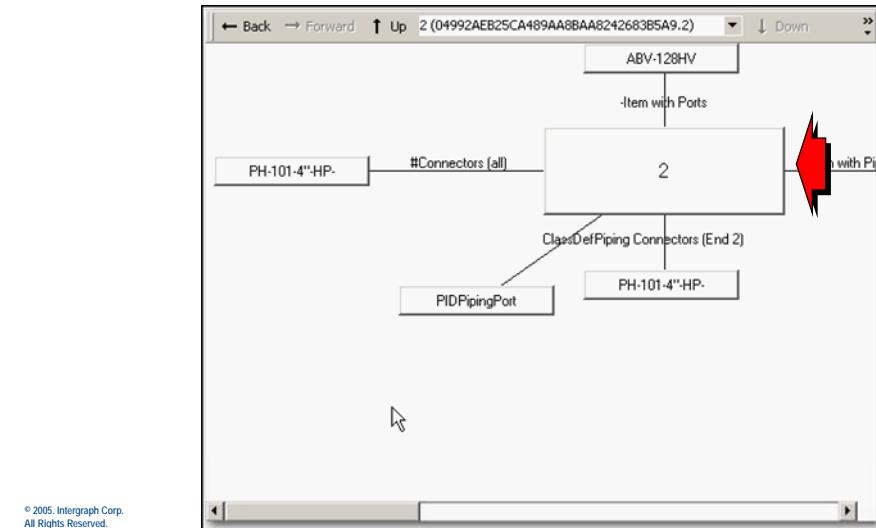
Click any of the related objects in the UML view to display the details for that object.





Viewing a Data File – Tree/UML View

Zoomed in view of piping port **2** in the *Data Tree/UML* view.

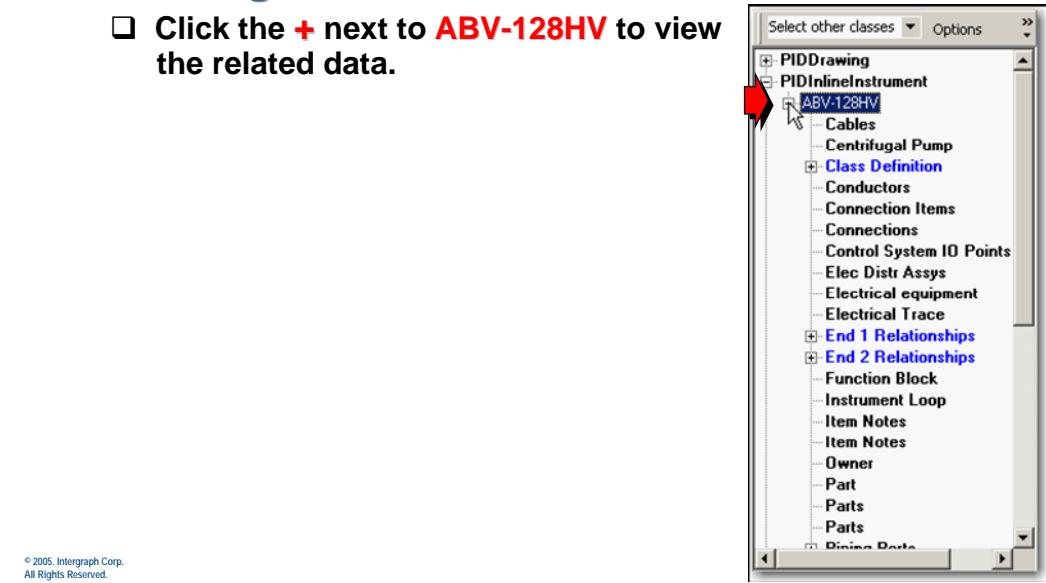


Continue to drill down in the tree view to see more data.



Viewing a Data File – Tree/UML View

- Click the **+** next to **ABV-128HV** to view the related data.



Viewing a Data File – Tree/UML View

Continue to drill down by clicking the + next to **Piping Ports** to view related data.

© 2005, Intergraph Corp.
All Rights Reserved.

Select the object to be viewed in the right pane.

Viewing a Data File – Tree/UML View

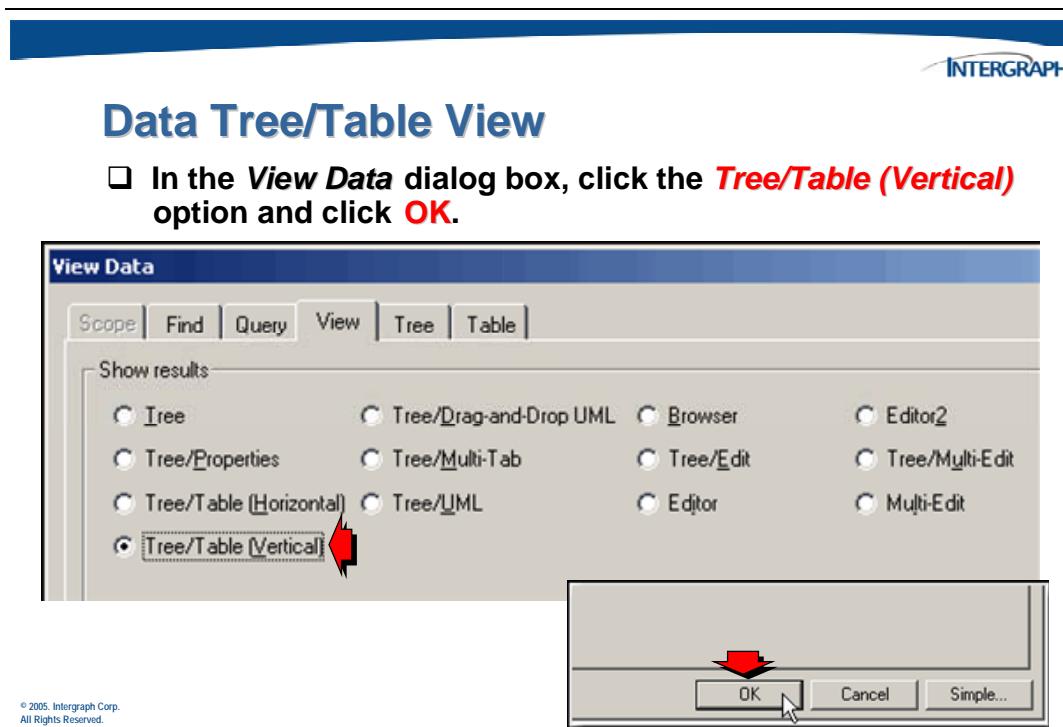
Click **Piping Port 1** to view the details for this object.

© 2005, Intergraph Corp.
All Rights Reserved.

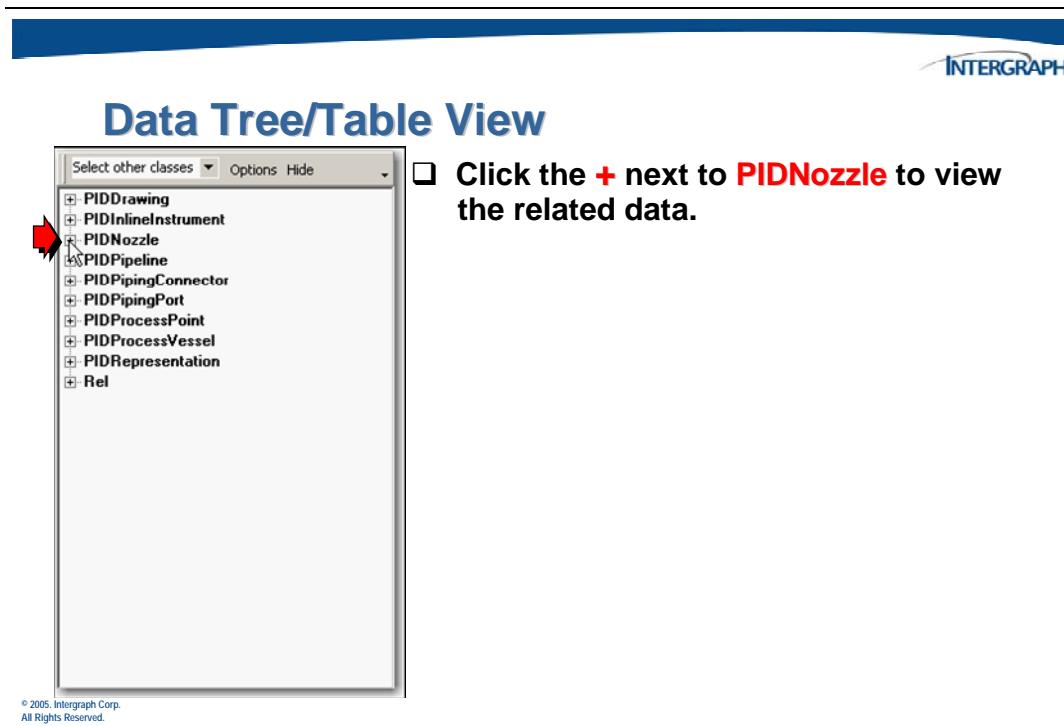
6.2.2 Data Tree/Table View

Like the Schema Tree/Table view, the Data **Tree/Table** view allows you to traverse relationships in the tree and see information for items selected in the tree view in a table format.

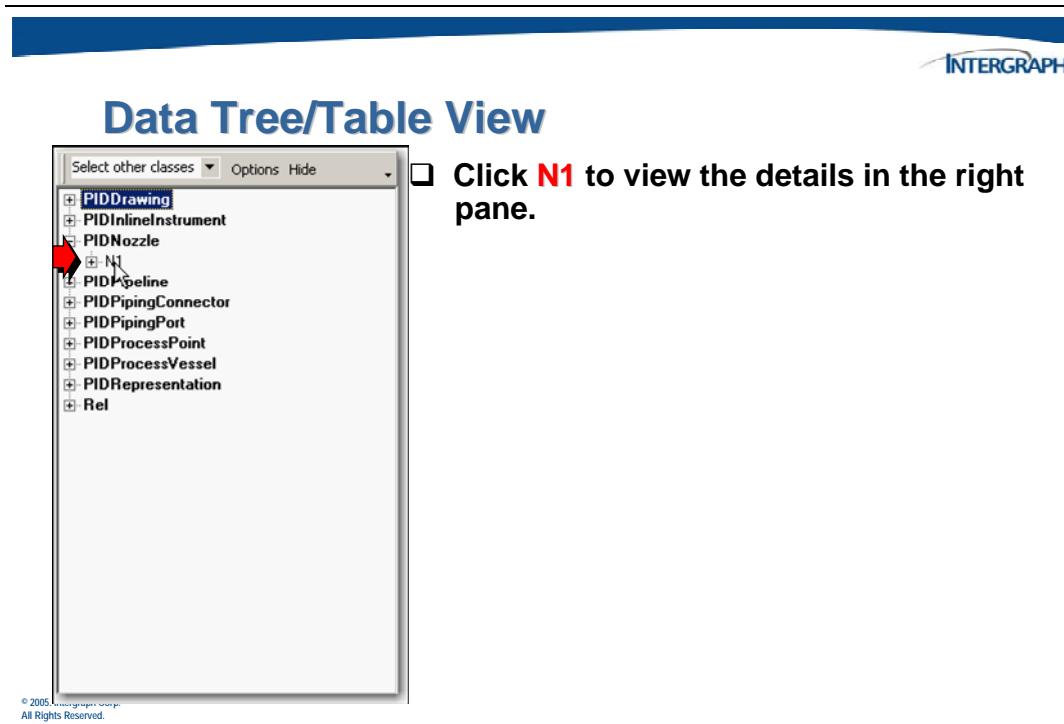
Click the **View** tab in the *View Data* dialog box.



The *Data Tree/Table* window appears.



To traverse the relationships in the tree view, expand the nodes.



The vertical table displays row headers along the left side of the table and data in vertical columns. In the following example, a **Vertical** table display is used.

The screenshot shows the Data Tree/Table View interface. On the left, there is a tree view of P&ID objects. A red arrow points from the 'N1' node in the tree view to the corresponding row in the vertical table on the right. The right pane displays a vertical table with the following data:

	N1
UID	EB3C26F0C3AE49388C9A5BDBD85AC6C3
Name	N1
Description	
Dry_Striped_Weight	
Dry_Installed_Weight	
Wet_Operating_Weight	
DesignWeight	
TestWeight	
CoatingRequirement	
CoatingType	
Exterior_CoatingArea	
Exterior_SurfaceTreatmentType	
Exterior_SurfaceTreatmentRequirement	
AuxiliaryTreatmentRequirement	
AuxiliaryTreatmentType	
CoatingColor	
InsulatedItem_OperatingTemperature	
InsulationSurfaceArea	
InsulationWeight	
InsulTemp	
InsulThickSrc	
InsulSpec	
InsulCompositeMall	@{12EB2941-B69C-11D6-BDB9-00104BCC2
AvgInsulDens	

© 2005, Intergraph Corp.
All Rights Reserved.

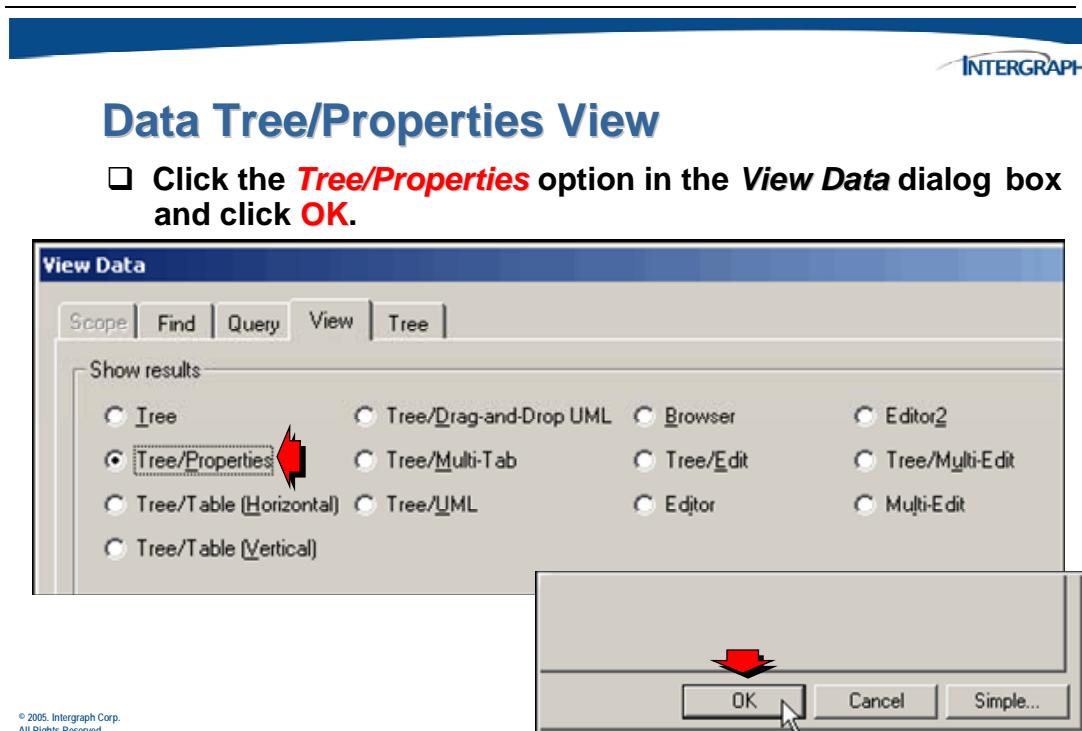
6.2.3 Data Tree/Properties View

The **Tree/Properties** view combines the same tree view used elsewhere in the Schema Editor with a properties view that emulates display of properties in the right pane in SmartPlant Foundation. This properties view identifies each interface for the selected object with text on a shaded background. Following each interface are the properties exposed by that interface and their values for the selected object.

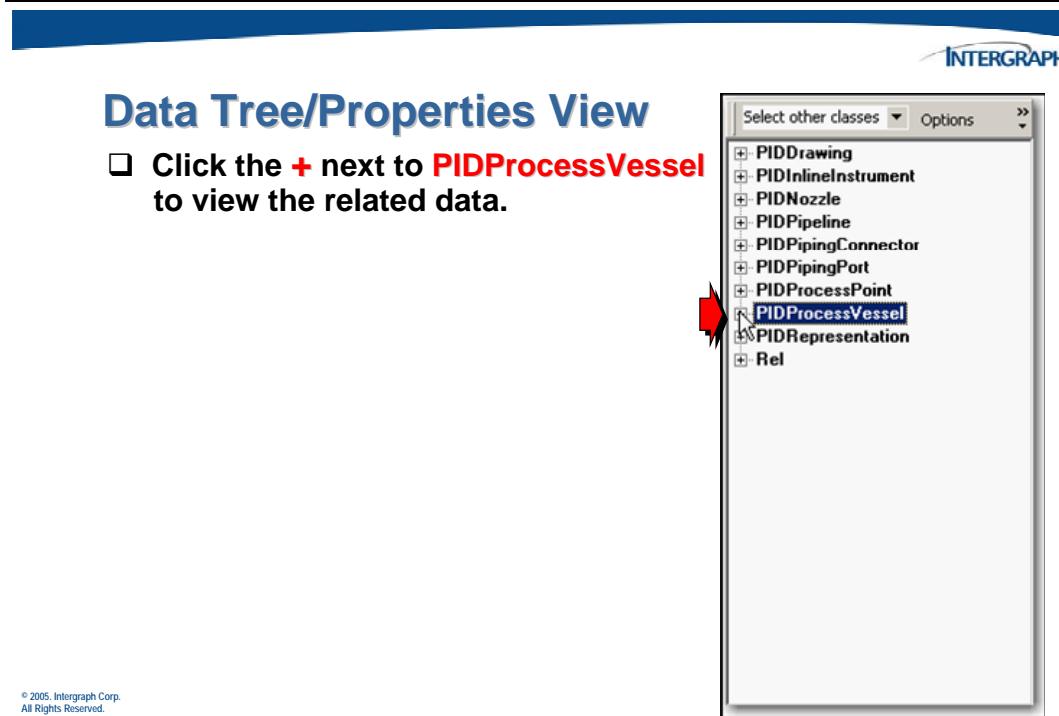
The properties view is currently the only view in the Schema Editor that explicitly converts enumerated property values from the UID of the enumerated entry to the name associated with that entry. Other views display the UID for the enumerated entry as it would appear in the XML file, such as @EE407.

The Tree/Properties view provides the most useful information when you use it to view data, instead of the schema, tool schemas, or meta schema. If you select an object in the tree view, you can see its interfaces in the properties view.

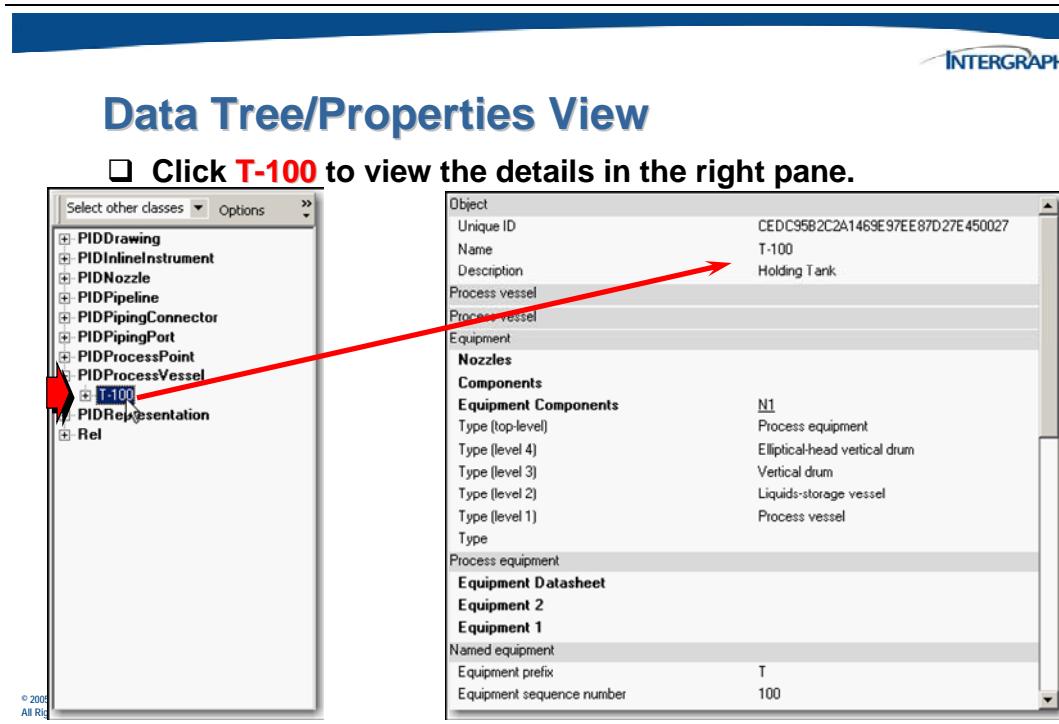
Click the **View** tab in the *View Data* dialog box.



The *Data Tree/Properties* window appears. Drill down to expand the tree to see other objects.



Select the object that you want to see in the Properties view.



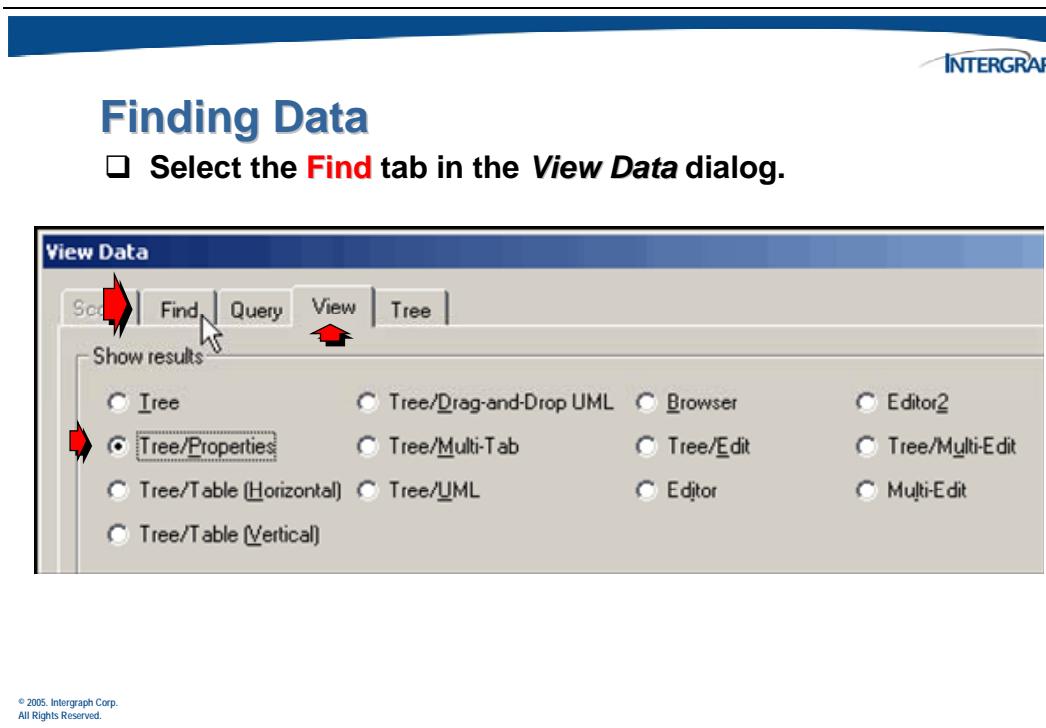
6.3 Finding Data Objects

In the Schema Editor, you can locate particular objects in 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.

6.3.1 View Data Find Tab

To find an object in the Schema Editor, choose either the *Edit > Find* menu command or use the **Find** tab in the *View Data* dialog box. This allows you to define search criteria and to limit the scope of your search to the data files. Using the *Find* dialog box is especially useful if you know part of the UID, name, or description of the object that you want to find.

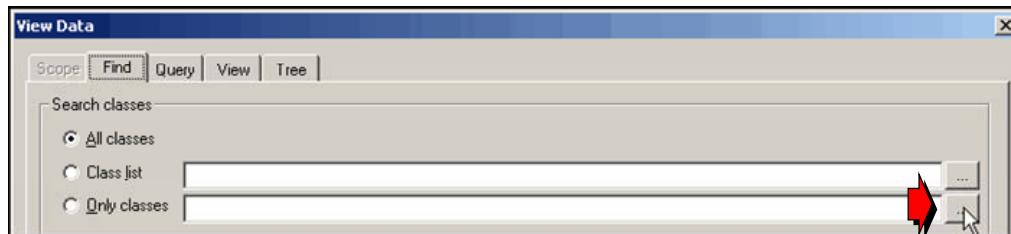
In the following example, click the *Data File View* button first, then click the **Find** tab in the *View Data* dialog box.





Finding Data

- Click the browse button (...) to display a list of classes.



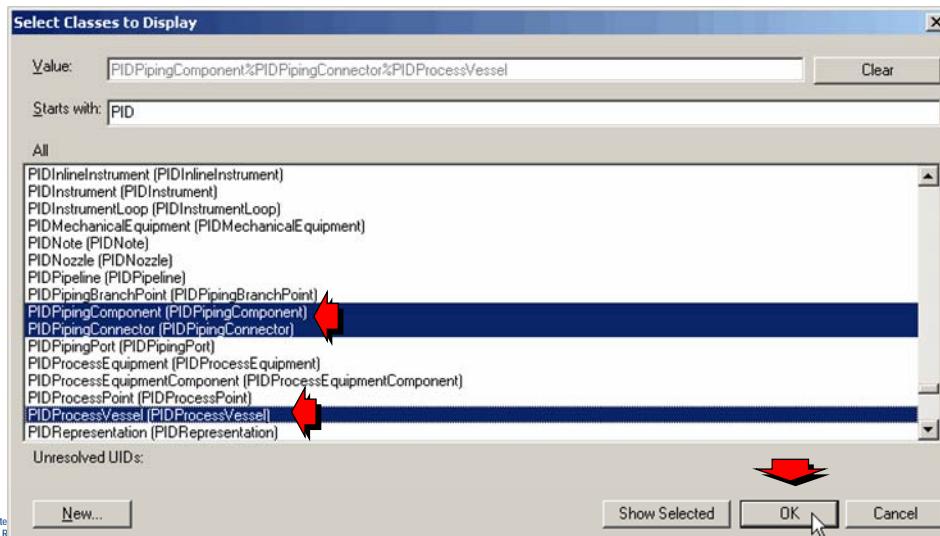
© 2005, Intergraph Corp.
All Rights Reserved.

The *Select Classes to Display* dialog box appears.



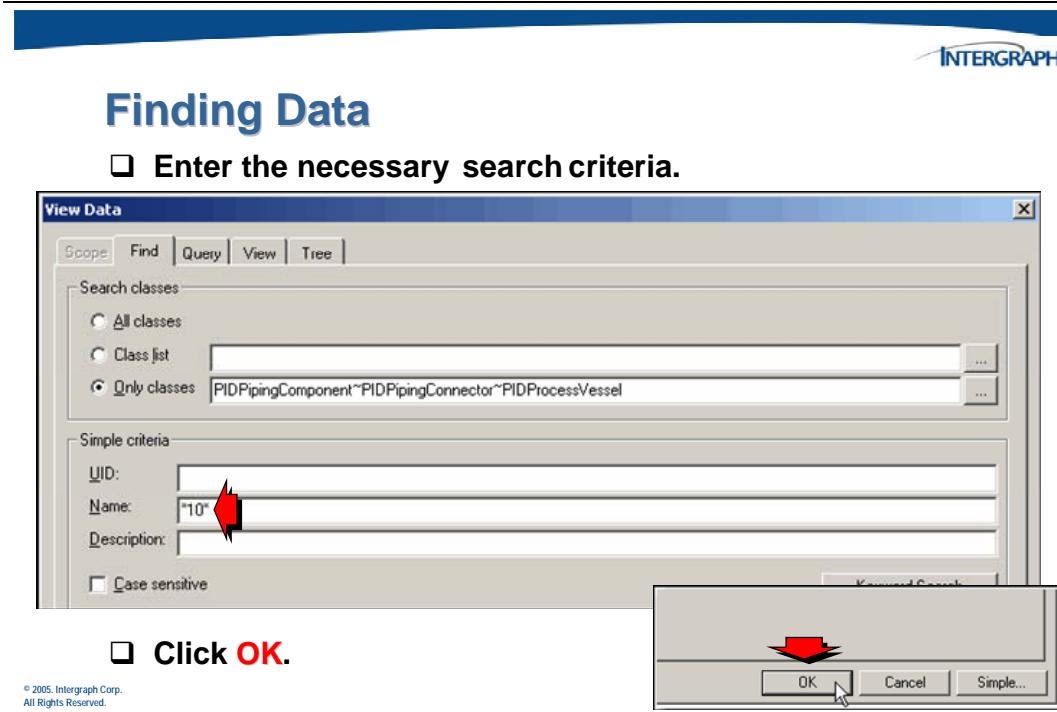
Finding Data

- Select one or more classes and click **OK**.

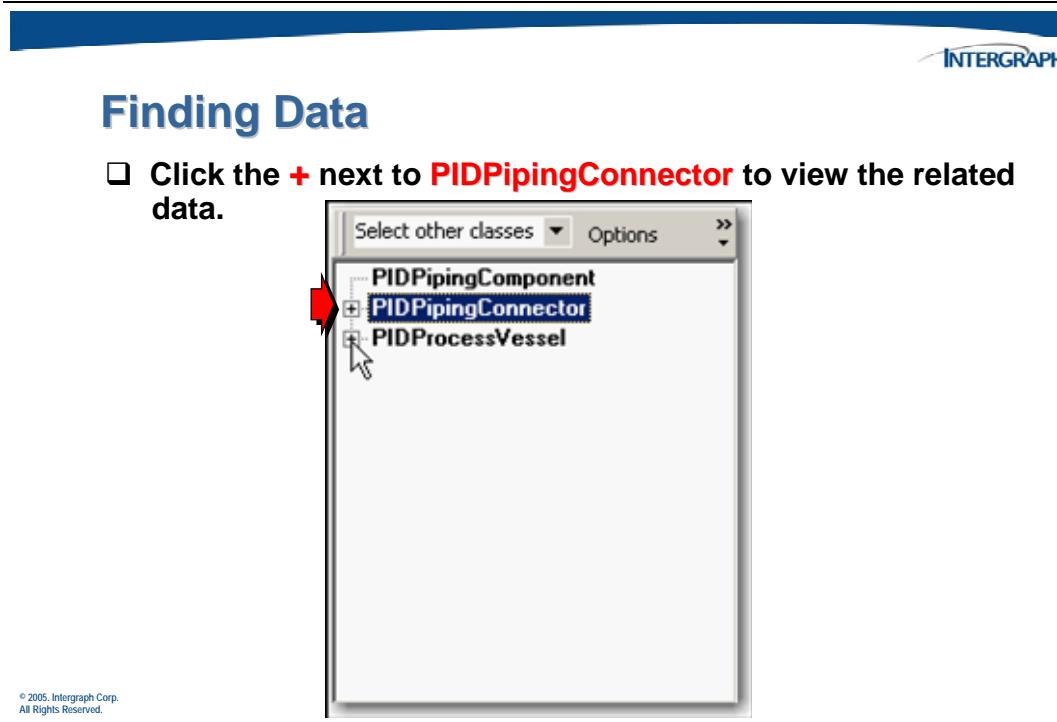


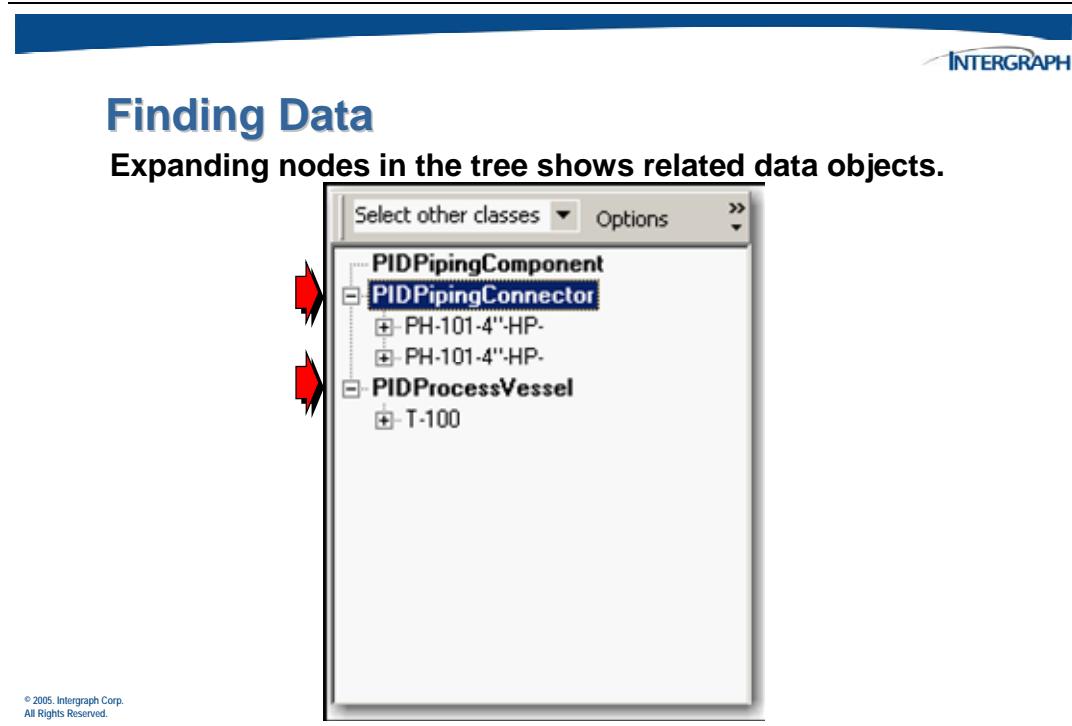
© 2005, Inte
All Rights R

In this example, the software searches for all *PIDPipingComponent*, *PIDPipingConnector* and *PIDProcessVessel* objects that have *10* in the *Name*.

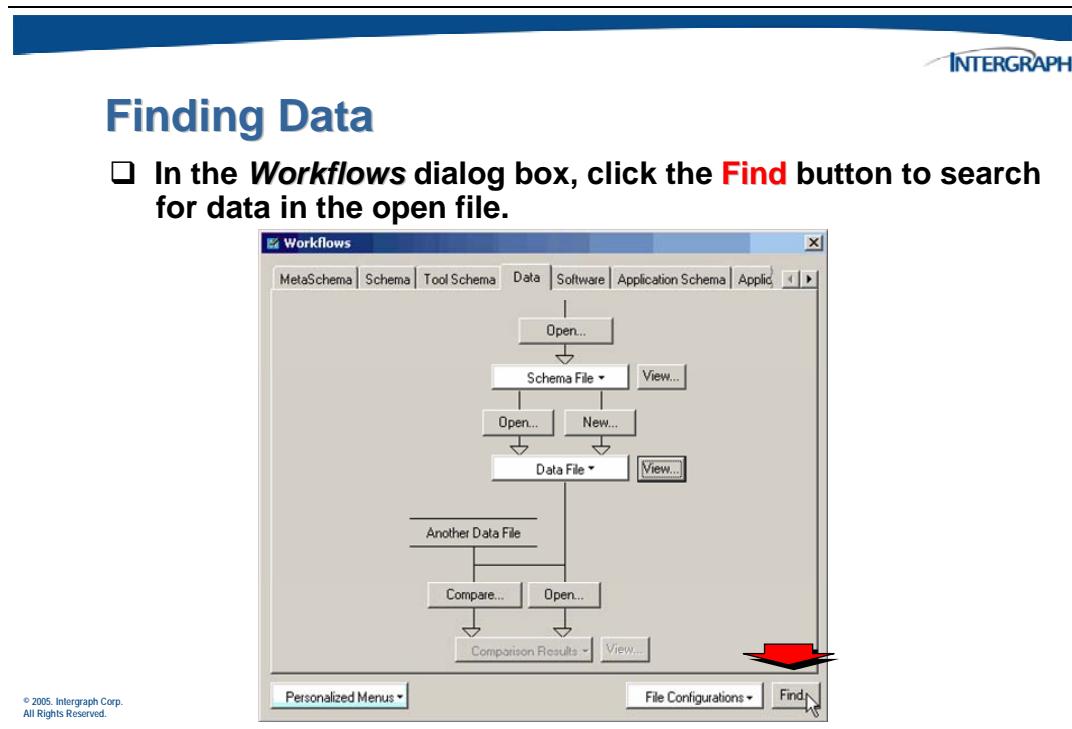


A Data Tree/Properties view displays the results of the search. Expand the tree to see related data.





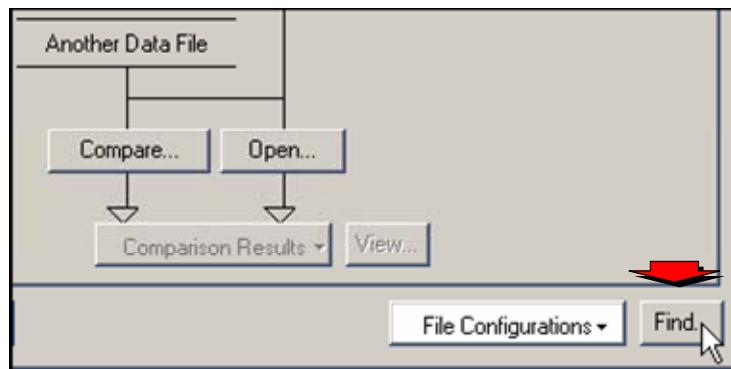
In the following example, the **Find** button in the *Workflows* dialog box is used to find objects.





Finding Data

Zoomed in view of the **Workflows** dialog box.



© 2005, Intergraph Corp.
All Rights Reserved.

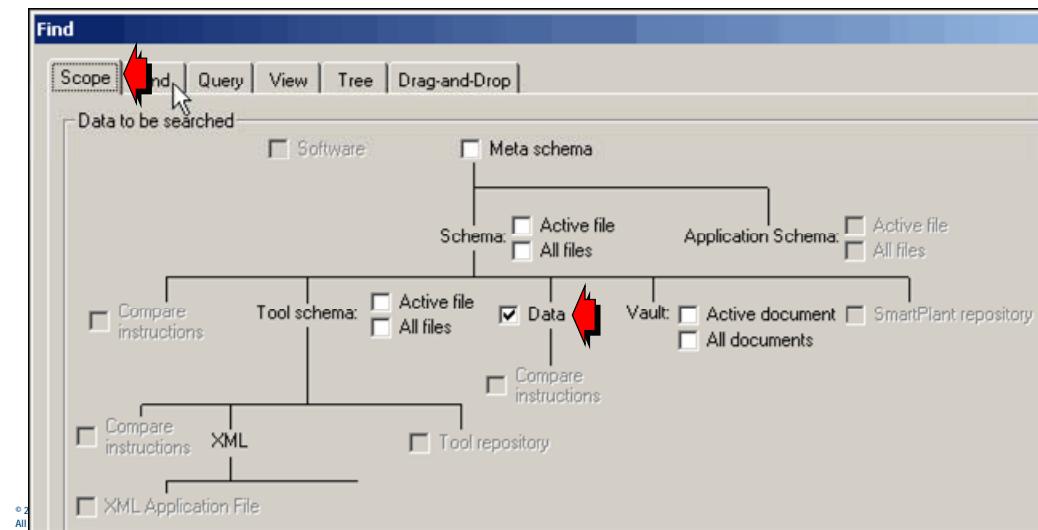
Click the **Scope** tab and select the appropriate check boxes to identify what data you want to include in your search.

For example, if you want to search the data file for a particular object, select the **Data** check box.

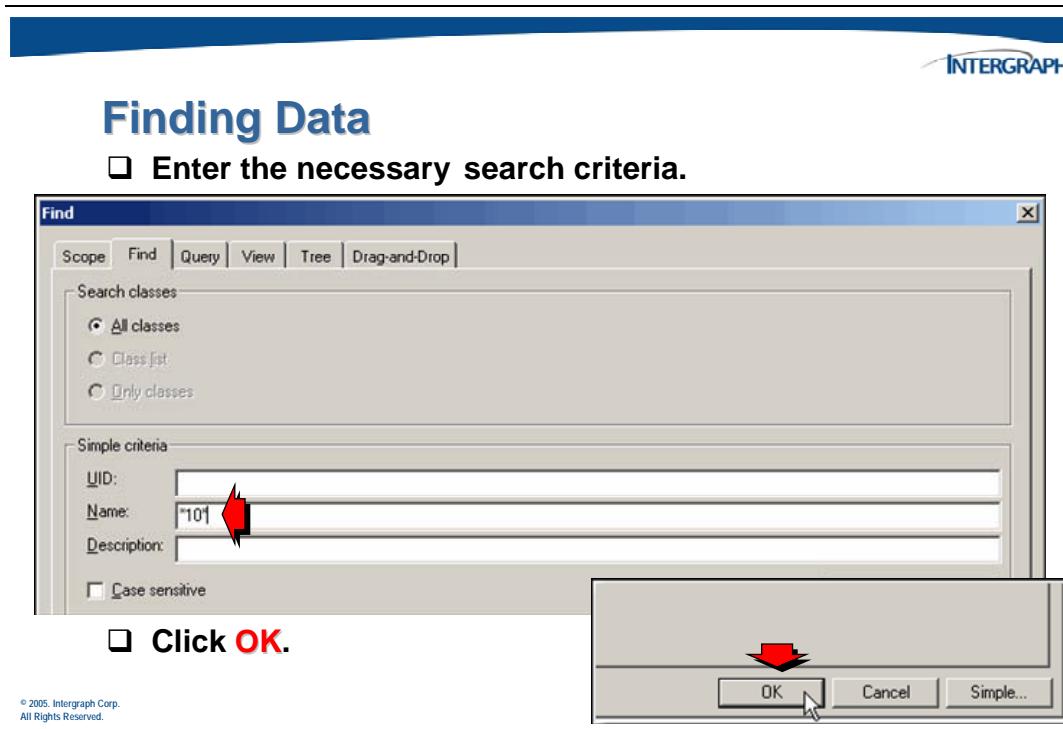


Finding Data

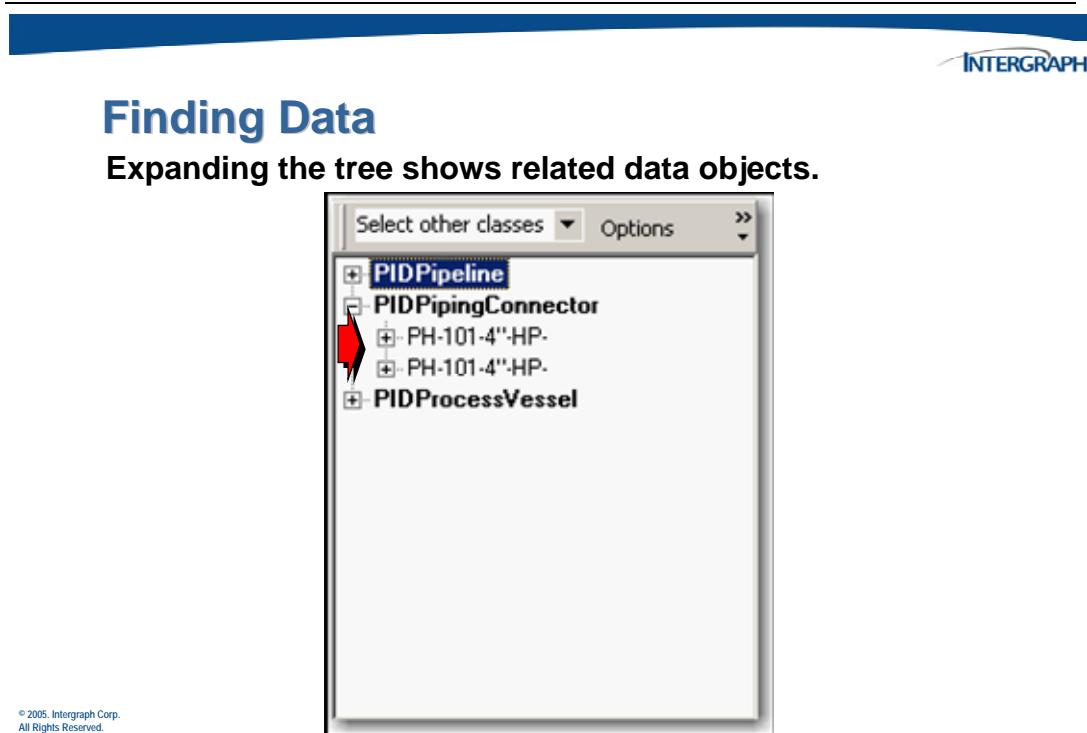
You can select the scope of the search to be performed.



Click the **Find** tab and enter the search criteria to locate the desired data.



In this example, the software searches for all objects that have *10* in the *Name* from *All classes*. A Data Tree/Properties view displays the results of the search.



6.4 Comparing Data Files

In the Schema Editor, you can compare schemas, tool 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 Framework authoring tools, and for seeing differences between the schemas from different authoring tools.



Comparing Files

There are **three** types of comparisons in the Schema Editor:

- Full Comparison**
- Detection of Tombstones**
- Comparison Using Views**

© 2005. Intergraph Corp.
All Rights Reserved.

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.

Detection of Tombstones

This type of comparison will be discussed in detail in a later section.

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.

6.4.1 Full Comparison

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.

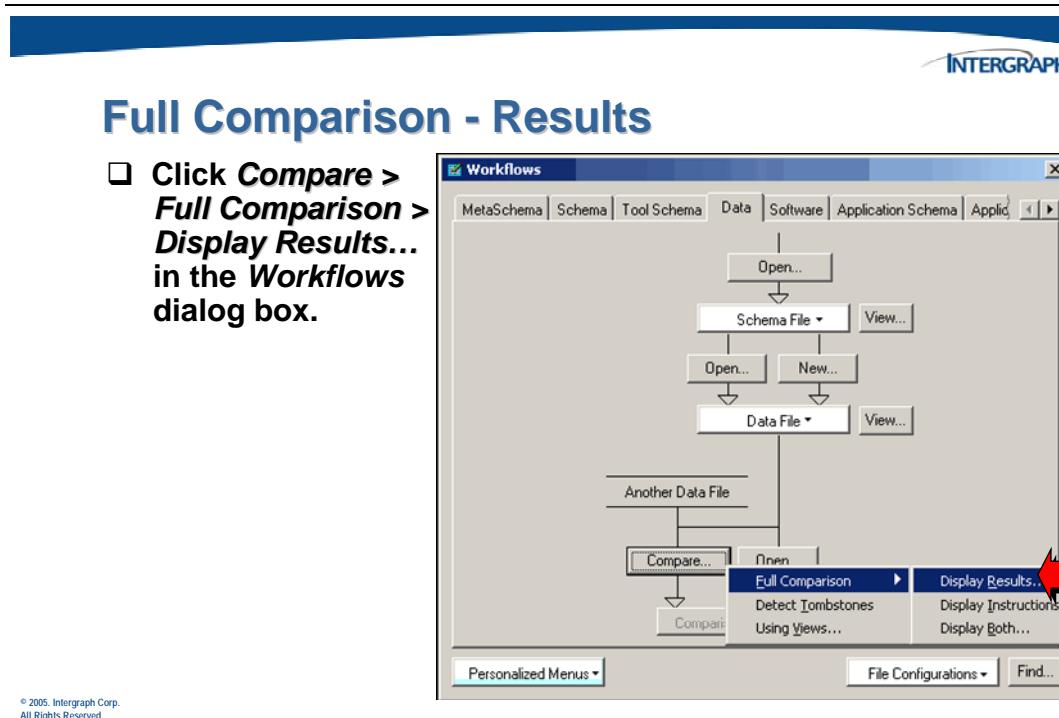
Performing a full comparison of two files, you can choose whether you want to display the comparison results alone, the comparison instructions, or both the comparison results and instructions. The comparison results list the objects that appear in one file, but not the other, and the objects that are different in the two files. The comparison instructions are a set of instruction objects that define what would happen in one of the authoring tools when one file is superceded by the other.

In this section, you will see the following comparison examples:

- A full comparison that displays comparison results
- A full comparison that displays comparison instructions

In the Schema Editor, you can also choose to view both comparison results and comparison instructions when you compare files.

To perform a full comparison and display the comparison results, first, open the data file to which you want to compare another file.

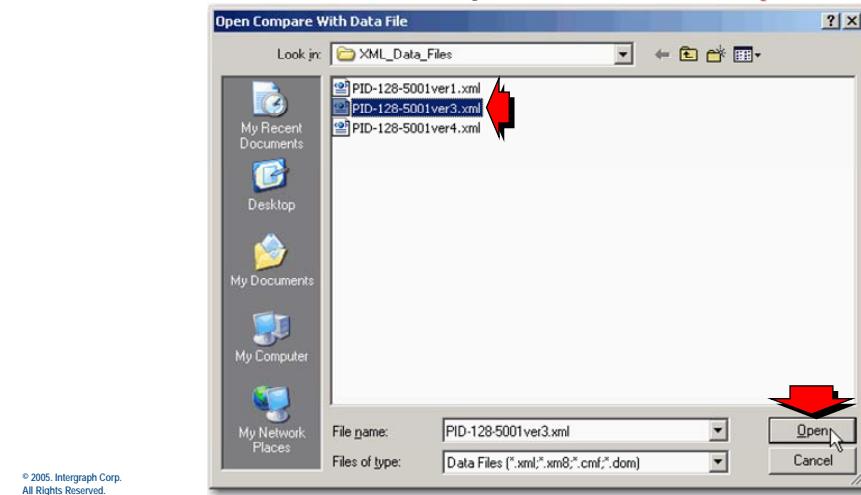


An *Open Compare With Data File* dialog box displays.



Full Comparison - Results

- In the **Open Compare With Data File** dialog box, select an XML data file for comparison and click **Open**.

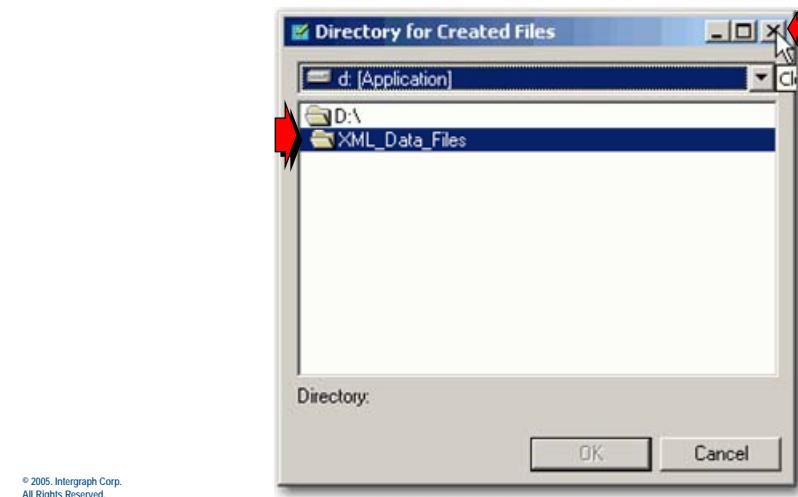


A *Directory for Created Files* dialog appears.



Full Comparison - Results

- Specify the folder in which you want to store the comparison results output file.

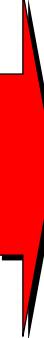


Comparison results appear in *Notepad* for viewing. The comparison results list the objects that appear in one file, but not the other, and the objects that are different in the two files.



Full Comparison - Results

A text output file containing the comparison results is created and displayed in Notepad.



```
EF544.txt - Notepad
File Edit Format View Help
*****
only In D:\XML_Data_Files\PID-128-500iver1.xml
*****
only In D:\XML_Data_Files\PID-128-500iver3.xml
*****
PIDNozzle N2 (20EF7E077CCD43C0B184CABD9C78D934)
Rel1 (E0C-20EF7E077CCD43C0B184CABD9C78D934)
End 1 = N2 (20EF7E077CCD43C0B184CABD9C78D934)
End 2 = V-108 (E19B534810444F1CA62EF1FCC97CC57)
RelDef = EquipmentComponentComposition
PIDNozzle N1 (2BD973F1096F44AE098F7DF14CB5110)
Rel1 (E0C-2BD973F1096F44AE098F7DF14CB5110)
End 1 = N1 (2BD973F1096F44AE098F7DF14CB5110)
End 2 = RP-102 (929ACE2B2E76462997E62B199298C3E6)
RelDef = EquipmentComponentComposition
PIDNozzle N2 (0E53C538592848BEB6B7FB2E03C3465A)
Rel1 (E0C-DE53C538592848BEB6B7FB2E03C3465A)
End 1 = N2 (0E53C538592848BEB6B7FB2E03C3465A)
End 2 = RP-102 (929ACE2B2E76462997E62B199298C3E6)
RelDef = EquipmentComponentComposition
PIDPipingConnector PH-104-4-HP- (B0A147124FC04C8FA552633F4C40D43E)
PIDProcessPoint (B0A147124FC04C8FA552633F4C40D43E.PPT)
Rel1 (PRP-B0A147124FC04C8FA552633F4C40D43E.PPT)
End 1 = (B0A147124FC04C8FA552633F4C40D43E.PPT)
End 2 = PH-104-4-HP- (B0A147124FC04C8FA552633F4C40D43E)
RelDef = ProcessPointCollection
RelCN-B0A147124FC04C8FA552633F4C40D43E
End 1 = PH-104-4-HP- (B0A147124FC04C8FA552633F4C40D43E)
End 2 = PH-104 (936EA3C4986F435BA261A38DE61CA91B)

```

© 2005, Intergraph Corp.
All Rights Reserved.

The output will display any new or changed objects in the compared XML file.



Full Comparison - Results

Comparison results (con't)

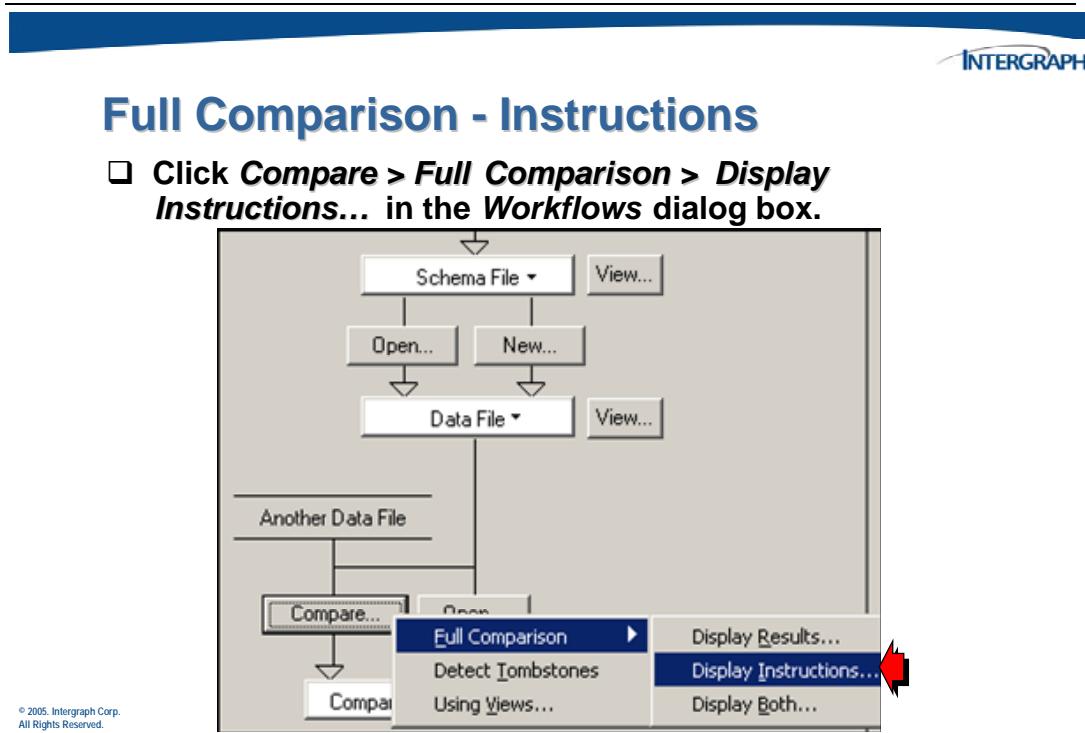


```
EF511.txt - Notepad
File Edit Format View Help
*****
End 1 = (F40149607D6C4F1D9A671DF771C24E44)
End 2 = N2 (2BD973F1096F44AE098F7DF14CB5110)
RelDef = DrawingItems
PIDRepresentation (B67765E3E26E46EDB8D0004AE1C2E70A)
Rel (DRC-4CECEAA6B99D4490A557D9B4134435-B67765E3E26E46EDB8D0004AE1C2E70A)
End 1 = 128-5001 (4CECEAA6B99D4490A557D9B4C4134435)
End 2 = (B67765E3E26E46EDB8D0004AE1C2E70A)
RelDef = DwgRepresentationComposition
Rel (DR-104-4-HP- (B67765E3E26E46EDB8D0004AE1C2E70A)
End 1 = (B67765E3E26E46EDB8D0004AE1C2E70A)
End 2 = N2 (0E53C538592848BEB6B7FB2E03C3465A)
RelDef = DrawingItems
PIDPipeline PH-104 (936EA3C4986F435BA261A38DE61CA91B)
Rel (PBS-936EA3C4986F435BA261A38DE61CA91B-N3-FMCUHPQRGJCA-RDAWMMF3)
End 1 = PH-104 (936EA3C4986F435BA261A38DE61CA91B)
End 2 = N3-FMCUHPQRGJCA-RDAWMMF3
RelDef = PBSItemCollection
PIDMechanicalEquipment RP-102 (929ACE2B2E76462997E62B199298C3E6)
Rel (PBS-929ACE2B2E76462997E62B199298C3E6-N3-FMCUHPQRGJCA-RDAWMMF3)
End 1 = RP-102 (929ACE2B2E76462997E62B199298C3E6)
End 2 = N3-FMCUHPQRGJCA-RDAWMMF3
RelDef = PBSItemCollection
PIDProcessVessel V-108 (E19B534810444F1CA62EF1FCC97CC57)
Rel (PBS-E19B534810444F1CA62EF1FCC97CC57-N3-FMCUHPQRGJCA-RDAWMMF3)
End 1 = V-108 (E19B534810444F1CA62EF1FCC97CC57)
End 2 = N3-FMCUHPQRGJCA-RDAWMMF3
RelDef = PBSItemCollection
*****
In Both But Different
*****
```

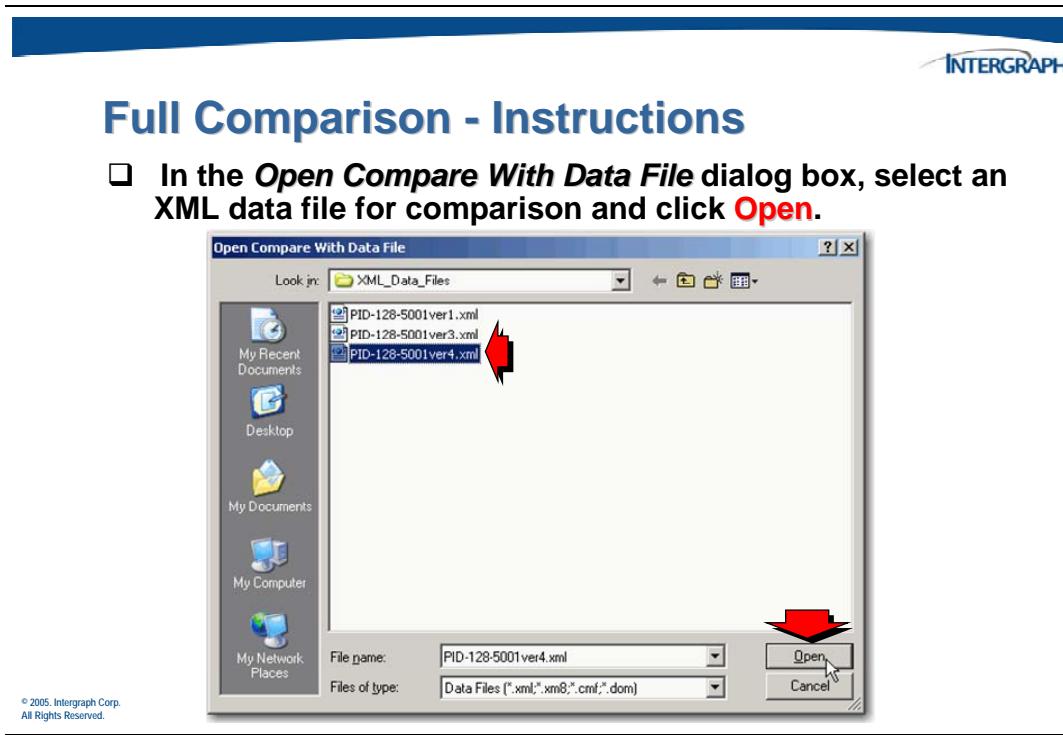
© 2005, Intergraph Corp.
All Rights Reserved.

After you review the comparison results, you can close Notepad by clicking **File > Exit**. The text file containing comparison results is stored in the location you specified earlier.

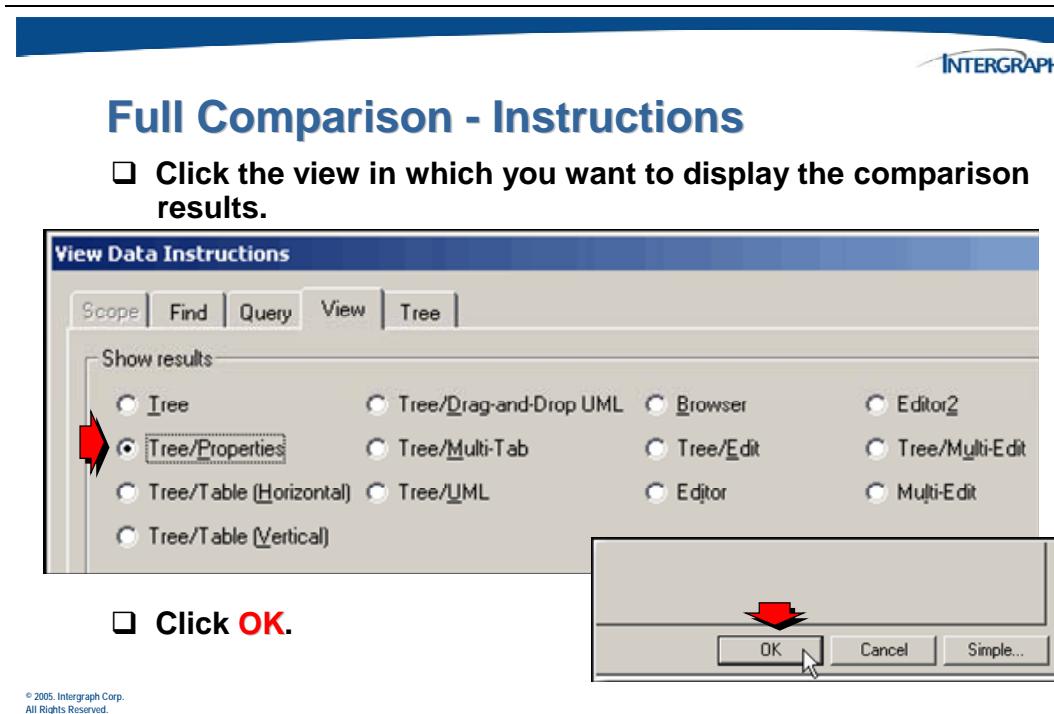
You can also choose to view on the comparison instructions generated when you compare two files. These comparison instructions are a set of instruction objects that define the changes necessary to go from the file you selected for comparison to the file you opened first in the Schema Editor.



An *Open Compare With Data File* dialog box appears.



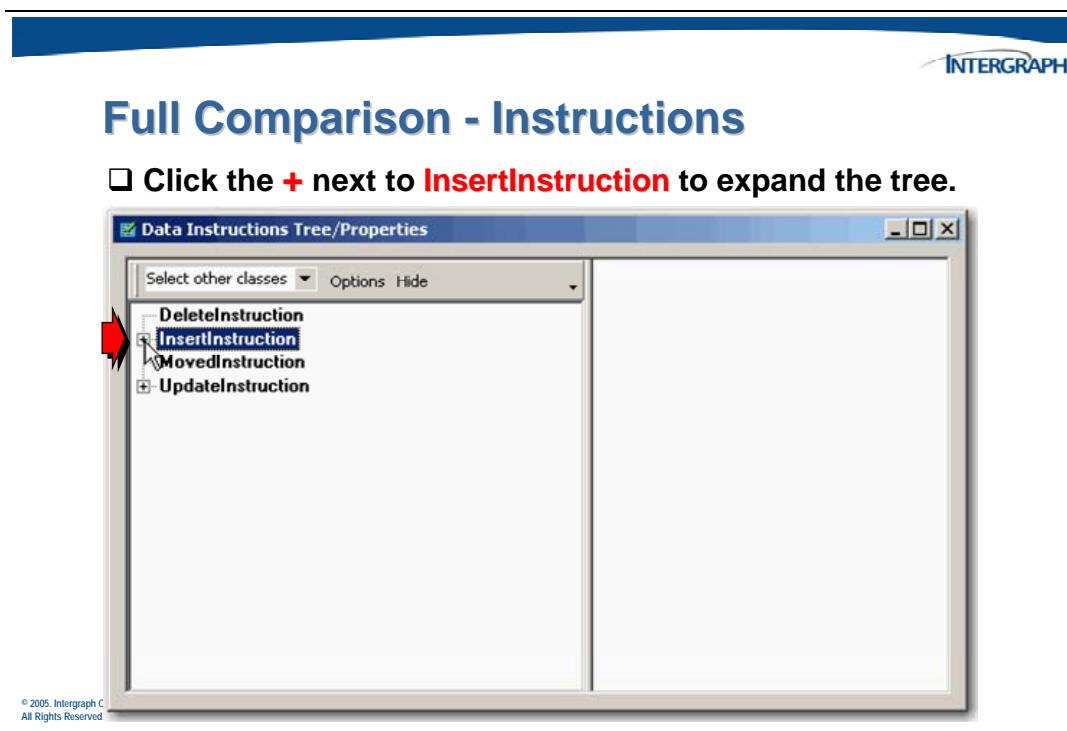
When you choose to view comparison instructions, the **View Data Instructions** dialog box appears to allow you to select the view in which you want to display the comparison instructions.



A *Data Instructions Tree/Properties* window appears with the comparison instructions. These comparison 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.

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.

Expand the tree to see comparison instructions of different types.



Expanding the tree shows *UpdateInstruction* objects.

The screenshot shows the 'Data Instructions Tree/Properties' dialog box. The left pane displays a tree view of instructions, with a red arrow pointing to the 'Insert_F-12830LP' node under the 'InsertInstruction' category. The right pane shows the properties for this object:

- Object**: Unique ID (23236723-9887-4DE4-8565-E21E1535D52C), Name (Insert_F-12830LP)
- Transaction**: Transformation required
- Reference Object**: Reference Object's Class (PIDInstrumentLoop), Reference Object's Unique ID (623FD2CA1C944579B49005148D9A07A3), Reference Object's Name (F-12830LP), Container Name (D:\XML_Data_Files\PID-128-5001ver4.xml)
- Property Values**: Property Values (Object%Name%F-12830LP%PBSItem%ConstructionStatus%NewConstruction%PBSItem%ConstructionStatus%@78299A84-9F3D-11D6-BD47-00104BCC2659\%NamedInstrumentLoop%LoopIdentifier%F-12830LP%NamedInstrumentLoop%LoopPrefix%Infl1%NamedInstrumentLoop%LoopSequenceNo%12830%NamedInstrumentLoop%LoopSuffix%LP%PBSItem%PlannedFacility%InstrumentLoop%PBSItemCollection%ExpandableThing%INoteCollection%NamedInstrumentLoop)
- Added Interfaces**: None listed
- Update Instruction**: Insert Instruction

© 2005, Intergraph Corp.
All Rights Reserved.

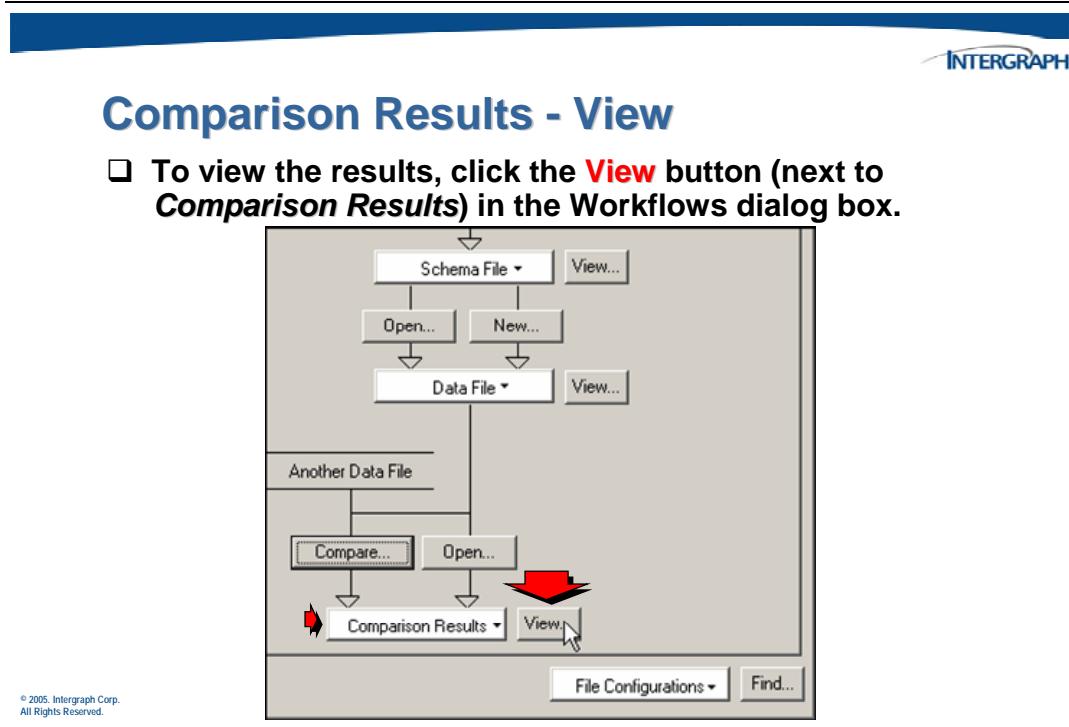
Expand the tree to view other instructions such as *UpdateInstructions*.

The screenshot shows the 'Data Instructions Tree/Properties' dialog box. The left pane displays a tree view of instructions, with a red arrow pointing to the 'Update_P-110-4"-1C0031-HS' node under the 'UpdateInstruction' category. The right pane shows the properties for this object:

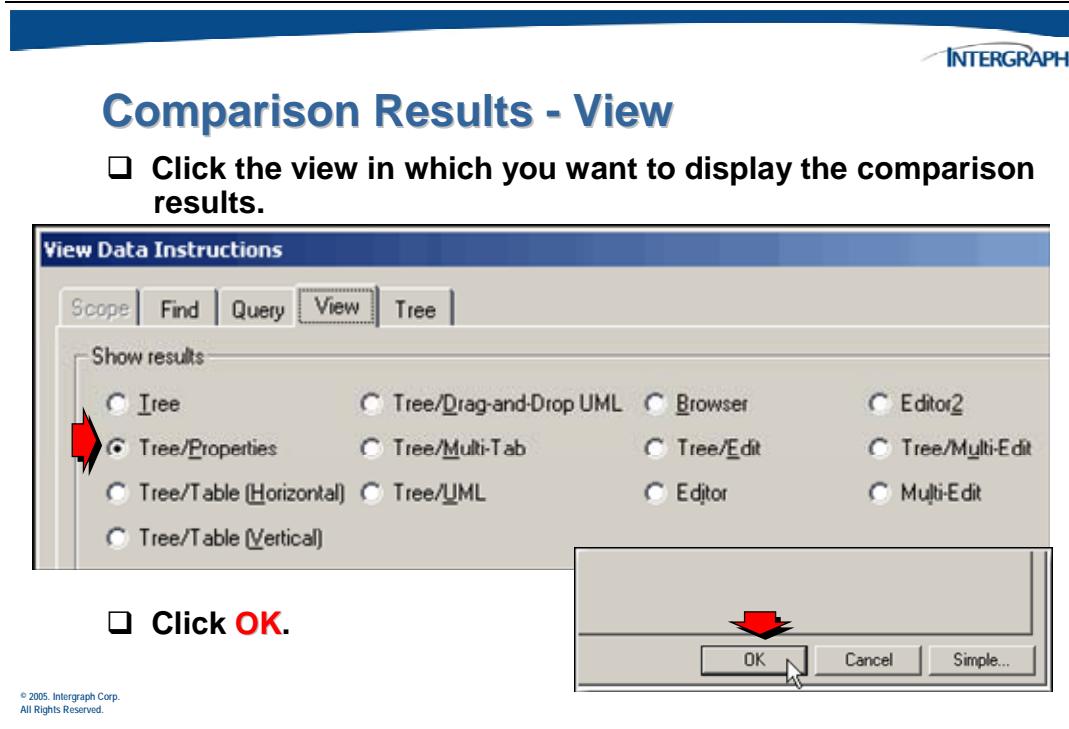
- Transaction**
- Transform**: Transformation required
- Reference Object**: Reference Object's Class (PIDPipingConnector), Reference Object's Unique ID (C410CD9D7EF442AF922C2C7C840268C8), Reference Object's Name (P-110-4"-1C0031-HS), Container Name (D:\XML_Data_Files\PID-128-5001ver4.xml)
- Property Values**: Property Values (Object%Name%P-110-4"-1C0031-HS%InsulatedItem%InsulTemp%43 F%InsulatedItem%AvgInsulDens%20 lbm/ft^3%InsulatedItem%InsulPurpose1%@{CB37D1DB-9D6B-11D6-BDA5-00104BCC2B69}\%InsulatedItem%InsulPurpose3%@{CB37D241-9D6B-11D6-BDA5-00104BCC2B69}\%InsulatedItem%InsulPurpose2%@{CB37D211-9D6B-11D6-BDA5-00104BCC2B69})
- Added Interfaces**: InsulatedItem
- Update Instruction**: None listed
- Removed Interfaces**: None listed

© 2005, Intergraph Corp.
All Rights Reserved.

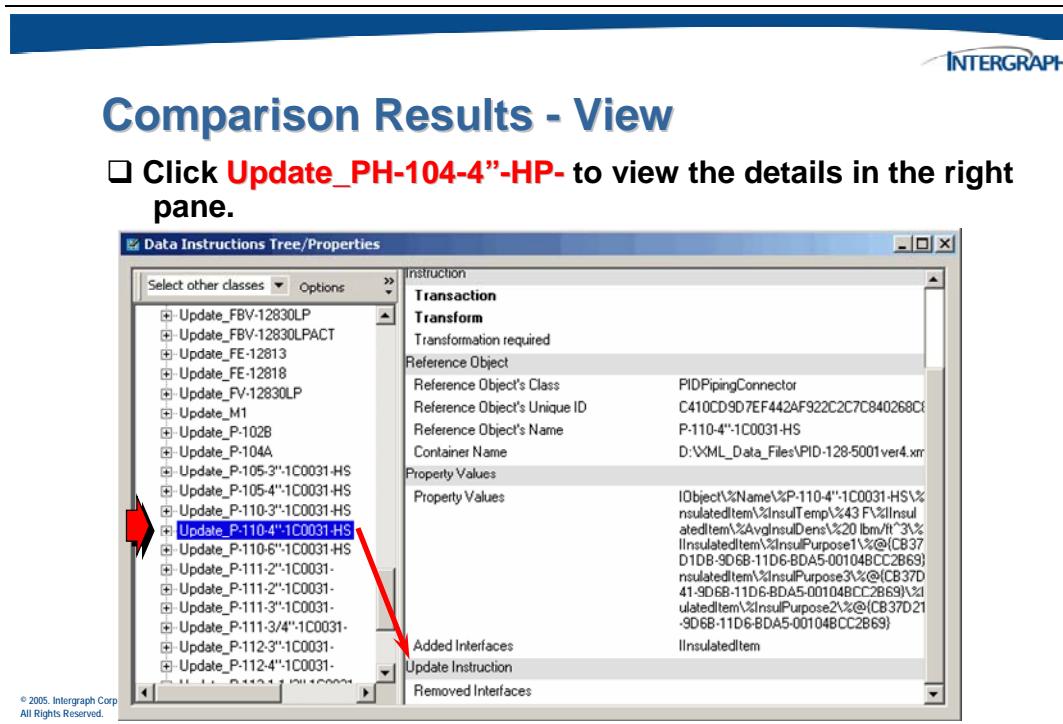
Another way to view the results is by selecting the *Comparison Results View* button. This button is only available after you compare two files as shown previously.



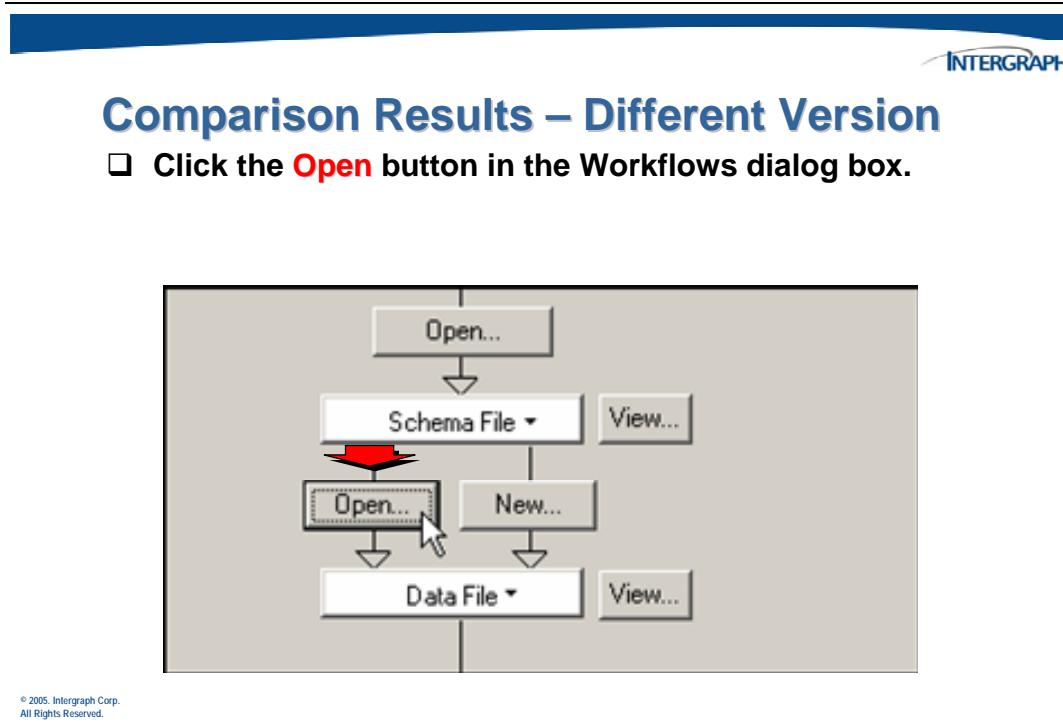
Again, a *View Data Instructions* dialog box appears.



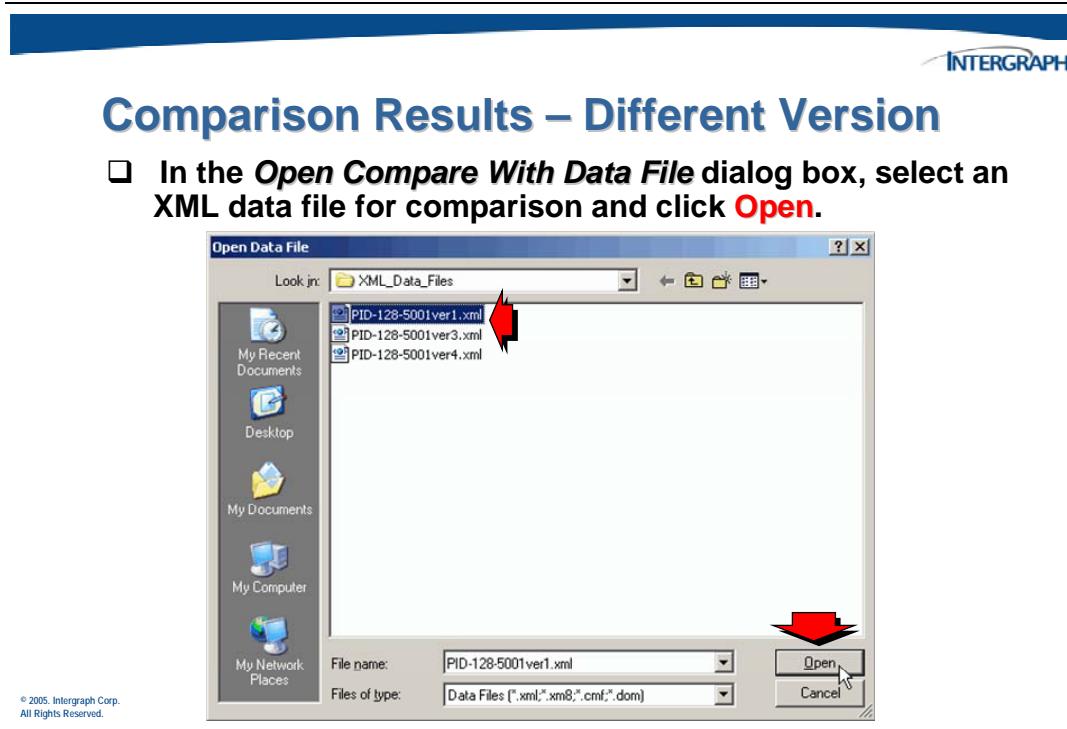
A *Data Instructions Tree/Properties* window appears.



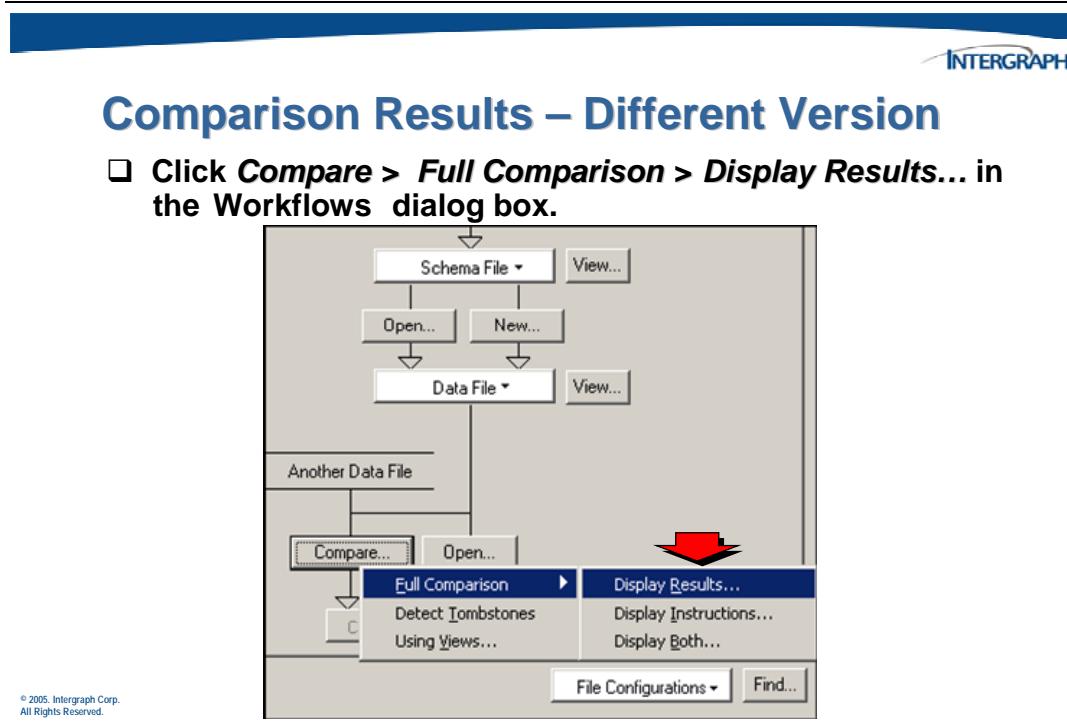
If different versions need to be compared, the first version data file must be opened.



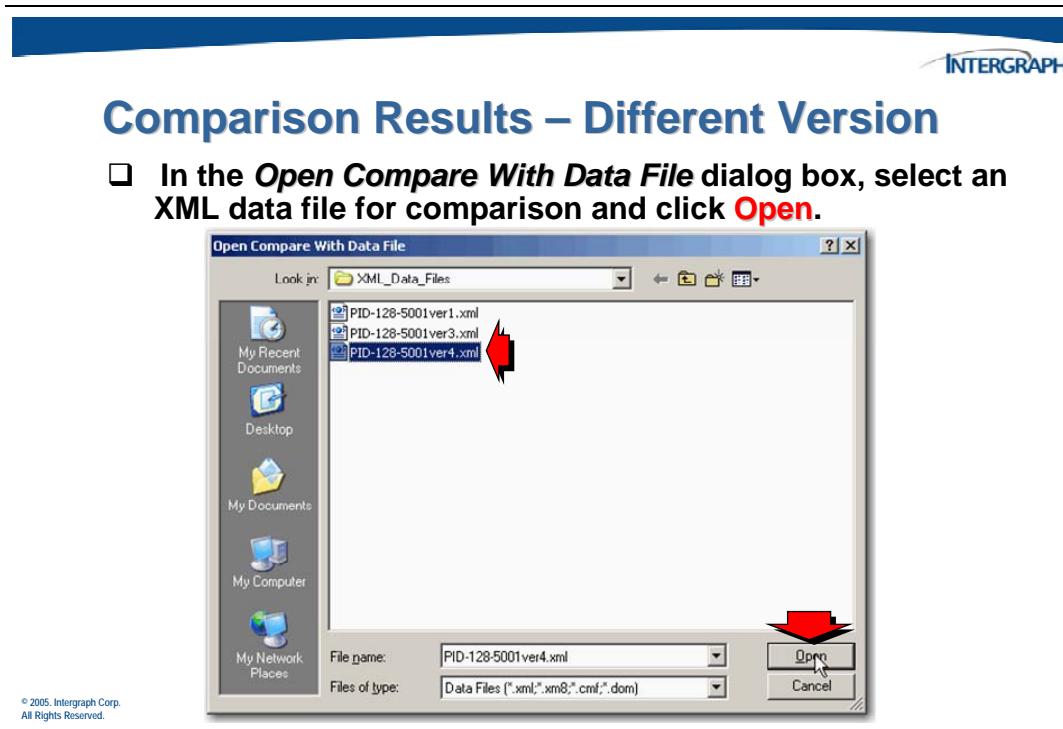
In this example, the versions to be compared will be version 1 and version 4. So, the XML version 1 file must first be opened.



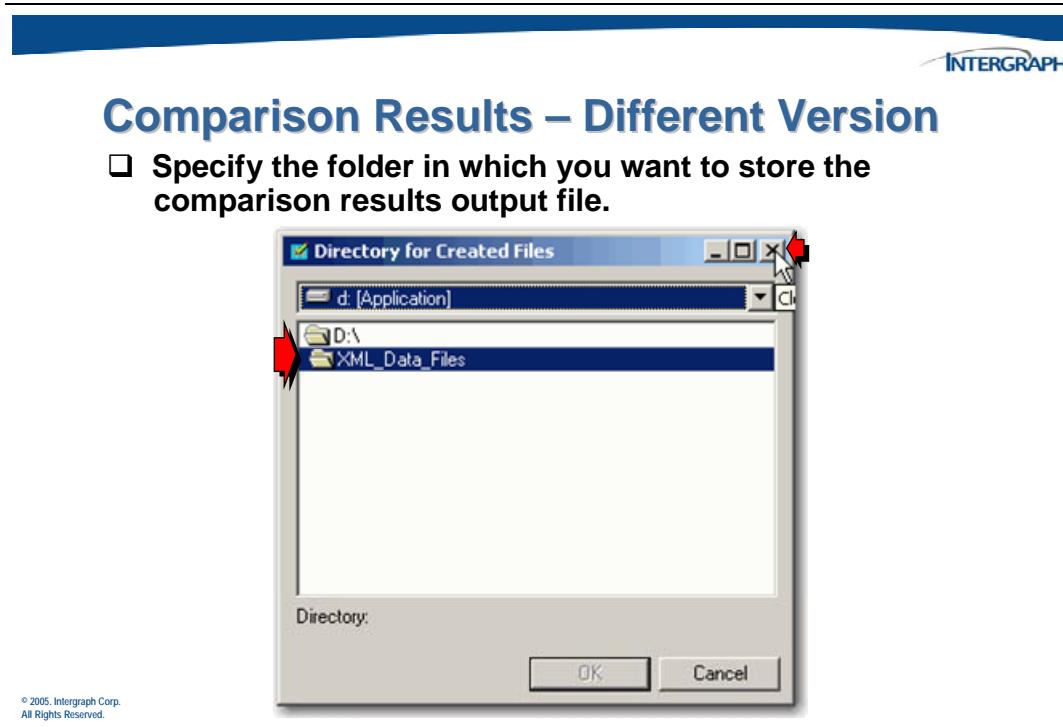
Next, select the type of comparison to be performed.



An *Open Compare With Data File* dialog box appears.



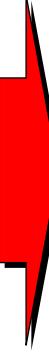
This will be the delta version to be compared.





Comparison Results – Different Version

A text output file containing the comparison results is created and displayed in Notepad.

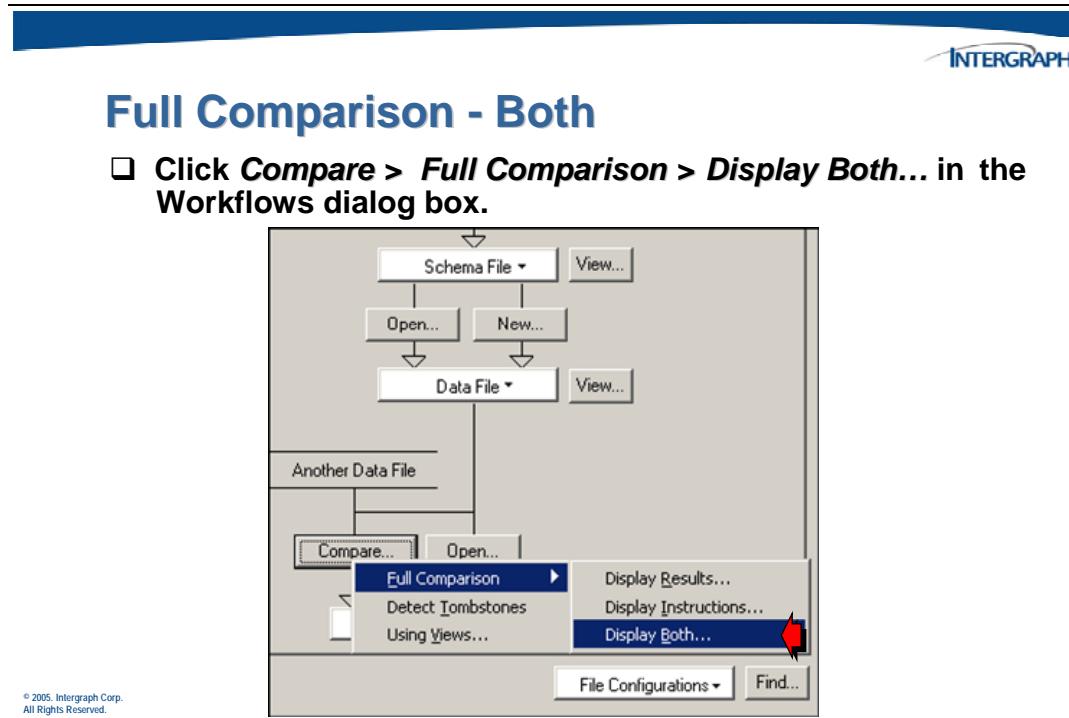


```
EF555.txt - Notepad
File Edit Format View Help
*****
only In D:\XML_Data_Files\PID-128-5001ver1.xml
*****
only In D:\XML_Data_Files\PID-128-5001ver4.xml
*****
Rel {ILA-5FAB54766CB5436C836CF25B31C393B7-8CF16D40035C41E3A075249271B281FC)
End 1 = ABV-12815LP (5FAB54766CB5436C836CF25B31C393B7)
End 2 = A-12815LP (8CF16D40035C41E3A075249271B281FC)
RelDef = InstrumentLoopAssembly
Rel {ILA-39CE344B304E4485875AF0BBE46A743B-623FD2CA1C944579B49005148D9A07A3)
End 1 = FBV-12830LP (39CE344B304E4485875AF0BBE46A743B)
End 2 = F-12830LP (623FD2CALC944579B49005148D9A07A3)
RelDef = InstrumentLoopAssembly
Rel {ILA-0CDD50DCDE4846BA9DC6B25D9BEF536-613F9A53AF9449BD9FF11C869A65B81A)
End 1 = ABV-12832LP (0CDD50DCDE4846BA9DC6B25D9BEF536)
End 2 = A-12832LP (613F9A53AF9449BD9FF11C869A65B81A)
RelDef = InstrumentLoopAssembly
Rel {ILA-6DA2C8D058274CC2B30714FC880A64F1-CD80626FCF6C46C080CC290C0ACD0E5A)
End 1 = ABV-12831LP (6DA2C8D058274CC2B30714FC880A64F1)
End 2 = A-12831LP (CD80626FCF6C46C080CC290C0ACD0E5A)
RelDef = InstrumentLoopAssembly
Rel {ILA-AFDBSEE38C1F4E3990715F165858FCB3-CD80626FCF6C46C080CC290C0ACD0E5A)
End 1 = ABV-12831LPACT (AFDBSEE38C1F4E3990715F165858FCB3)
End 2 = A-12831LPACT (CD80626FCF6C46C080CC290C0ACD0E5A)
RelDef = InstrumentLoopAssembly
Rel {ILA-3240A32160BA4081B6D44336040B7B70-613F9A53AF9449BD9FF11C869A65B81A)
End 1 = ABV-12832LPACT (3240A32160BA4081B6D44336040B7B70)
End 2 = A-12832LPACT (613F9A53AF9449BD9FF11C869A65B81A)
```

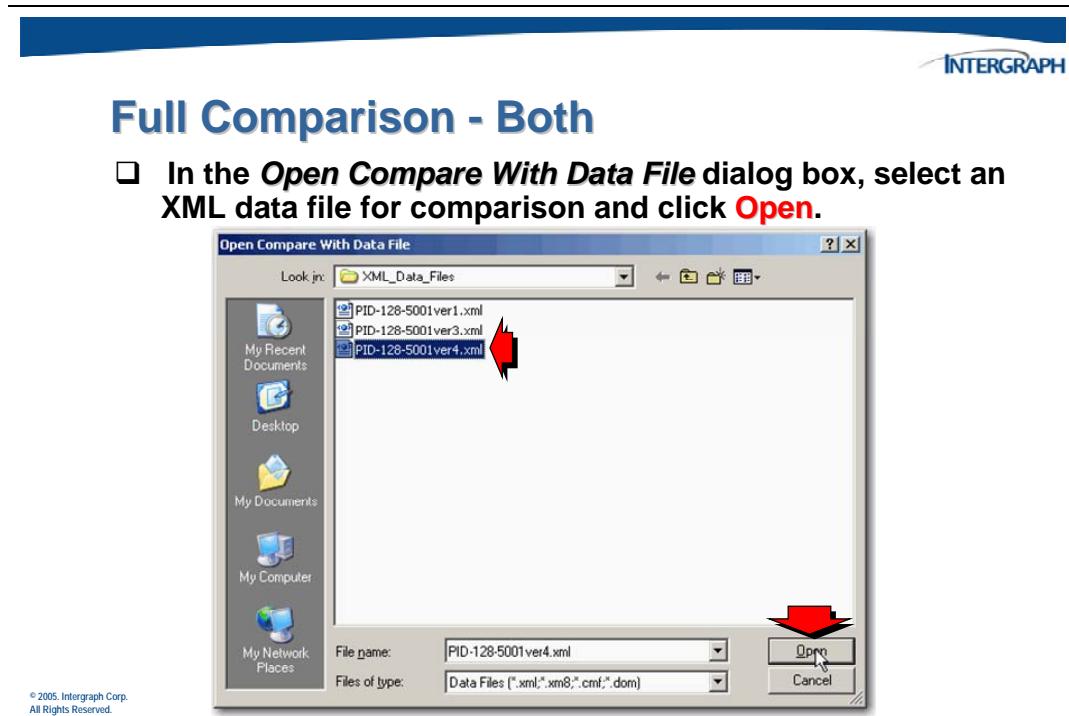
© 2005. Intergraph Corp.
All Rights Reserved.

Use the scroll bar to see the remainder of the comparison results.

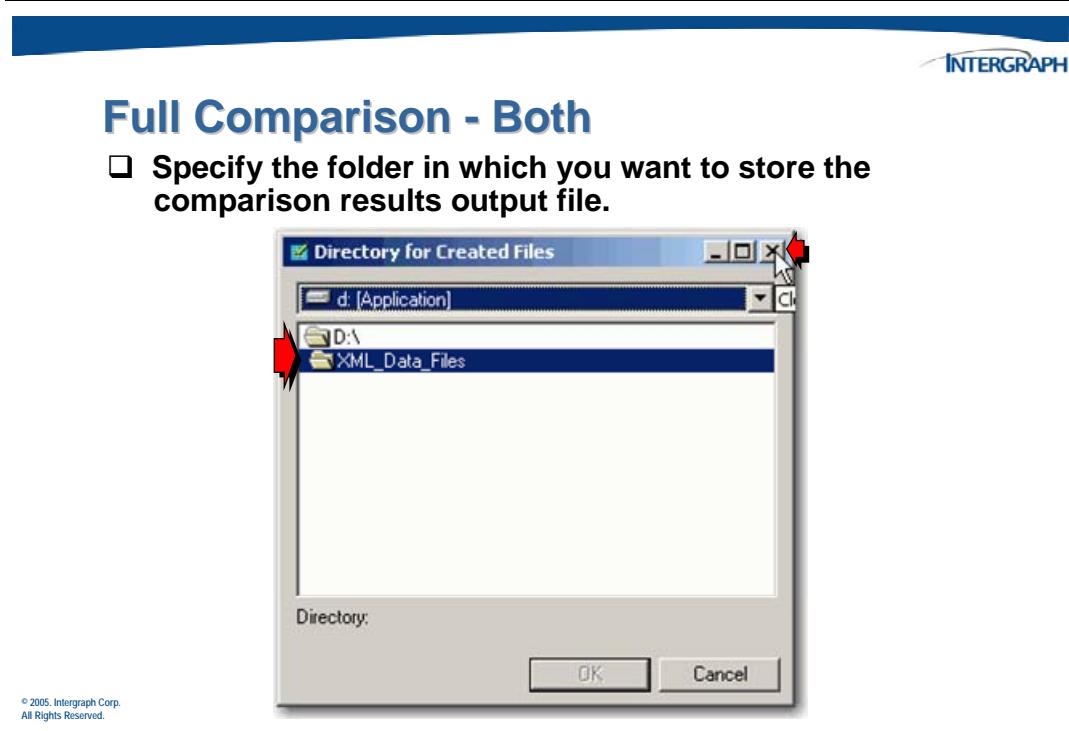
When comparing data files, the option to *Display Results* or *Display Instructions* have been demonstrated. In the following example, comparison of two file xml's are shown with the option to *Display Both* (results and instructions).



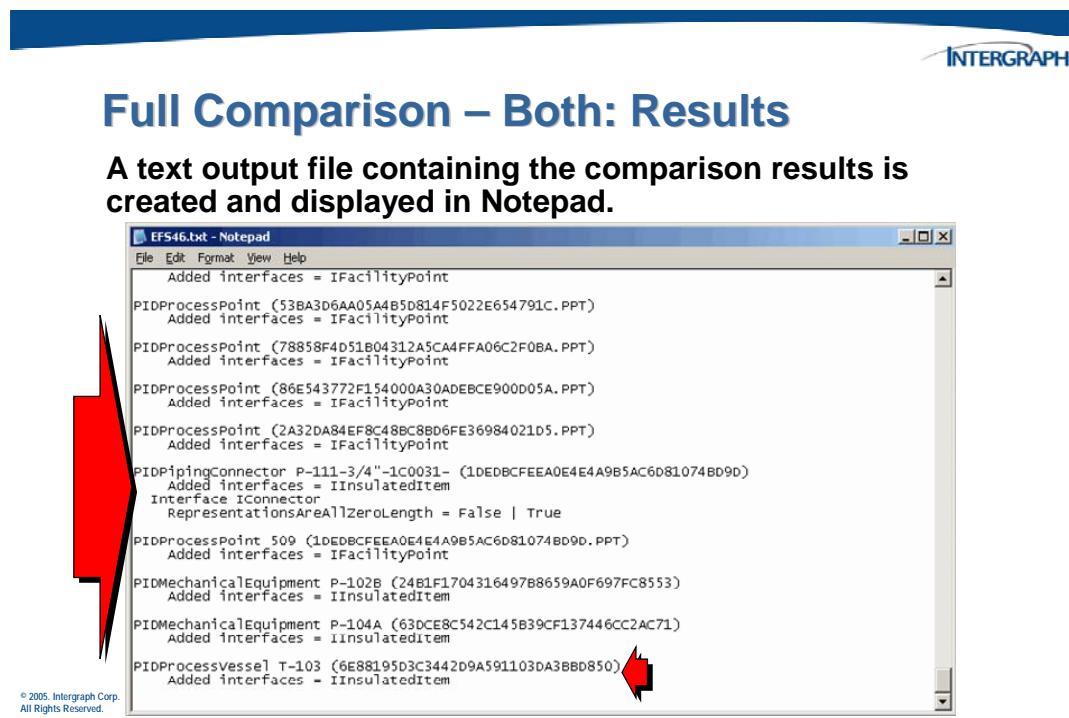
Selecting the *Display Both* option will show results and instructions. Choose the XML file version to be used to compare against.



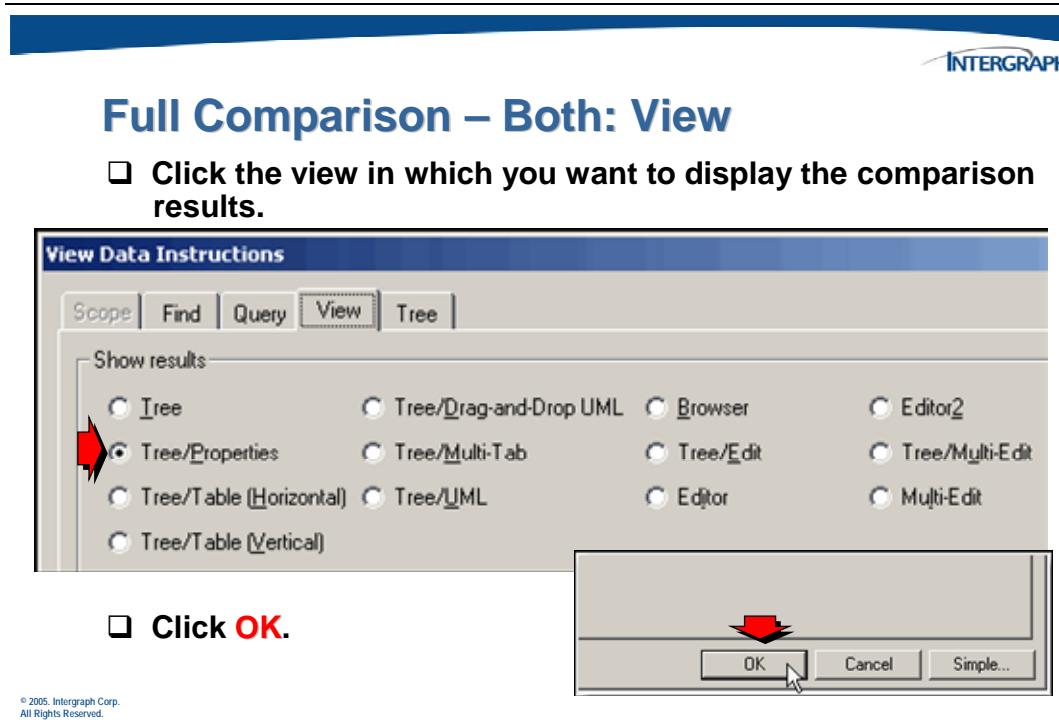
A *Directory for Created Files* dialog displays.



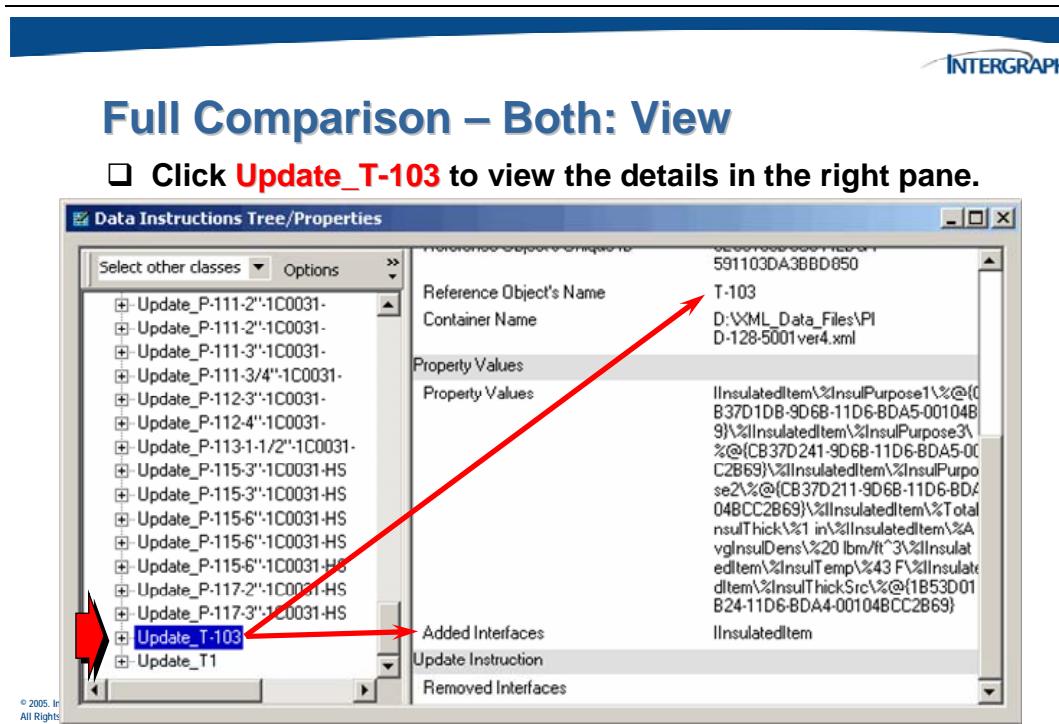
Comparison results appear first in Notepad for viewing.



Now select the view format to for the comparison results.



A *Data Instructions Tree/Properties* window displays.



6.4.2 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.

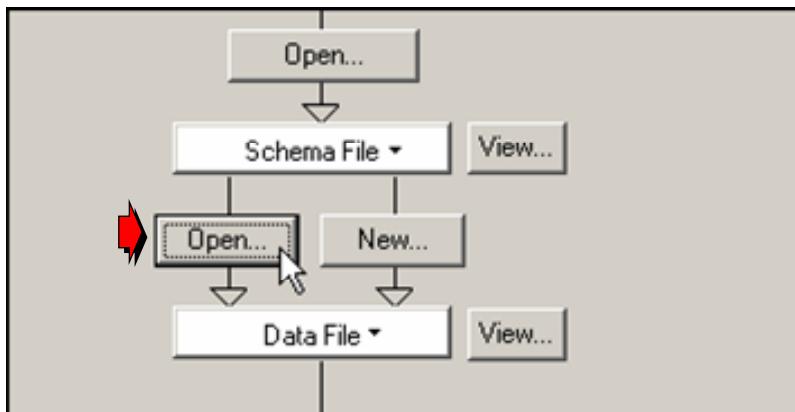
Because tombstone detection does not involve a full comparison of two files, it typically runs faster than a full comparison.

First, open the data file to which you want to compare another file.



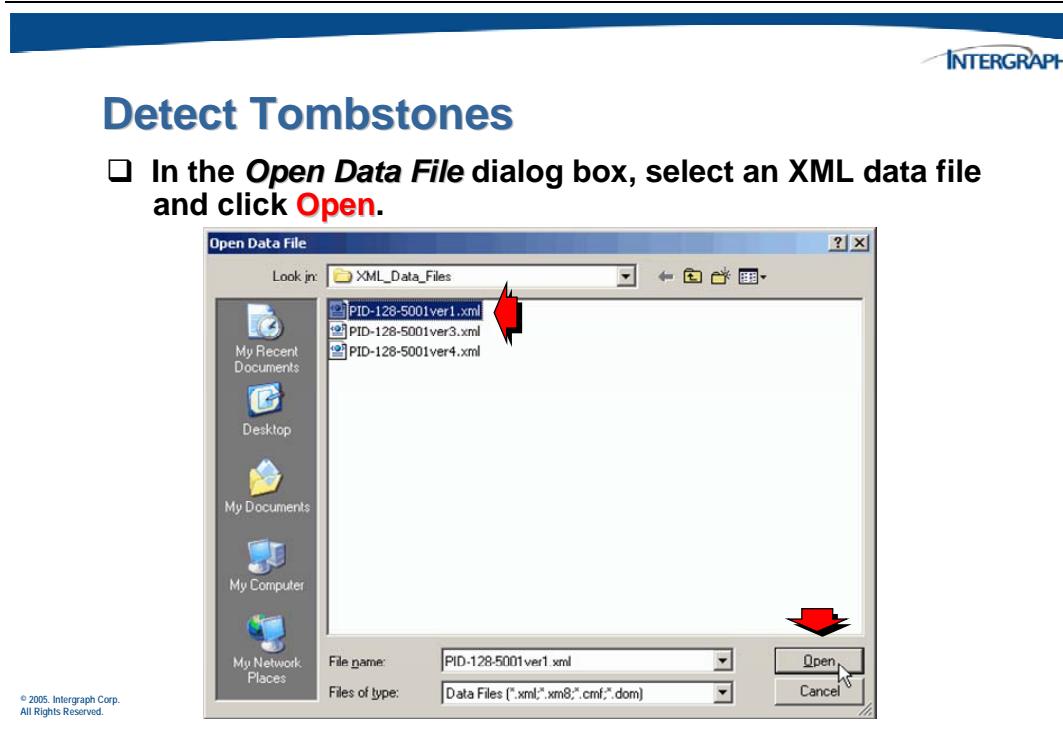
Detect Tombstones

- In the **Workflows** dialog box, click the **Open** button to open an **XML** data file.

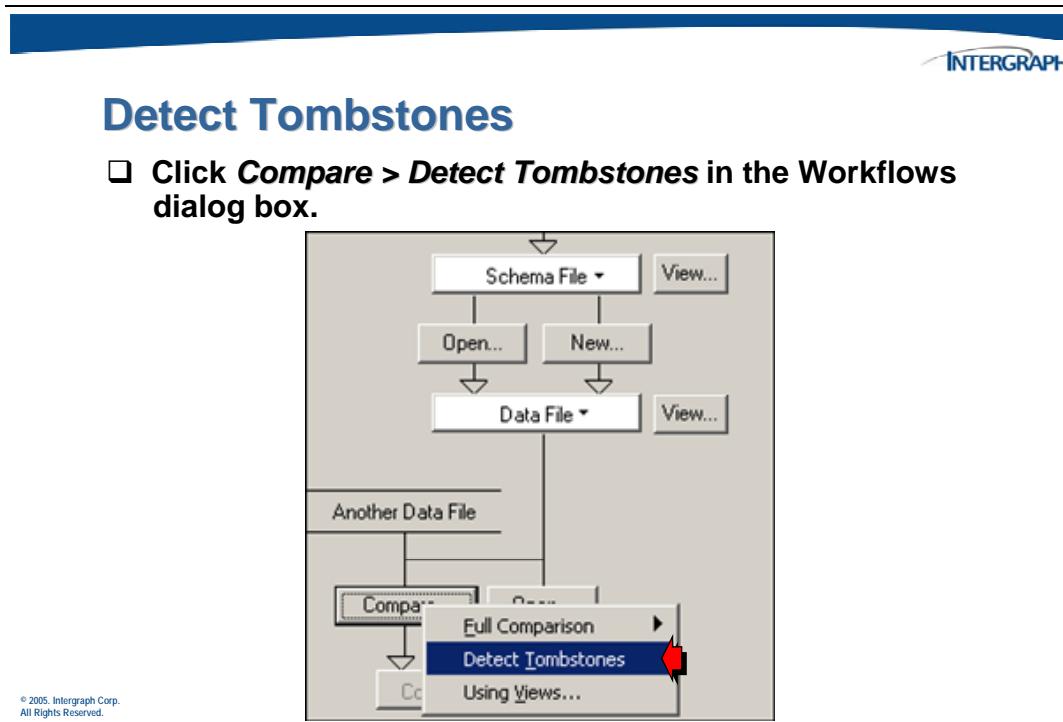


© 2005, Intergraph Corp.
All Rights Reserved.

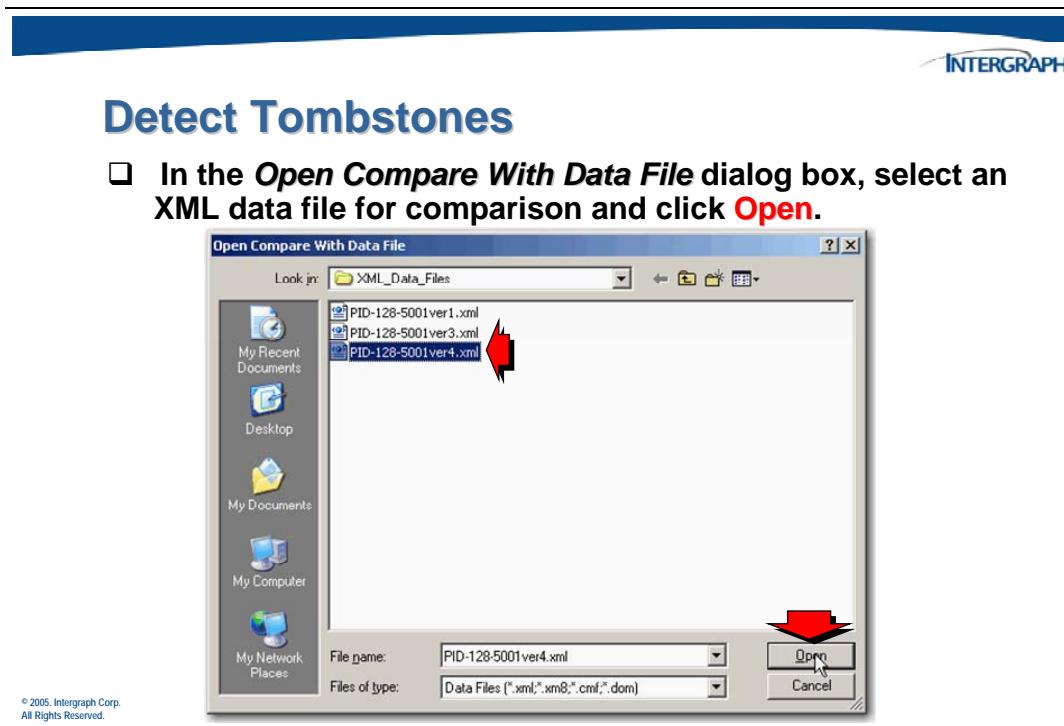
An *Open Data File* dialog box will display.



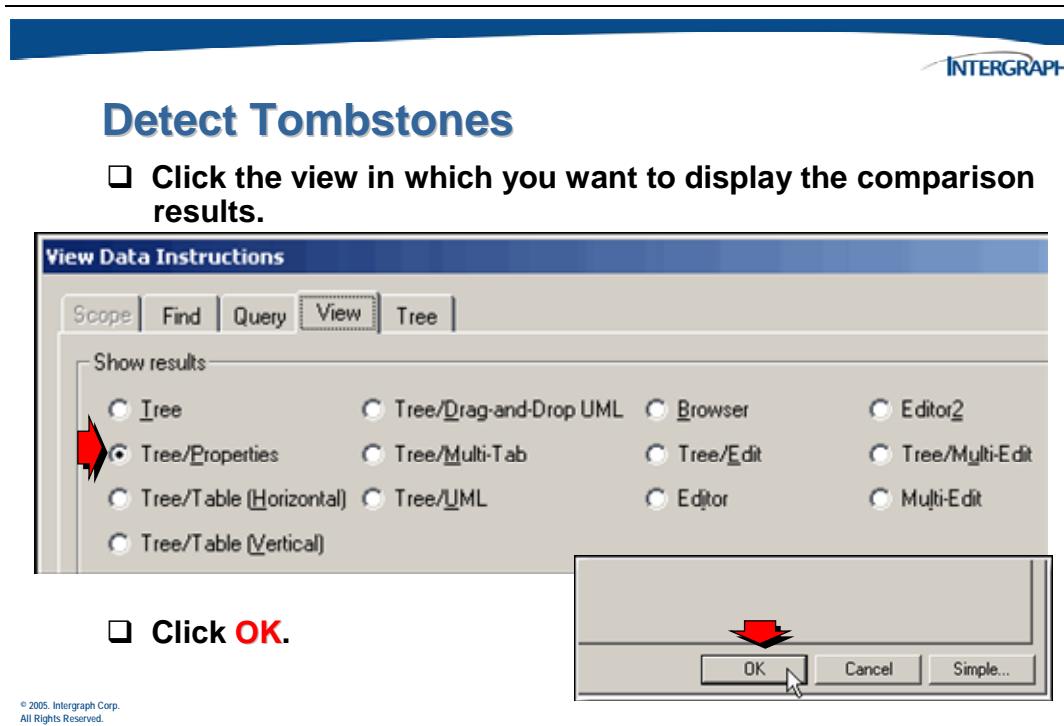
Select the **Compare** button from the *Workflows* dialog.



Choose the XML file version to be used for comparison.



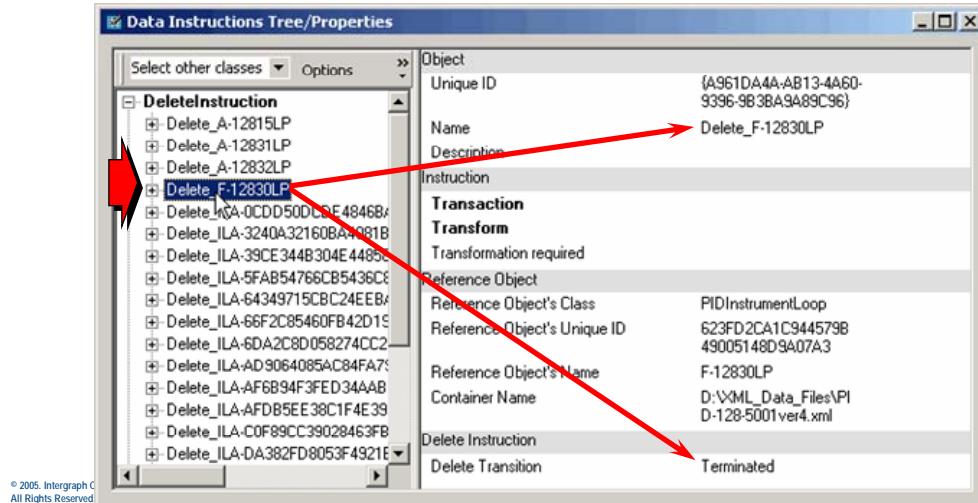
Now select the view format to for the comparison results.



A *Data Instructions Tree/Properties* window displays. The Schema Editor identifies items as tombstones when they do not exist in one file you opened in the Schema Editor but do exist in the other file you selected for comparison.

Detect Tombstones

- Click **Delete_F-12830LP** to view the details in the right pane.



6.5 Activity – Viewing and Finding Data

The goal of this activity is to familiarize you with using the Schema Editor to view data files. You will start the Schema Editor and make sure the EF Schema.xml schema is open. You will then use different types of views to open and compare XML files.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant Foundation > SmartPlant Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
3. Open the custom SmartPlant schema configuration file, **adc.cfg**.
 - Click the **File Configurations** button and select **Open Configuration...** from the *Schema Editor* Workflows dialog
 - Select the **adc.cfg** file and click **Open** (Path is C:\Program Files\Common Files\Intergraph\EF Schema\03.08)

Viewing Data Files

4. Open and view objects from a data file using the **Tree/UML** view type.
 - In the *Workflows* dialog box, select the **Data** tab, then click the **Open** button, which is just above the *Data File* button.
 - When the *Open Data File* dialog box appears, go to **D:\XML_Data_Files**, click **PID-128-5001ver1.xml** file and click **Open**.
 - In the *Workflows* dialog box, click the **View** button beside the *Data File* button.
 - When the *View Data* dialog box appears, click the **Tree/UML** option and click **OK**.
 - Expand the **PIDInlineInstrument** object.
 - Click the instrument object **ABV-12832**.
 - What happens to the right pane in the window?

 - Click the piping port object **1**.
 - What happens to the right pane in the window?

-
- Expand the **ABV-12832** object.
 - Expand the **Piping Ports** object.
 - Click the port object **2**.
 - Close the view window (click the **X** icon).
5. Open and view objects from a data file using the **Tree/Table (vert)** view type.
- In the *Workflows* dialog box, click the **View** button beside the *Data File* button.
 - When the *View Data* dialog box appears, click the **Tree/Table (Verticle)** option and click **OK**.
 - Expand the **PIDNozzle** object.
 - Click the **A5** nozzle object.
 - What happens to the right pane in the window?

 What is the value for **Piping Material**? _____
 - Close the view window.
6. Open and view objects from a data file using the **Tree/Properties** view type.
- In the *Workflows* dialog box, click the **View** button beside the *Data File* button.
 - When the *View Data* dialog box appears, click the **Tree/Properties** option and click **OK**.
 - Expand the **PIDProcessVessel** object.
 - Click the instrument object **T-103**.
 - What happens to the right pane in the window?

 Close the view window.

Finding Data Objects

7. Use the **Find** functionality to locate and view schema file components using the **Tree/Properties** view type.

- In the *Workflows* dialog box, click the **View** button beside the *Data File* button
 - When the *View Data* dialog box appears, click the **Find** tab.
 - Beside the *Only Classes* box, click the select (...) button.
 - In the *Select Classes to Display* dialog box, scroll down the list, choose **PIDPipingComponent**, **PIDPipingConnector** and **PIDProcessVessel**, and click **OK**.
 - In the *Name* box, type ***10*** and click **OK**.
 - Expand the **PIDPipingConnector** object.
 - What do you see in the view?

 - Repeat this for **PIDProcessVessel**.
 - Close the view window.
8. Using the **Find** functionality, set the scope to locate and view data file objects using the **Tree/Properties** view type.
- In the *Workflows* dialog box, click the **Find** button.
 - When the *Find* dialog box appears, select the Scope tab and then the **Data** check box and click **OK**.
 - In the *Name* box, type ***10*** and click **OK**.
 - Click on the displayed objects.
 - What do you see in the view?

 - Close the view window.

Comparing Data Files

9. Use the **Compare** functionality to view comparison results for two XML files.
- In the *Workflows* dialog box, click the **Compare** button, and then click **Full Comparison > Display Results...**
 - When the *Open Compare With Data File* dialog box appears, select the **PID-128-5001ver3.xml** file and click **Open**.
 - Browse to the following the folder for the comparison text file that the software creates during a compare: **D:\XML_Data_Files**.

- Scroll through and review the resulting text output file.
 - Close the Notepad window.
10. Use the **Compare** functionality to display comparison instructions.
- In the *Workflows* dialog, click the **Compare** button and then click **Full Comparison > Display Instructions**.
 - When the *Open Compare With Data File* dialog box appears, select the **PID-128-5001ver4.xml** file and click **Open**.
 - When the *View Data Instructions* dialog box appears, click the **Tree/Properties** option and click **OK**.
 - Expand the **InsertInstruction** object.
 - Click the one of the **Insert** instruction objects.
 - What do you see in the view?

 - Expand the **UpdateInstruction** object.
 - Click the one of the **Update** instruction objects.
 - What do you see in the view?

 - Close the view window.
11. Use an optional method to view comparison file results.
- In the *Workflows* dialog box, click the **View** button beside the **Comparison Results** button.
 - When the *View Data Instructions* dialog box appears, click the **Tree/Properties** option and click **OK**.
 - Expand the **UpdateInstruction** object.
 - Click one of the **Update** instruction objects.
 - Close the view window.
12. Open both the **PID-128-5001ver1.xml** and the **PID-128-5001ver4.xml** files and display both the results and instructions.

13. Open and view tombstones from a data file using the **Tree/Properties** view type.
 - In the *Workflows* dialog box, click the **Open** button, which is just above the *Data File* button.
 - When the *Open Data File* dialog box appears, click **PID-128-5001ver1.xml** file and click **Open**.
 - In the *Workflows* dialog, click the **Compare** button and then click **Detect Tombstones**.
 - When the *Open Compare With Data File* dialog box appears, select the **PID-128-5001ver4.xml** file and click **Open**.
 - When the *View Data Instructions* dialog box appears, click the **Tree/Properties** option and click **OK**.
 - Expand the **DeleteInstruction** object.
 - Click one of the **Delete** instruction objects.
 - What do you see in the view?

- Close the view window.

14. **Exit** the Schema Editor and but don't save your work.

15. Once you have exited the Schema Editor, you may take a short break until the other students have finished this activity.

Summary:

In this activity, you accessed the Schema Editor and familiarized yourself with viewing and finding objects in an XML data file.

7

C H A P T E R

Introduction to Schema Mapping

7. Introduction to Schema Mapping

Schema Mapping provides a mechanism for the software to convert authoring tool data, described by the *tool map schema*, to SmartPlant data, described by the *SmartPlant schema*, and SmartPlant data back to tool data. Each tool that integrates with SmartPlant supplies the mapping between that tool and the SmartPlant schema for both publish and retrieve operations. The mapping is stored in the authoring tool map schema.

Mapping allows authoring tools to externalize the definition of mapping to isolate authoring tools from changes in the SmartPlant schema. This mapping defines correlations to the SmartPlant schema strictly using UIDs, which should not change.



Objectives of SmartPlant Mapping

The Schema Editor allows you to perform mapping between *Tool Map Schemas* and the *SmartPlant Schema*.

The SmartPlant schema supports one to many and many to many mapping.

© 2005. Intergraph Corp.
All Rights Reserved.

In the Schema Editor, you can modify how authoring tool data maps into and out of the SmartPlant schema by modifying the authoring tool mapping. A common modification to existing tool mapping is the extension of *tool enumerated lists* that are then mapped to enumerated lists in the SmartPlant schema.

To help you understand how mapping works, this chapter will cover the following procedures:

- ❑ Extending an existing enumerated list in an authoring tool (SmartPlant P&ID)
- ❑ An introduction to the *Schema Editor Mapping Environment*
- ❑ Synchronizing the enumerated list in the authoring tool to the tool map schema
- ❑ Extending an existing enumerated list in the SmartPlant schema

- Mapping the property to and from the SmartPlant schema for publish and retrieve with the tool map schema

In previous releases, mapping from the “tool schema” to the SmartPlant schema was done based on the underlying SmartPlant schema objects (class, interface, property and edge definitions). This required the person doing the mapping to be intimately familiar with that part of the overall SmartPlant schema. Feedback from SmartPlant users indicated that this was a burdensome requirement, in that it required significant effort for an individual to attain the appropriate level of knowledge of the SmartPlant schema.



Objectives of SmartPlant Mapping

Simplify the process required to add properties and enumerations across the enterprise:

- Centralized
- Dynamic
- Less steps

Provide a centralized user interface:

- Graphical
- Easier to use

Enable a more gradual learning curve:

- Expose only what is required for mapping

To reduce the effort and knowledge required of the tool integrator defining the mapping, a simpler, view-based approach is now available. This approach will present the user with a “business object” view of the schema that will be much easier to understand and utilize.



What SmartPlant Mapping Does

Reduces the number of steps required to perform mapping:

- Allows the tool’s meta-data to be directly changed from the mapping environment.**

Uses ViewDefs and existing mapping relationships to suggest possible SmartPlant Schema changes and tool meta-data changes.

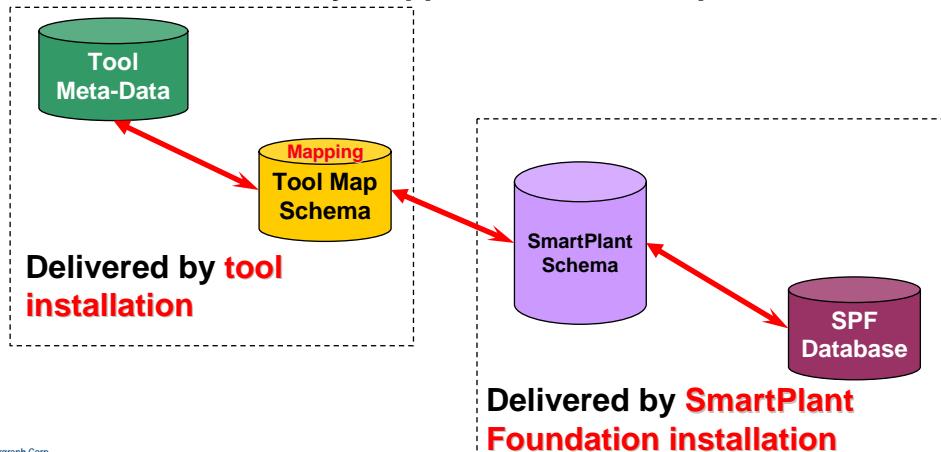
To reduce the effort and knowledge required of the tool integrator defining the mapping, a simpler, view-based approach is now available. This approach will present the user with a “business object” view of the schema that will be much easier to understand and utilize.

To simplify the mapping and to enable the definition of “transformations” associated with the mapping, the GUI was enhanced to use a more graphical (versus tabular) approach to the mapping. The overall complexity of the previous user interface was replaced by a more intuitive, graphical approach.



Mapping with the Schema Editor

Any changes can be added to the tool meta-data (authoring tool data base), then the tool map schemas and SmartPlant schemas, and finally mapped in the **tool** map schema.



© 2005, Intergraph Corp.
All Rights Reserved.



Mapping with the Schema Editor

Every class in the tool schema that is to be published to TEF maps to a class or interface definition in the Framework schema.

Mapping to primary interface definitions is recommended.

Properties that are of interest to upstream applications are also mapped to the appropriate properties in the Framework schema.

© 2005, Intergraph Corp.
All Rights Reserved.



Mapping with the Schema Editor

In each tool schema, the **MapClassToClass** and **ClassToMapClass** relationship definitions identify the mapping from the tool's class definition to a corresponding object in the Framework schema and the mapping from the Framework schema to the tool schema.

The mapping in one direction may not necessarily match the mapping in the other direction.

© 2005, Intergraph Corp.
All Rights Reserved.



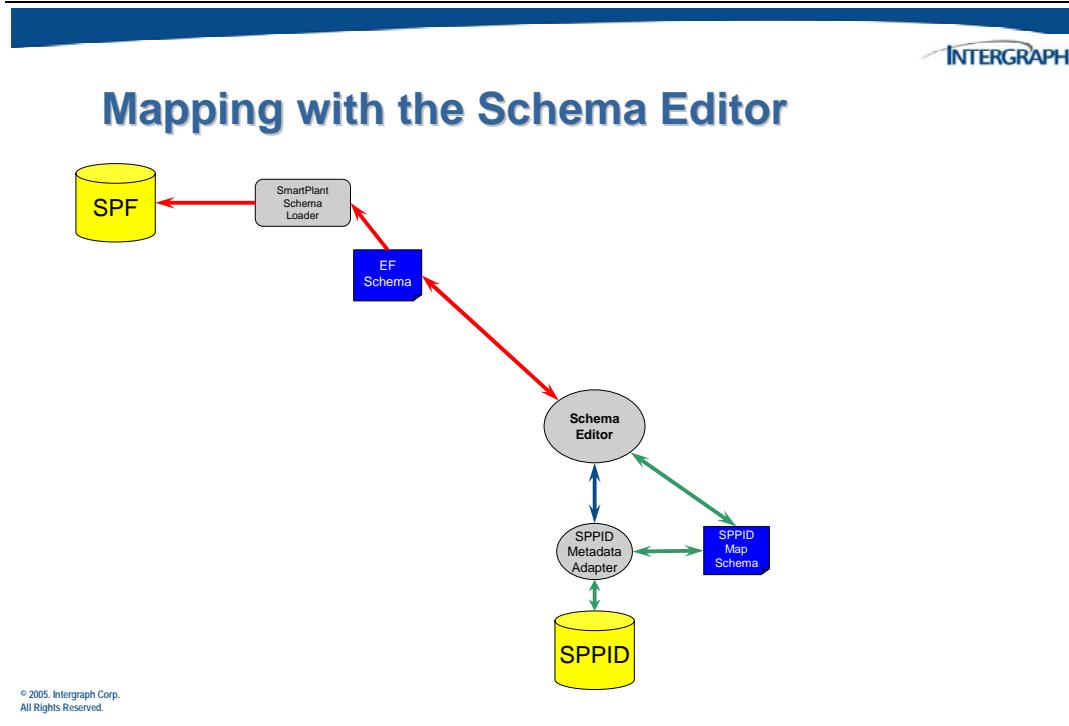
Mapping with the Schema Editor

The mapping between property definitions in the tool schema and their counterparts in the Framework schema is done using the **MapPropertyToProperty** and **PropertyToMapProperty** relationship definitions.

If a property definition is enumerated or has units of measure, it has a relationship of type **MapPropertyMapEnumList** that relates it to the appropriate enumerated list or unit of measure list.

© 2005, Intergraph Corp.
All Rights Reserved.

Data originating from an authoring tool's database (metadata) can now be manipulated from the Schema Editor using a **metadata adapter**, which will be delivered by each individual authoring tool.

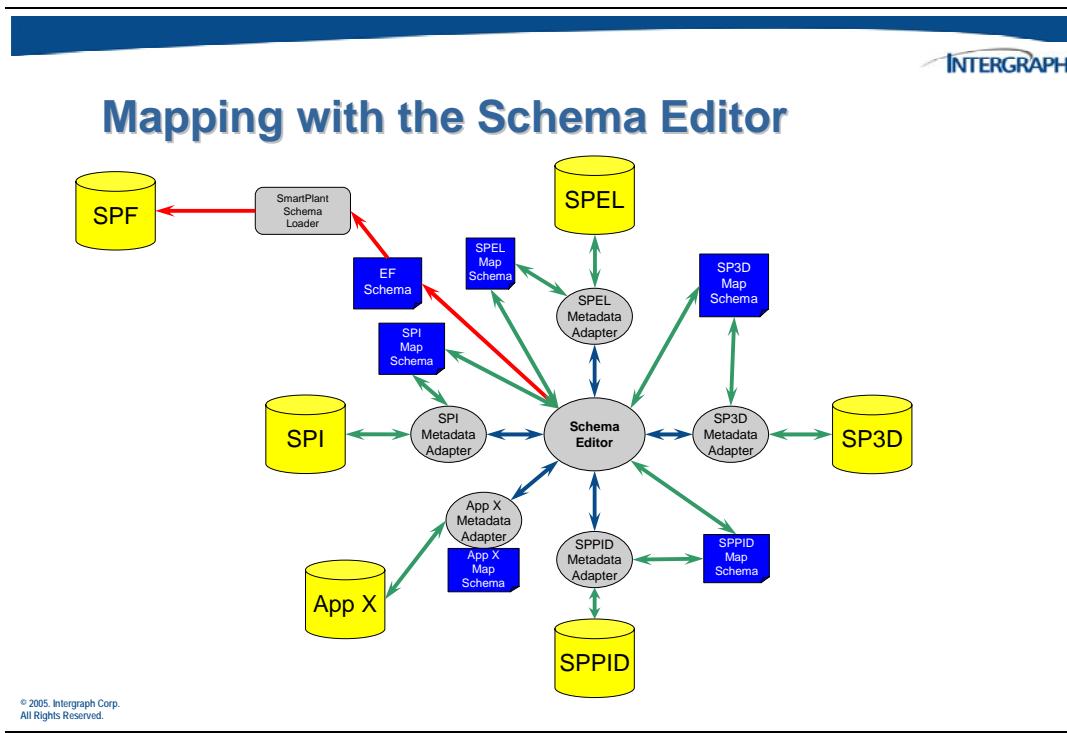


The metadata adapter can be used to read the data definitions in the tool meta-schema. If new definitions are found in the authoring tool database, the metadata adapter can extract these new definitions and automatically add them to the tool map schema. This is called synchronizing the tool meta-schema and the tool map schema and occurs when the Schema Editor connects to SmartPlant and the tool map schema is opened.

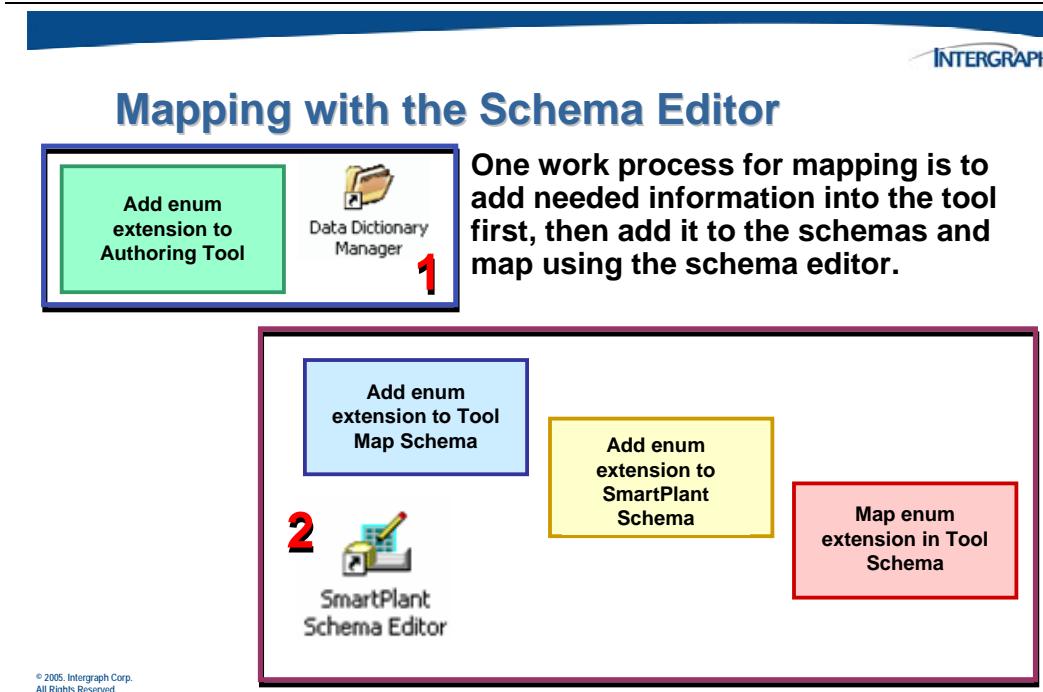
After the tool meta-schema and tool map schema have been synchronized, new definitions can be added to the SmartPlant schema (EFSchema) and then mapped via the Schema Editor.

One of the last steps will be to upload the changed schema files to the SPF server so that they can be loaded into the SPF Admin database.

Every tool will utilize a metadata adapter as illustrated by the figure below.



Mapping data from the authoring tool to SmartPlant and back requires one of the following work processes. In some cases, extensions to the authoring tool already exist. To complete this process, the same extensions have to be integrated through out the schemas.



Another possibility is to use the schema editor and the metadata adapter to add any new needed definitions to the schemas and the tool meta-schema from a single user interface, the *Schema Editor Map Environment*. This process will be discussed in more detail in the next chapter.

The diagram is titled "Mapping with the Schema Editor". It features a central icon of a pencil and paper labeled "SmartPlant Schema Editor". Surrounding this icon are four colored boxes:

- A green box on the top left: "Add enum extension to Authoring Tool".
- A blue box above the central icon: "Add enum extension to Tool Map Schema".
- An orange box to the right of the central icon: "Add enum extension to SmartPlant Schema".
- A red box below the central icon: "Map enum extension in Tool Schema".

To the right of the central icon, there is a large block of text:

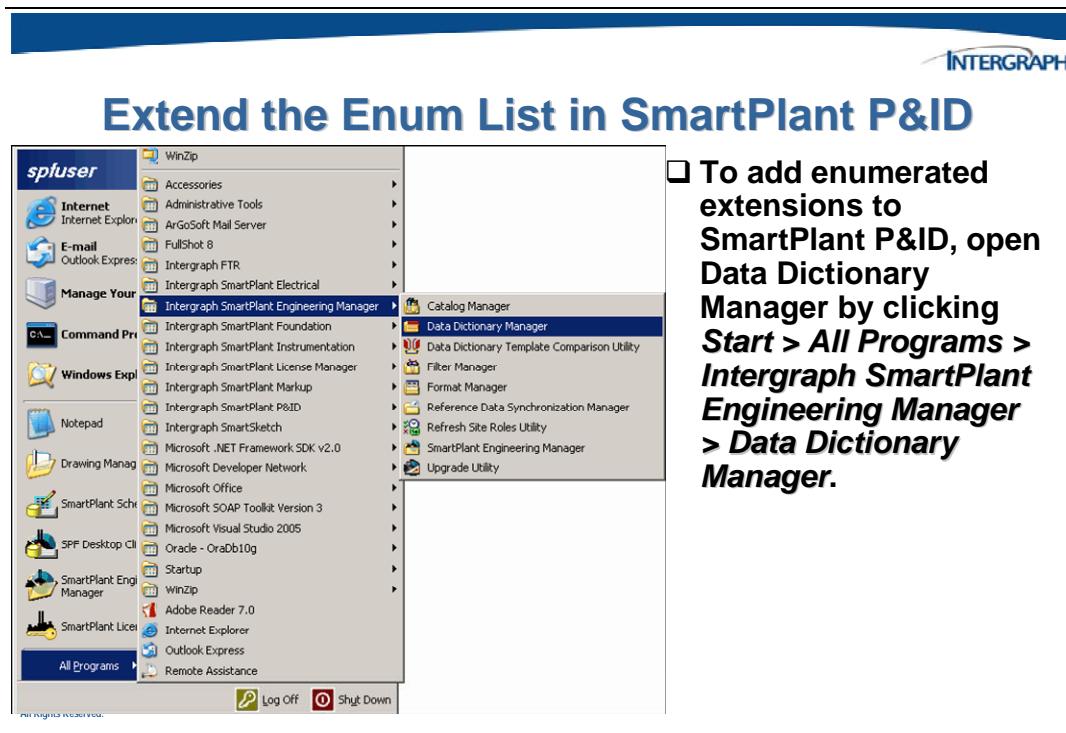
The other work process is to use the **schema editor to add new information to the tool meta schema, the schemas and perform the mapping from a single UI.**

© 2005 Intergraph Corp.
All Rights Reserved.

Configurable mapping, in which the authoring tool supplies a tool schema that defines the mapping between the tool schema and the SmartPlant schema and back, is not required for all tools that participate in SmartPlant. Authoring tool developers can hardcode the tool mapping in the tool adapter, although doing so is not recommended. The example in this chapter is only relevant for tools that implement configurable mapping.

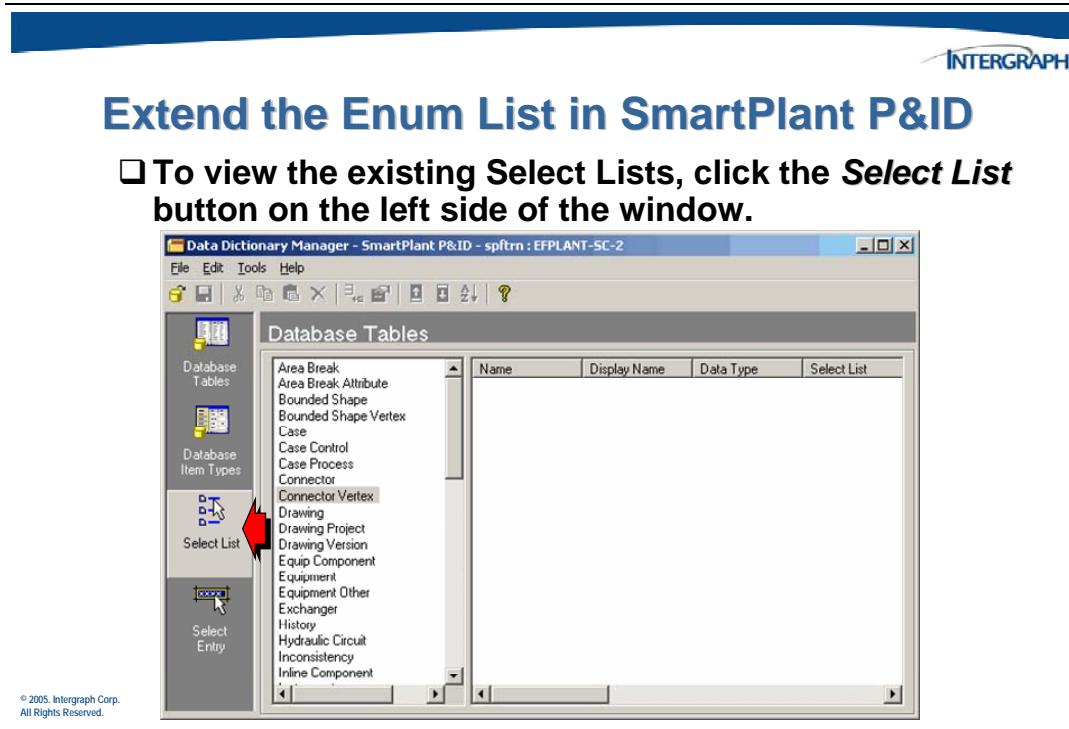
7.1 Extending Authoring Tool Enum Lists

The procedure for adding data definitions to the authoring tools differs from tool to tool. The following example describes the procedure for extending an existing *Select List* (enumerated list) in the SmartPlant P&ID data model using SmartPlant Data Dictionary Manager (delivered with SmartPlant Engineering Manager). For more information about adding properties to SmartPlant P&ID, see the SmartPlant Data Dictionary Manager documentation.

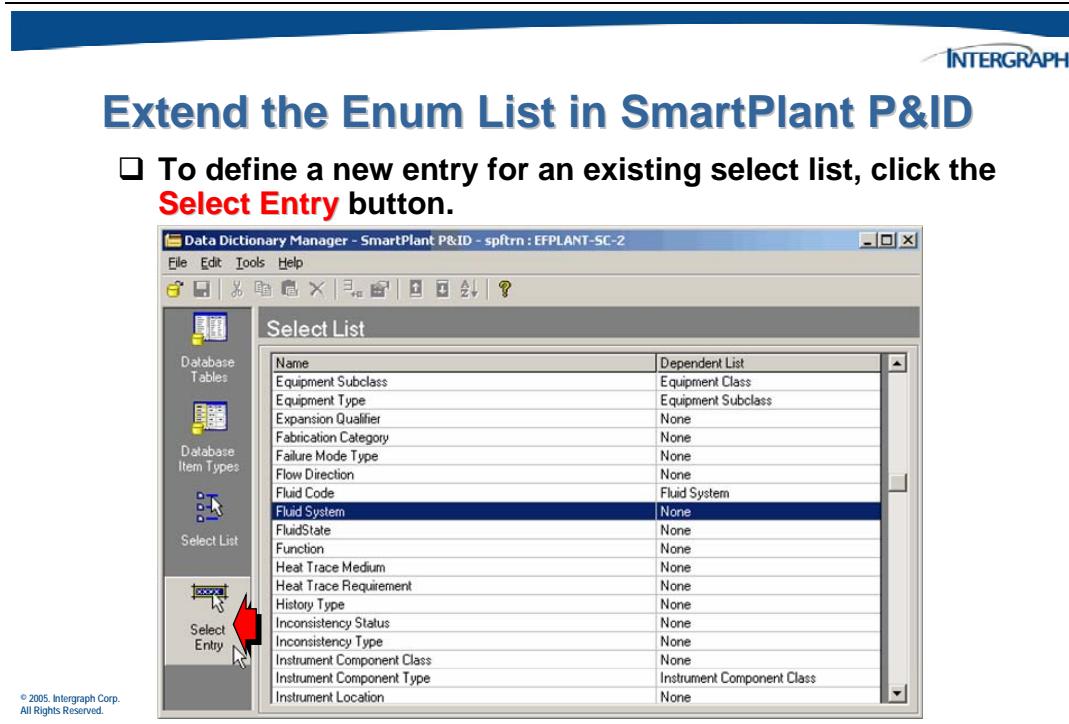


The following example describes how to extend an existing *Select List* in the SmartPlant P&ID data model. This type of operation is one of the most common ones performed.

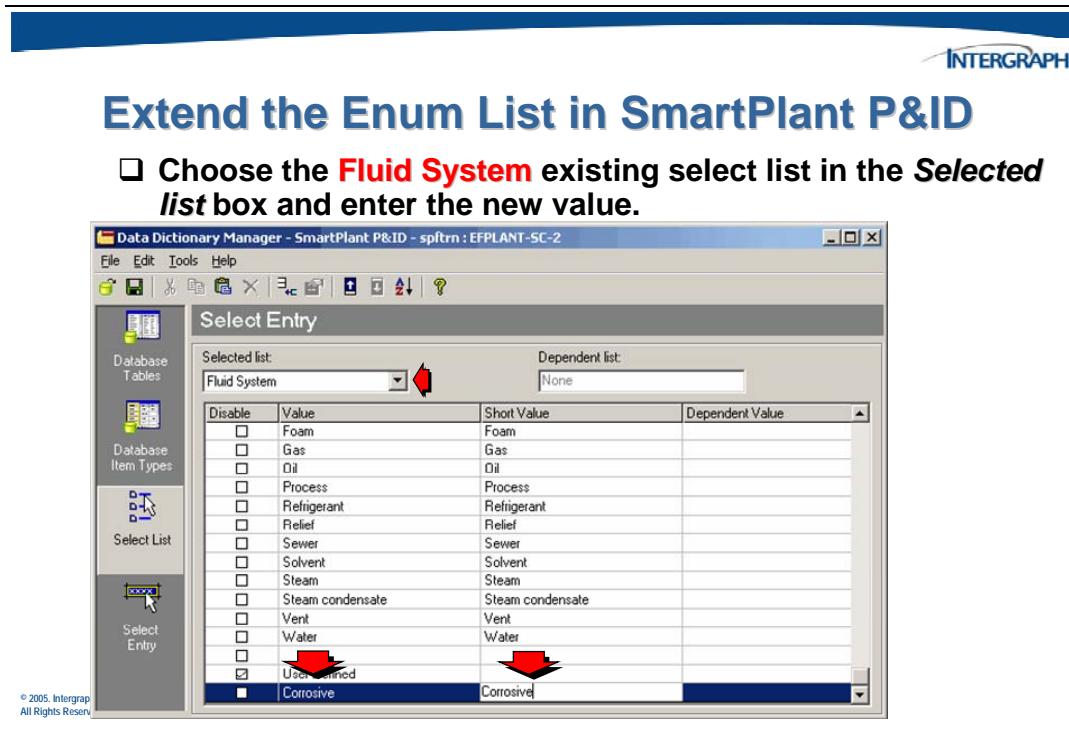
The *Data Dictionary Manager* application window will display.



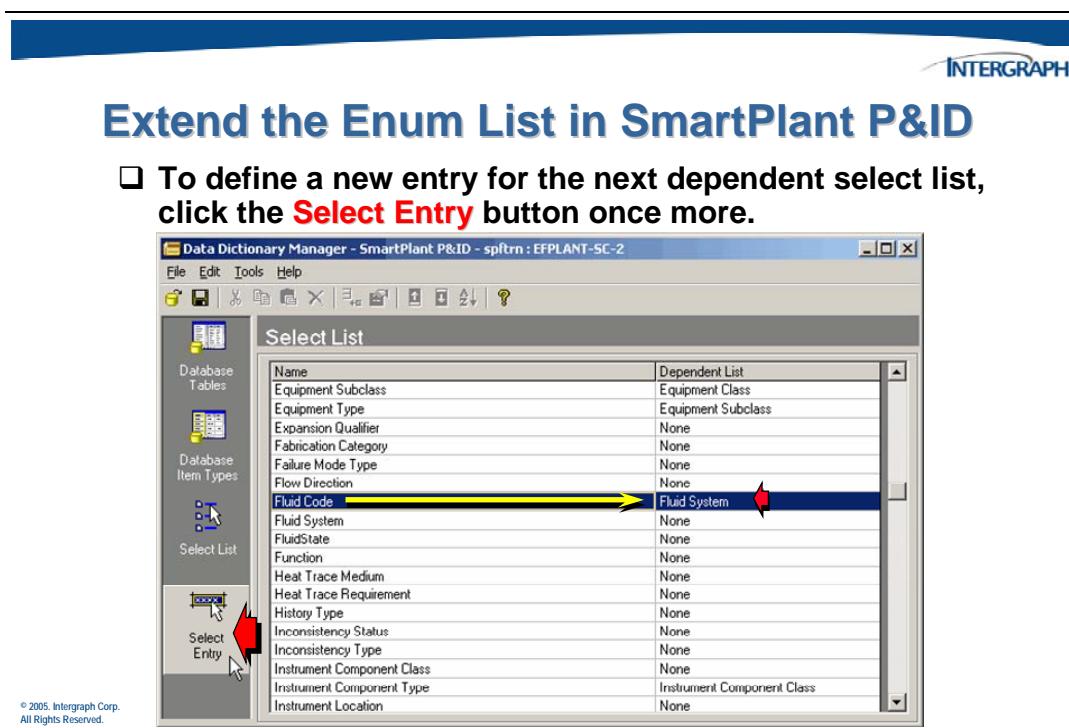
The *Select Entry* dialog will display.



Select the last entry in the list view (empty row) and type “**Corrosive**” as the *Value* and *Short Value*.



The *Fluid System* Select List has a dependency, **Fluid Code**, which requires new entries.



Add the first new select entry to the list by typing it in the last (empty) row of the table.

Extend the Enum List in SmartPlant P&ID

- Enter new **Value** and **Short Value** for the new **Fluid Code** entry then click the **Add Row** button to add another entry.

Disable	Value	Short Value	Dependent Value
<input type="checkbox"/>	WN	Brine water	Water
<input type="checkbox"/>	WP	Process water	Water
<input type="checkbox"/>	WPT	Potable water	Water
<input type="checkbox"/>	WPT_AI	Potable water or instrument air	Water
<input type="checkbox"/>	WR	Raw water	Water
<input type="checkbox"/>	WS	Sea water	Water
<input type="checkbox"/>	WSR	Sea water return	Water
<input type="checkbox"/>	WSS	Sea water supply	Water
<input type="checkbox"/>	WT	Test water	Water
<input type="checkbox"/>	WU	Utility water	Water
<input type="checkbox"/>	WW	Waste water	Water
<input type="checkbox"/>	WZ	Other water	Water
<input checked="" type="checkbox"/>	User Defined		
<input checked="" type="checkbox"/>	KA	(KA) Ammonia, Anhydrous	Corrosive

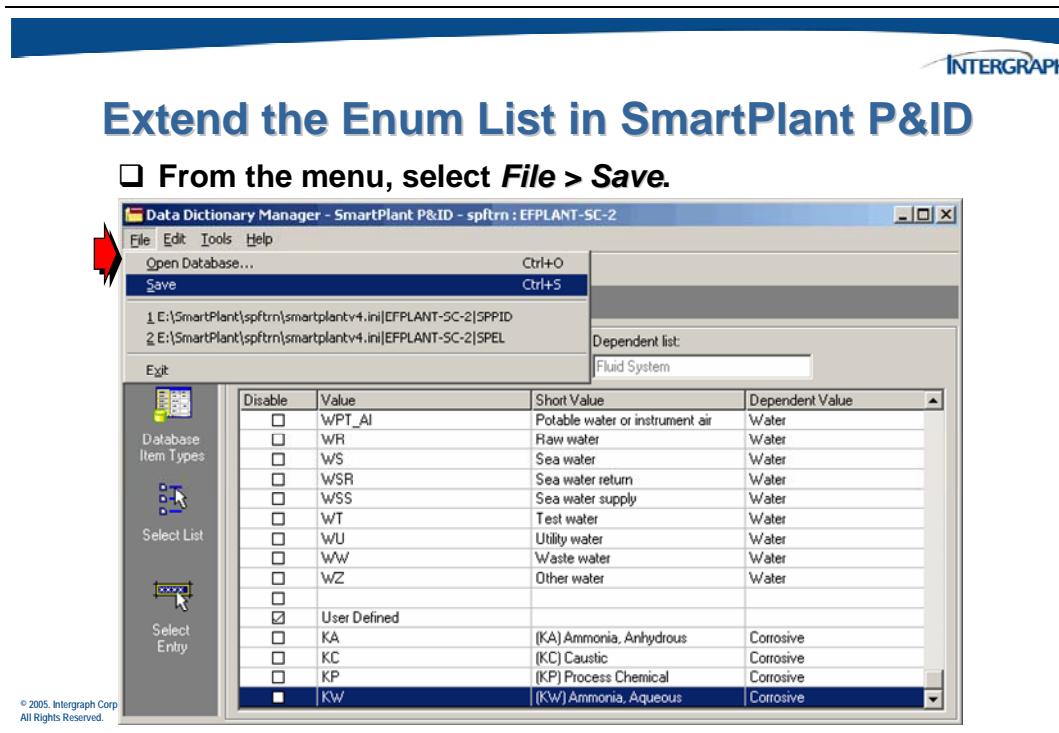
Define the next new select entry for the select list.

Extend the Enum List in SmartPlant P&ID

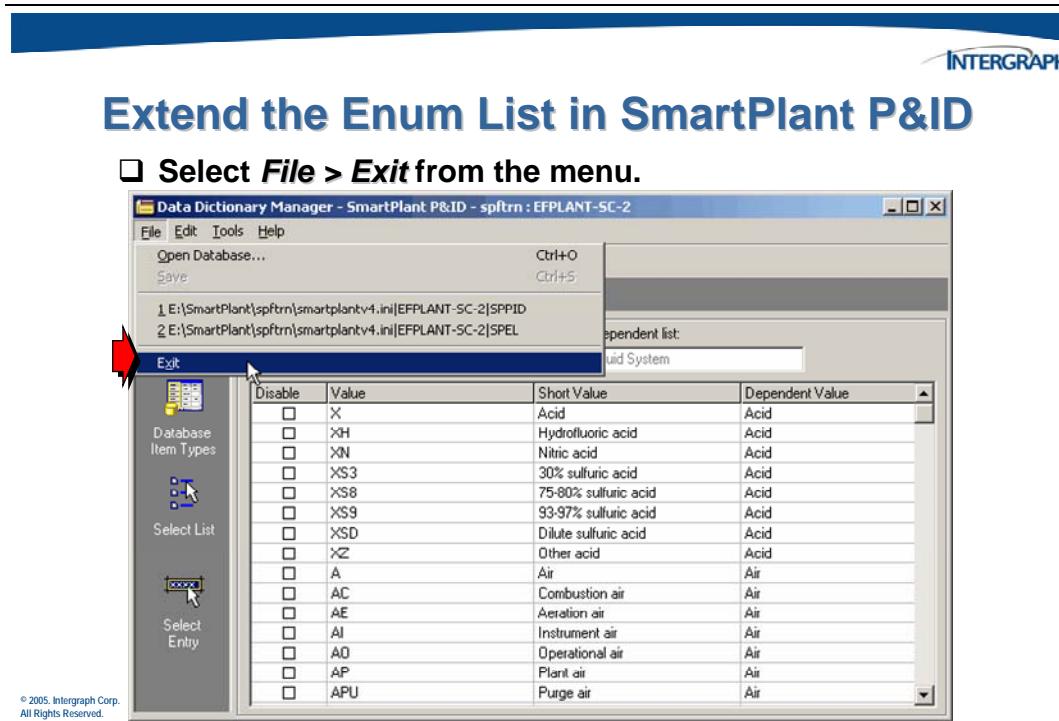
- On the toolbar, click the **Add Row** button to add another select entry.

Disable	Value	Short Value	Dependent Value
<input type="checkbox"/>	WP	Process water	Water
<input type="checkbox"/>	WPT	Potable water	Water
<input type="checkbox"/>	WPT_AI	Potable water or instrument air	Water
<input type="checkbox"/>	WR	Raw water	Water
<input type="checkbox"/>	WS	Sea water	Water
<input type="checkbox"/>	WSR	Sea water return	Water
<input type="checkbox"/>	WSS	Sea water supply	Water
<input type="checkbox"/>	WT	Test water	Water
<input type="checkbox"/>	WU	Utility water	Water
<input type="checkbox"/>	WW	Waste water	Water
<input type="checkbox"/>	WZ	Other water	Water
<input checked="" type="checkbox"/>	User Defined		
<input checked="" type="checkbox"/>	KC	(KC) Caustic	Corrosive

Define the final new select list entries and then save the changes.



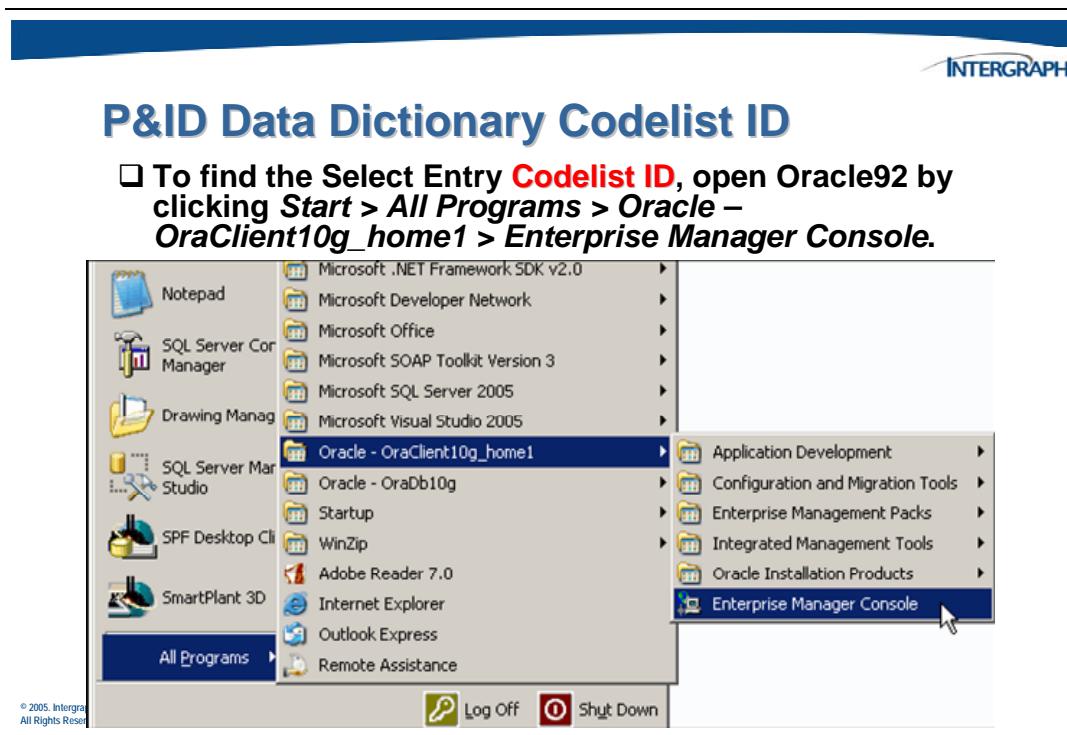
Once the changes have been saved, close the Data Dictionary Manager.



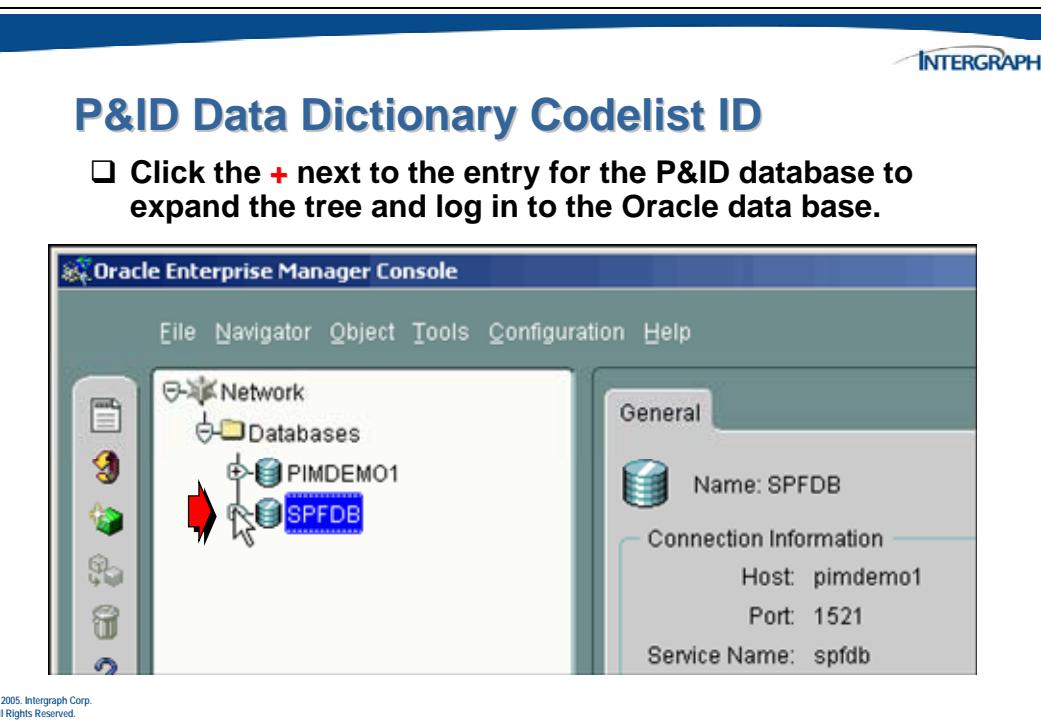
7.2 Finding the SmartPlant P&ID Data Dictionary Code List ID

In order to modify an existing enumerated list in the tool map schema, the enumerated codelist ID is used by the mapping environment software for creating the new entries. To see the ID's created by the Data Dictionary Manager, browse the contents of two Oracle tables in the SP P&ID database.

From the **Start** menu, launch the Oracle *Enterprise Manager Console* application.



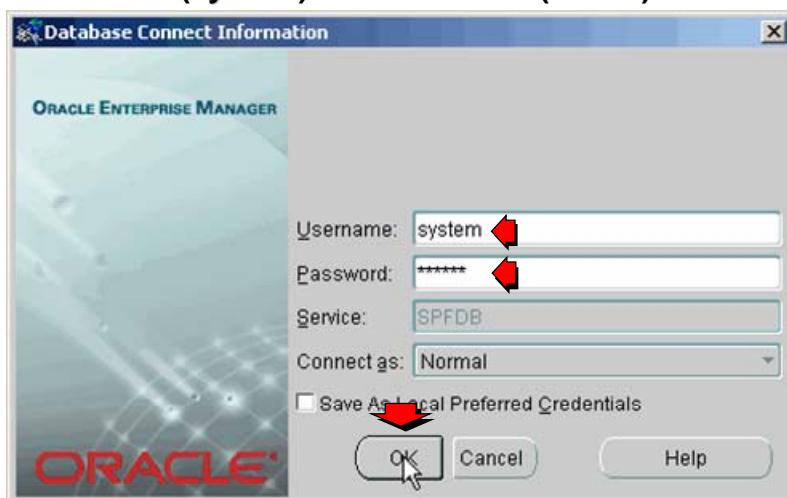
Open the database instance where the SmartPlant P&ID database is located.

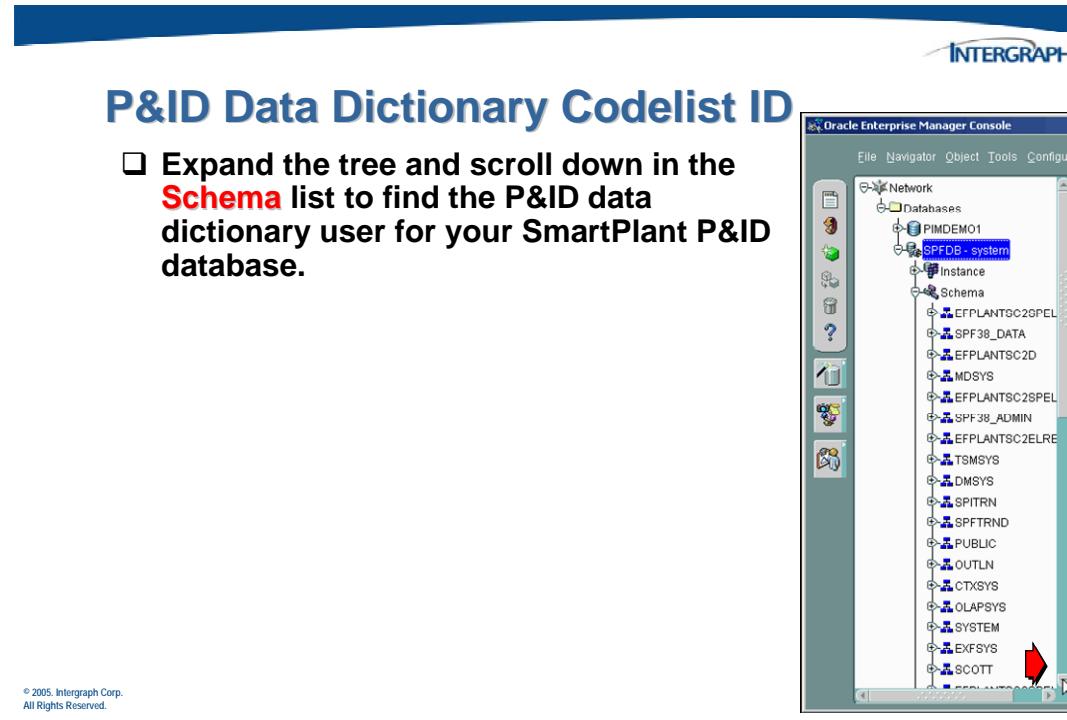


You will be prompted to supply a **username** and **password** to access the database instance.

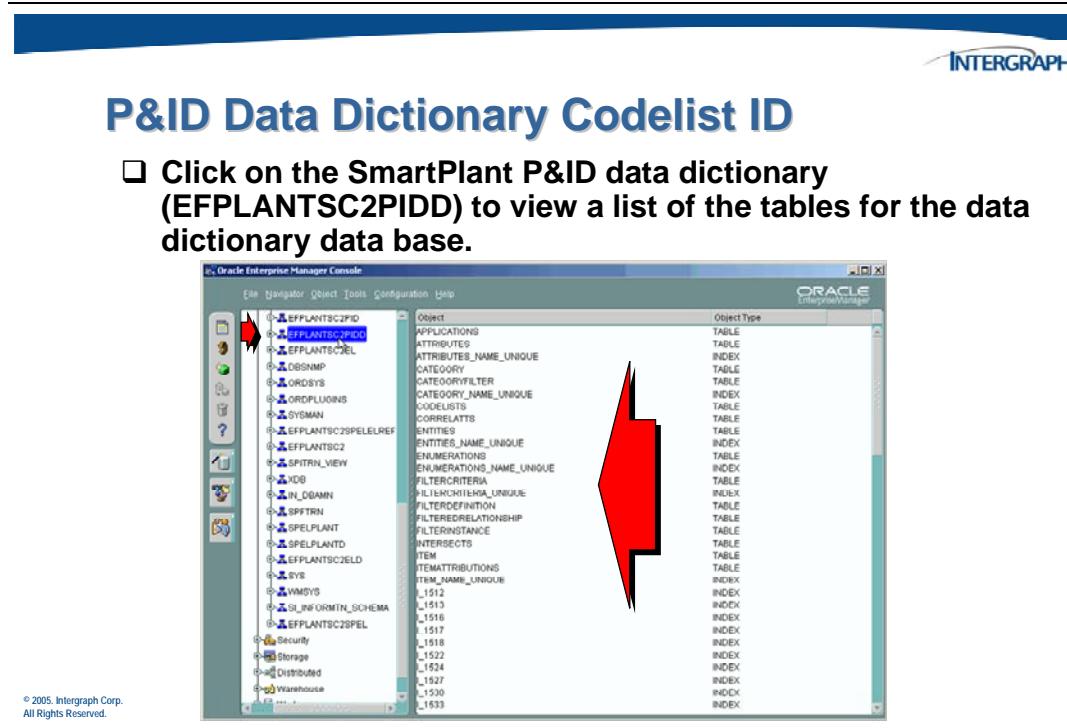
P&ID Data Dictionary Codelist ID

- In the **Oracle Enterprise Manager** dialog box, enter the **Username** (system) and **Password** (oracle) and click **OK**.

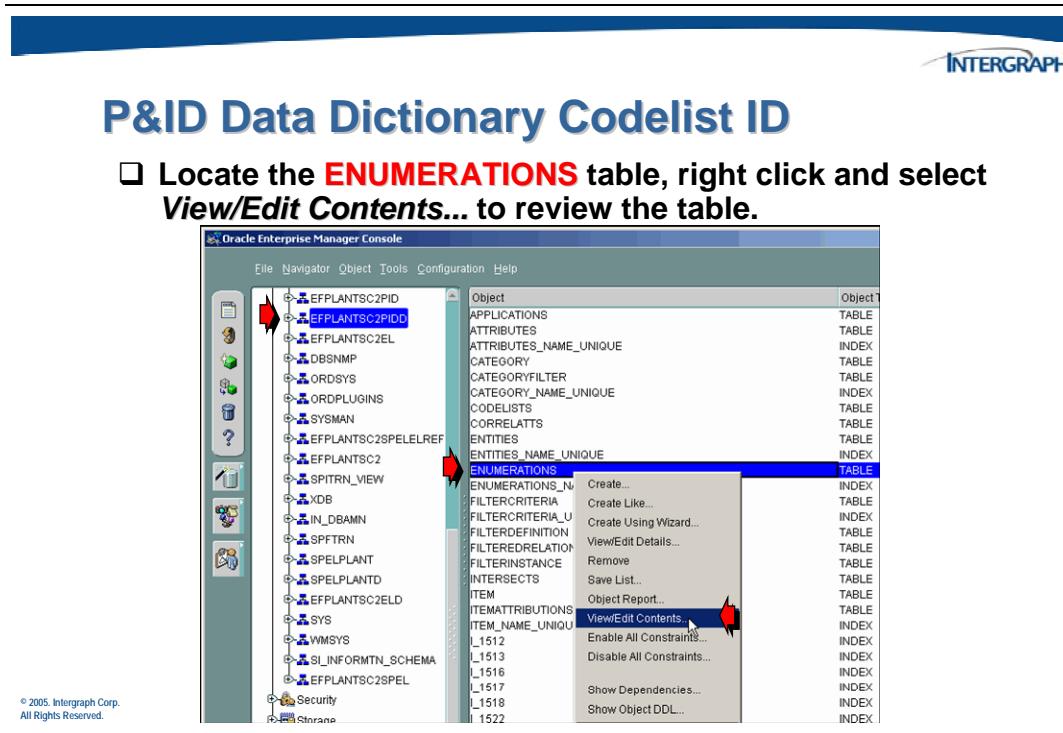




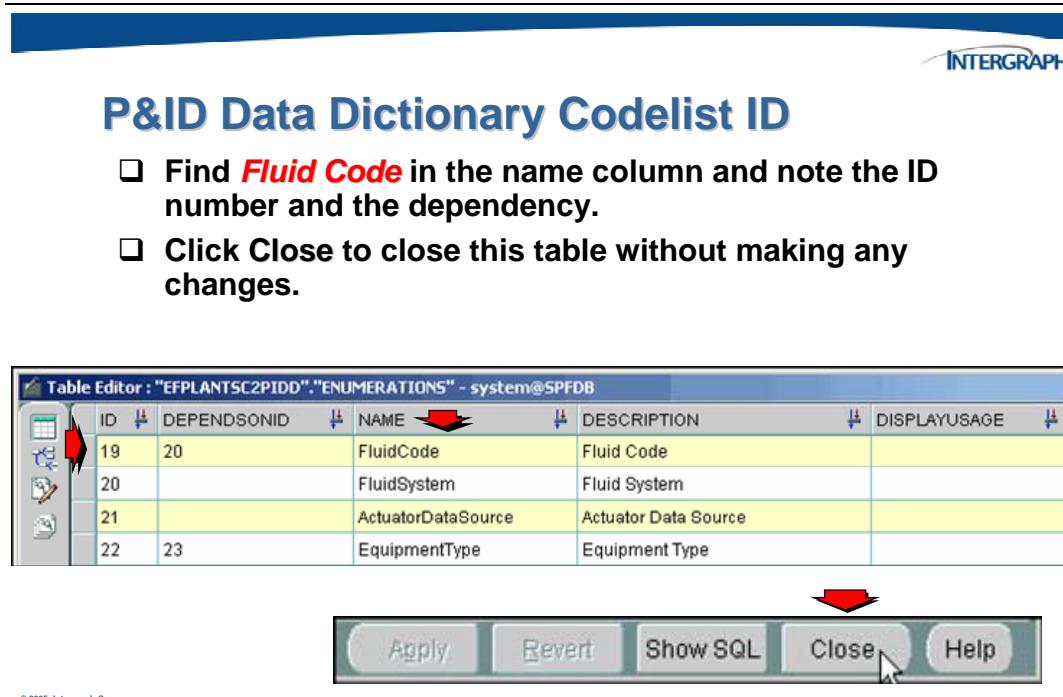
Next, locate the PID data dictionary user for your SmartPlant P&ID database. In the following example, the Oracle user is called **EFPLANTSC2PIDD**.



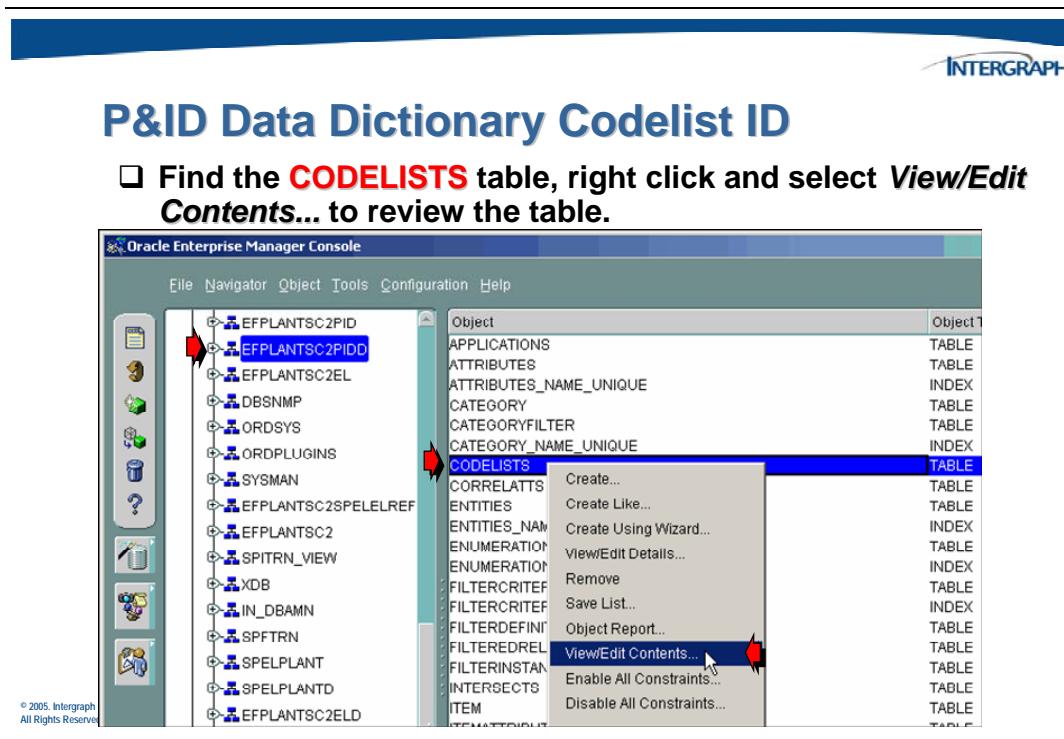
Select the ENUMERATIONS table, then right click to view and edit the table contents.



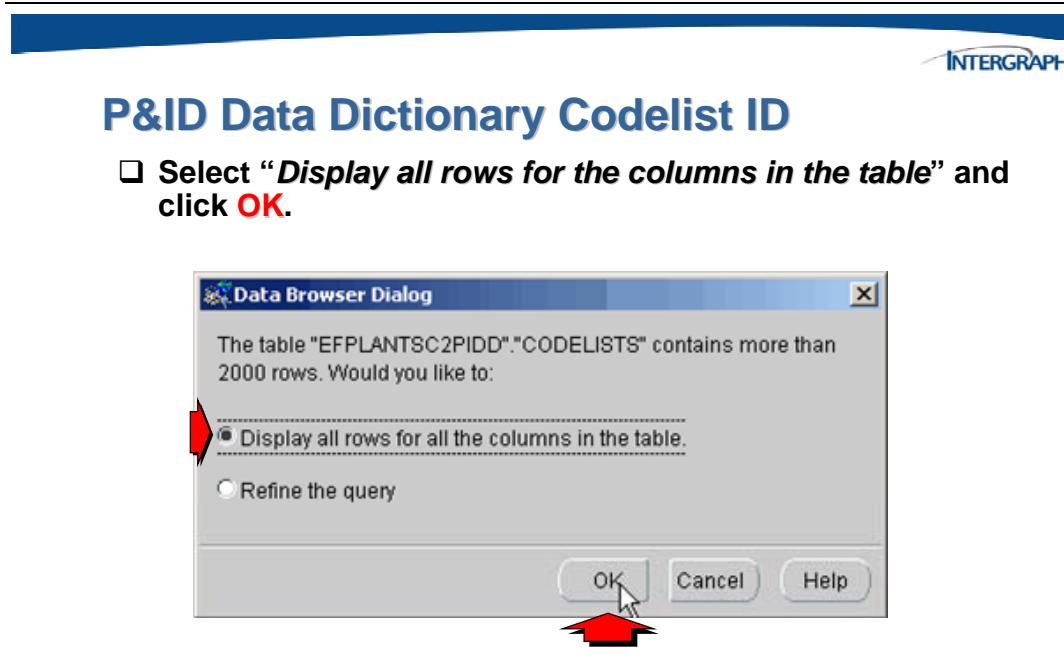
Locate the rows to view the ID's for *FluidSystem* and *FluidCode*. In this example they are **19** and **20** so make a note of these ID's for the mapping.



Close the table with out making any changes. Next, review the contents of the *Select Lists* from the CODELISTS table.



An informational “*Data Browser Dialog*” will be displayed.





P&ID Data Dictionary Codelist ID

- Sort the table by CODELIST_INDEX and scroll (if necessary) to appropriate name in the CODELIST_TEXT column that contain the code list values added earlier in the data dictionary manager.

The screenshot shows a Microsoft Excel-like Table Editor window titled "Table Editor : 'EFPLANTSC2PIDD'.'CODELIST_ID'". The table has columns: CODELIST_NUMBER, CODELIST_INDEX, CODELIST_TEXT, CODELIST_SHORT_TEXT, and CODELIST_CONST. The data includes several rows, with the last five highlighted in yellow. A red box surrounds the last five rows. Red arrows point from the following list items to specific parts of the interface:

- Sort the table by CODELIST_INDEX and scroll (if necessary) to appropriate name in the CODELIST_TEXT column that contain the code list values added earlier in the data dictionary manager.
- Match what you added in the Data Dictionary Manager to the values in the CODELIST_SHORT_TEXT column. For example, the entry would contain (KA) Ammonia, Anhydrous and the next column left, CODELIST_TEXT would contain KA. This will show you the values that were entered in the Data Dictionary Manager earlier.
- Locate the CODELIST_INDEX column and view the value for KA. In this example it is 10001. Next, find the CODELIST_NUMBER column and verify that the value is 19 which is the same as the value determined earlier for the enumeration ID.
- These values will be used to define a UID value of SP_19_10001 when the SmartPlant P&ID tool map schema file is extended. The UID value is SP+CODELIST_NUMBER + CODELIST_INDEX to give a value of SP_19_10001.

© 2005, Intergraph Corp.
All Rights Reserved.

Match what you added in the Data Dictionary Manager to the values in the CODELIST_SHORT_TEXT column. For example, the entry would contain **(KA) Ammonia, Anhydrous** and the next column left, CODELIST_TEXT would contain **KA**. This will show you the values that were entered in the Data Dictionary Manager earlier.

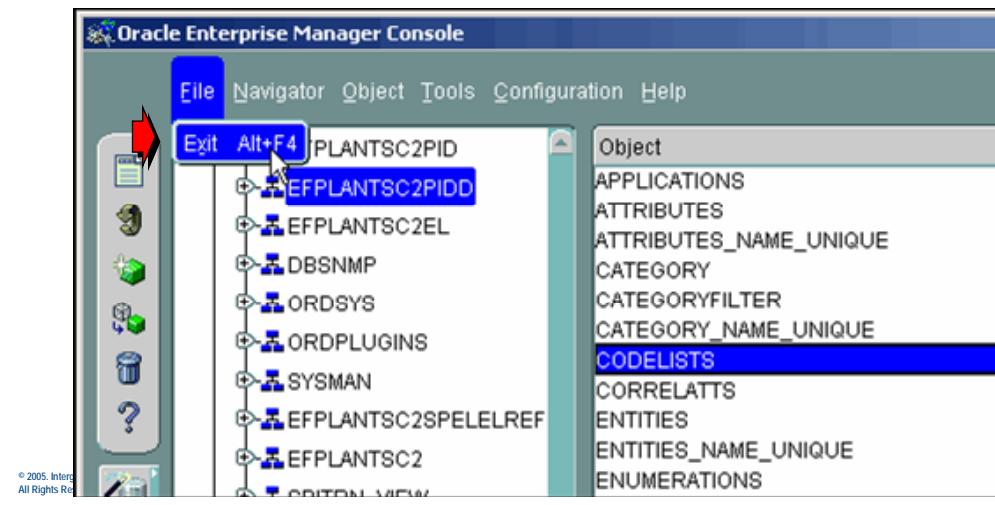
Locate the CODELIST_INDEX column and view the value for **KA**. In this example it is **10001**. Next, find the CODELIST_NUMBER column and verify that the value is **19** which is the same as the value determined earlier for the enumeration ID.

These values will be used to define a UID value of **SP_19_10001** when the SmartPlant P&ID tool map schema file is extended. The UID value is **SP+CODELIST_NUMBER + CODELIST_INDEX** to give a value of **SP_19_10001**.



P&ID Data Dictionary Codelist ID

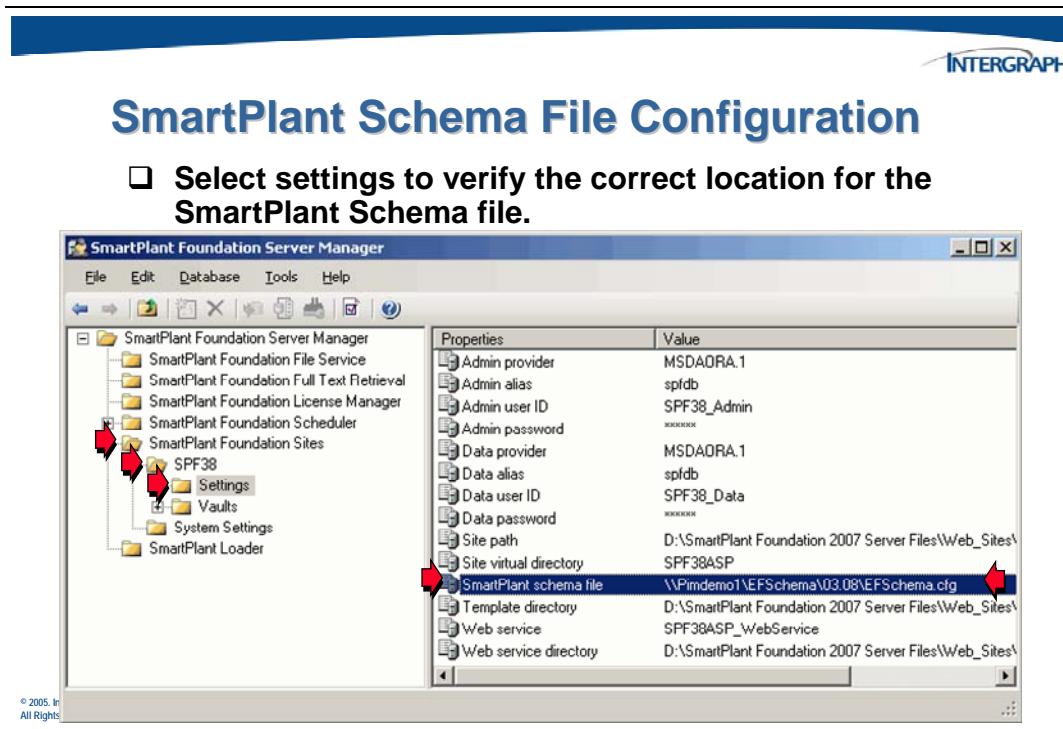
- Select **File > Exit** to exit from Oracle Enterprise Manager **without** making any changes.



7.3 SmartPlant Schema Additions

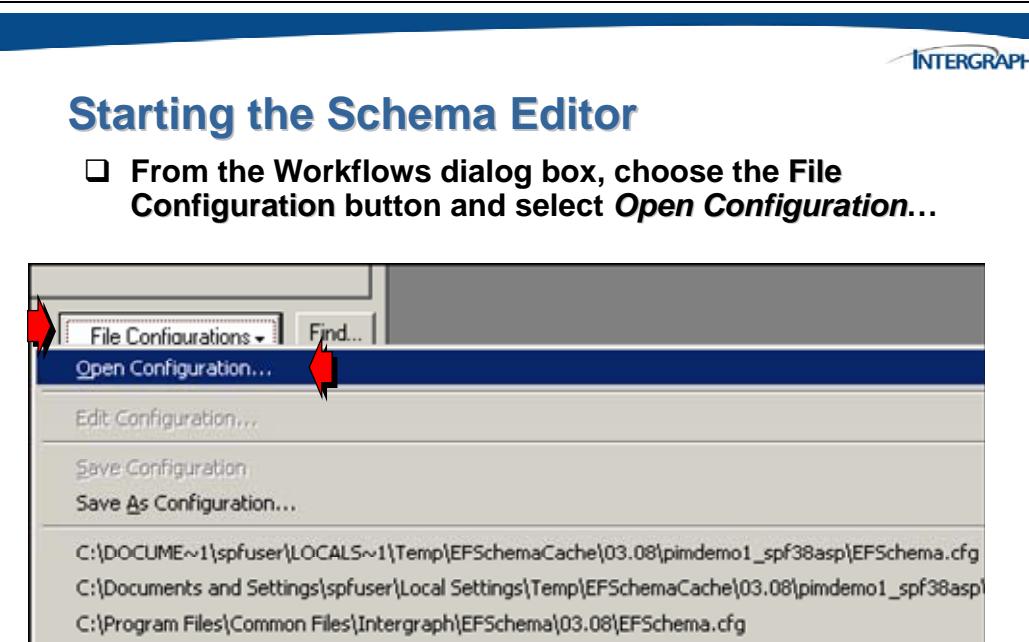
Before you can map a property from the tool schema to the SmartPlant schema and back, you must use the Schema Editor to add the new property to the SmartPlant schema.

In order for the Schema Editor to open the connect to SmartPlant and use the correct schema file, the SmartPlant schema file setting should be verified. This is done through SmartPlant Foundation Server manager by selecting the **SmartPlant Foundation Sites > <site name> > Settings >** path.



Every application that integrates with the SmartPlant framework should have a **Tool** object in the schema (for each plant). The *Name* for the Tool object should match the name specified for the application when it registers with SmartPlant.

Start the Schema Editor and use a configuration file to open the schemas.

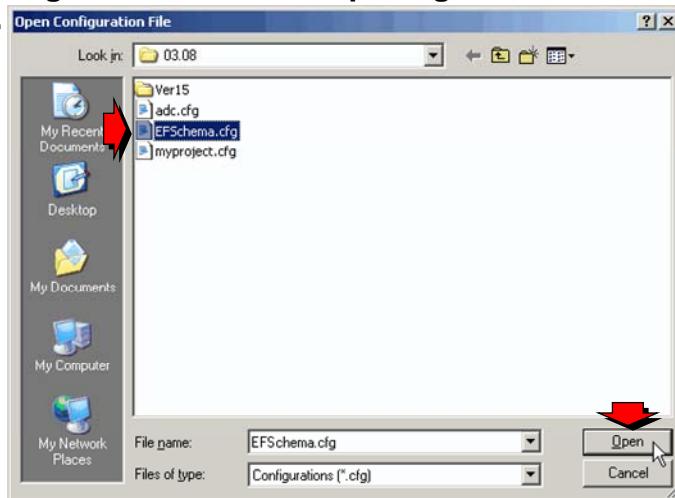


© 2005, Intergraph Corp.
All Rights Reserved.

The *Open Configuration file* dialog will display.

Starting the Schema Editor

- From the Workflows dialog box, choose the File Configuration button and select Open Configuration...



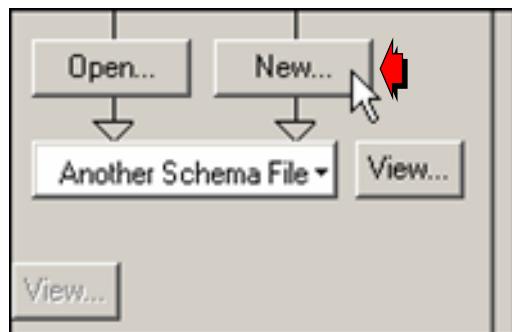
The configuration file **EFSchema.cfg** will open the master schema and all of the schemas necessary to add the extensions.

To create a new extension schema and to insure the EFSchema file is secure during editing, perform the following steps.



Starting the Schema Editor

- From the Workflows dialog box, create a new extension schema by selecting the New button above **Another Schema File**.



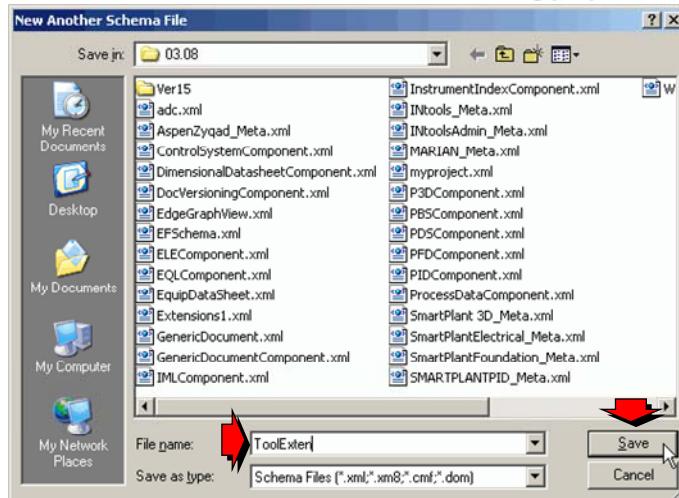
© 2005, Intergraph Corp.
All Rights Reserved.

Create the project schema using the **Another Schema File New** button.



Starting the Schema Editor

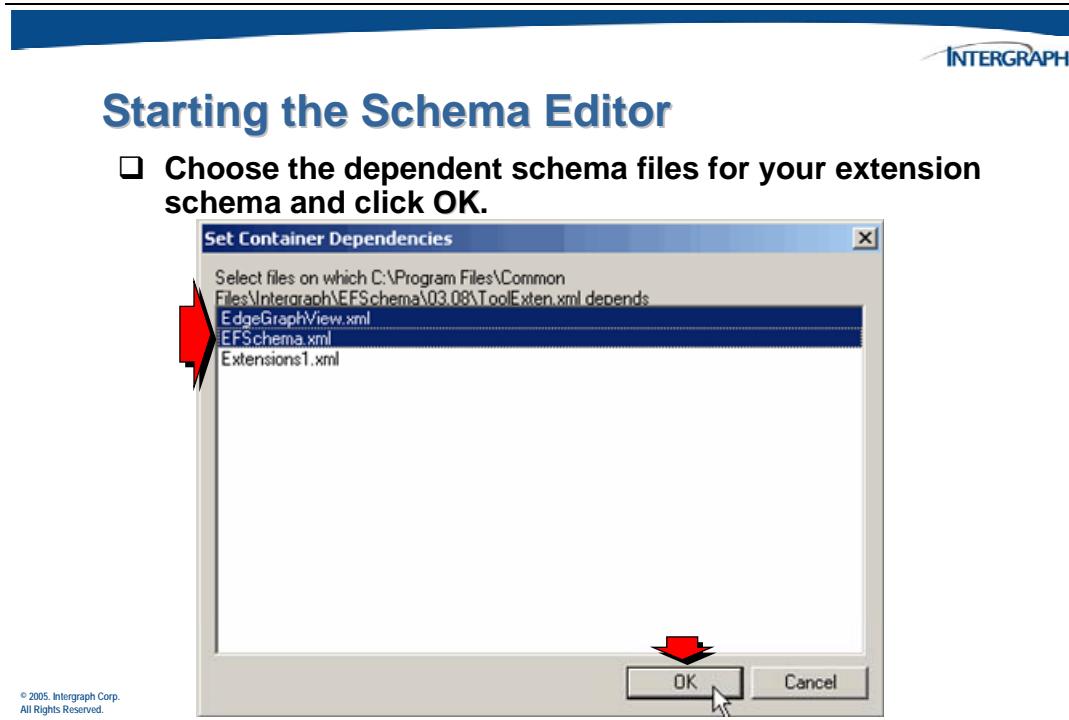
- In the **New Another Schema File** dialog, enter the name for the extension schema file and select **Save**.



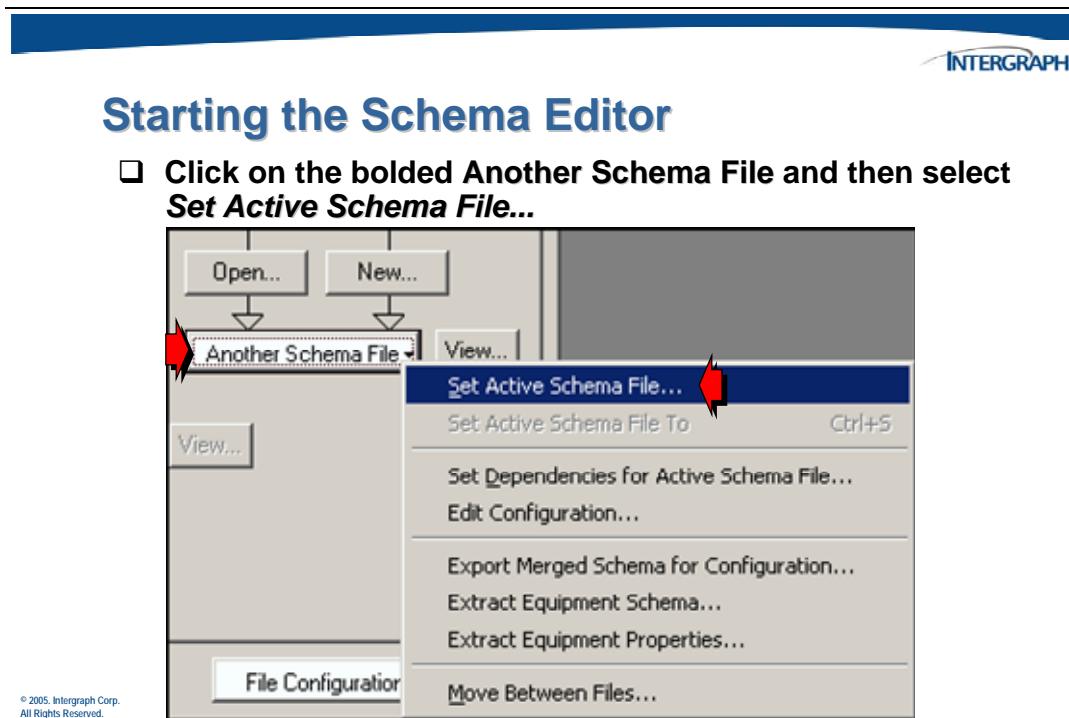
© 2005, Intergraph Corp.
All Rights Reserved.

Enter a name for the new project schema file, such as **ToolExten.xml**.

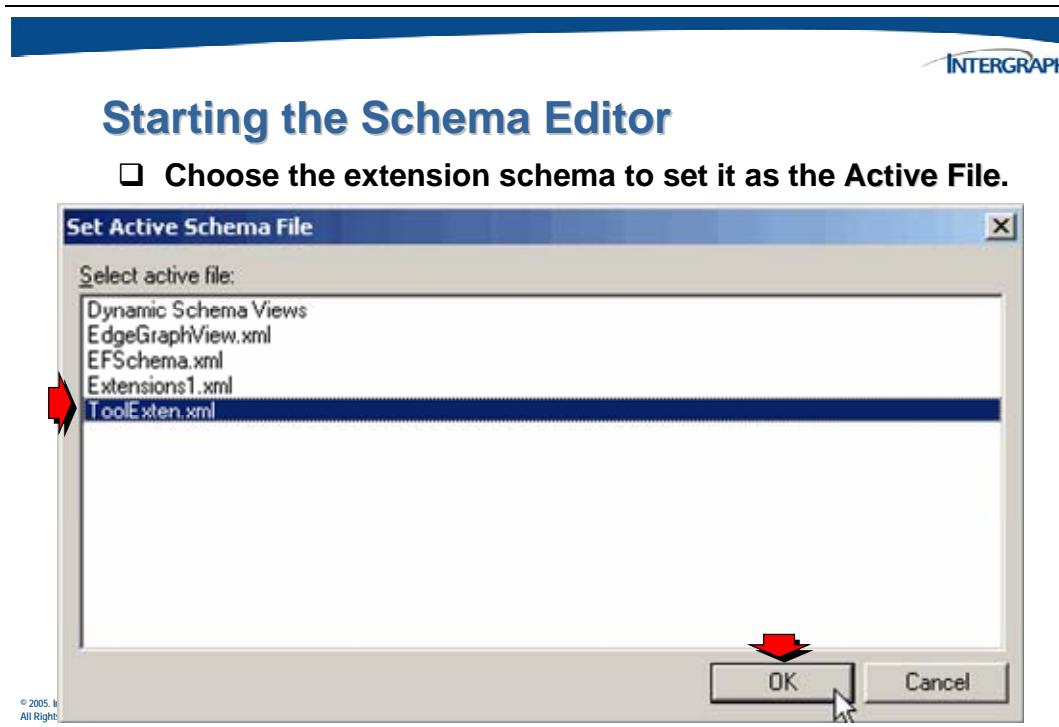
This will create an “empty container” xml that will store all site specific extensions. A *Set Container Dependencies* dialog will display with the **EFSchema.xml** file listed. Select the *master* schema file, other extension schemas which will make the **ToolExten.xml** dependent on the EFSchema.xml file.



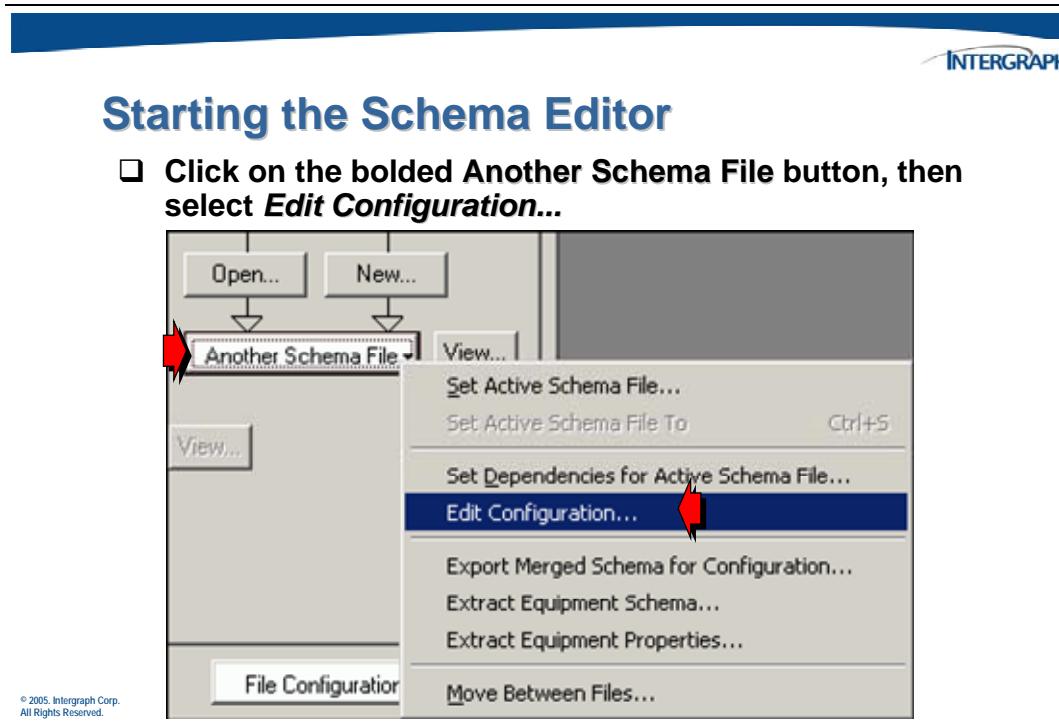
Next, set the file properties for the two files to work together by setting the **Active File**.



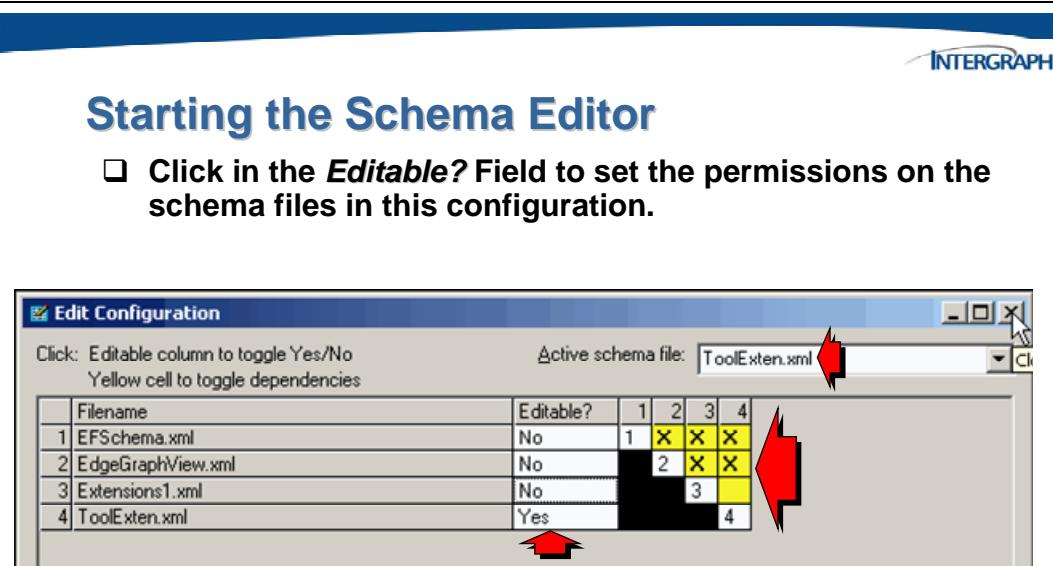
The *Set Active File* dialog will be displayed.



Finally choose the *Another Schema File* button again and select the *Edit Configuration* option.

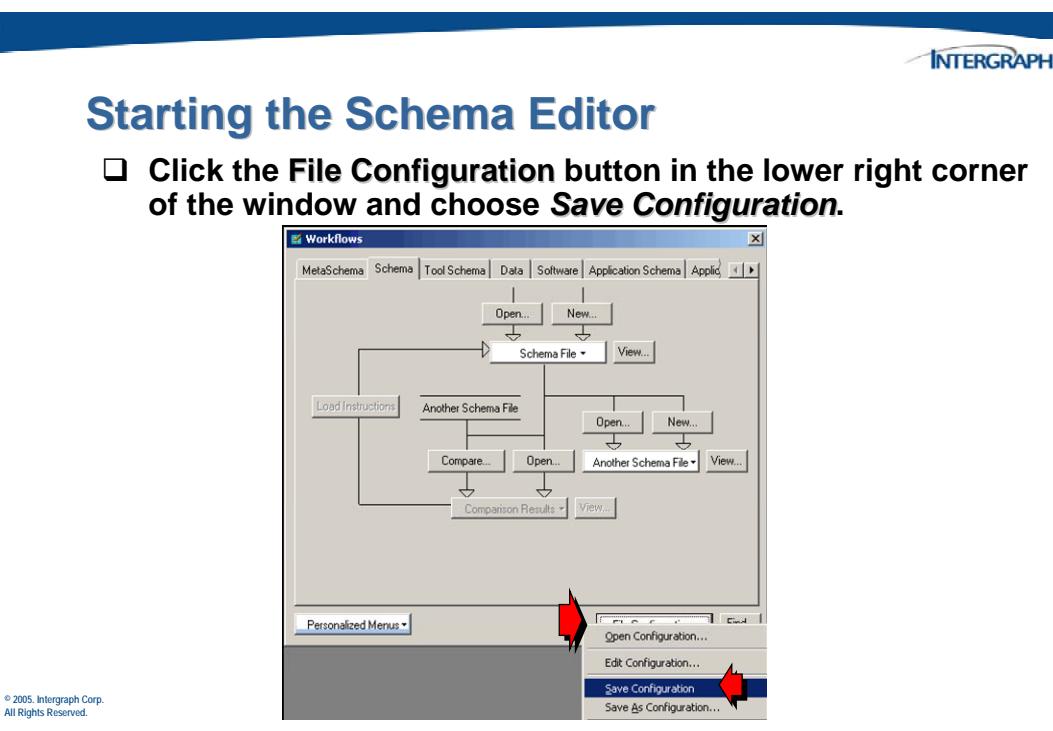


With this dialog you need to make the *EFSchema.xml* file and other extension schemas **Read Only** and the *ToolExten.xml* **Read/Write**.

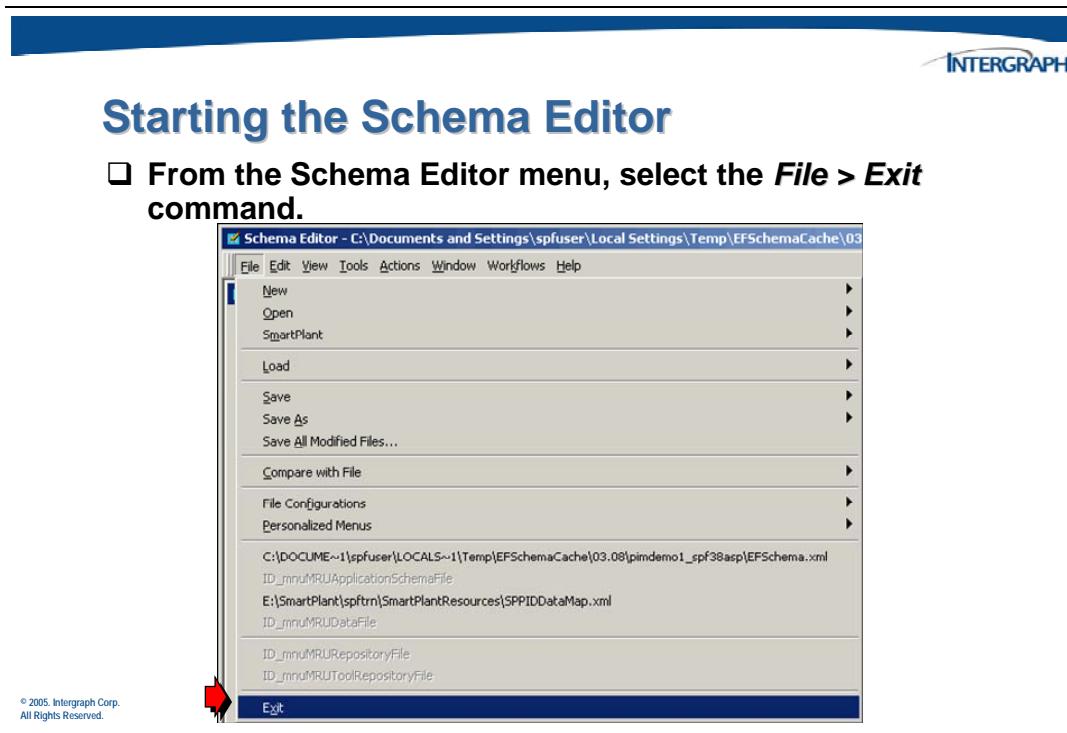


© 2005, Intergraph Corp.
All Rights Reserved.

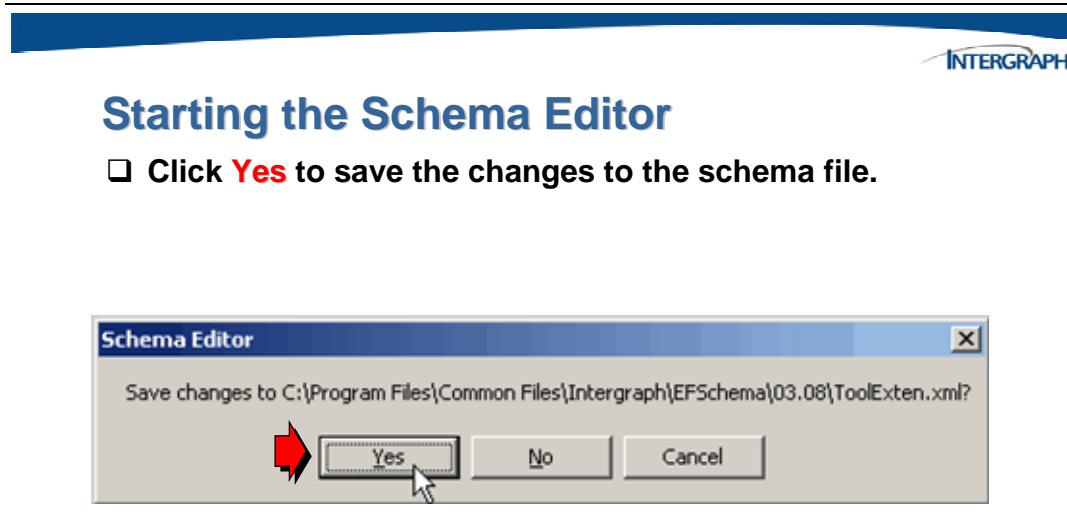
Now save the configuration for use in future editing sessions.



Use the **File** command on the main menu to **Exit** from the Schema Editor.

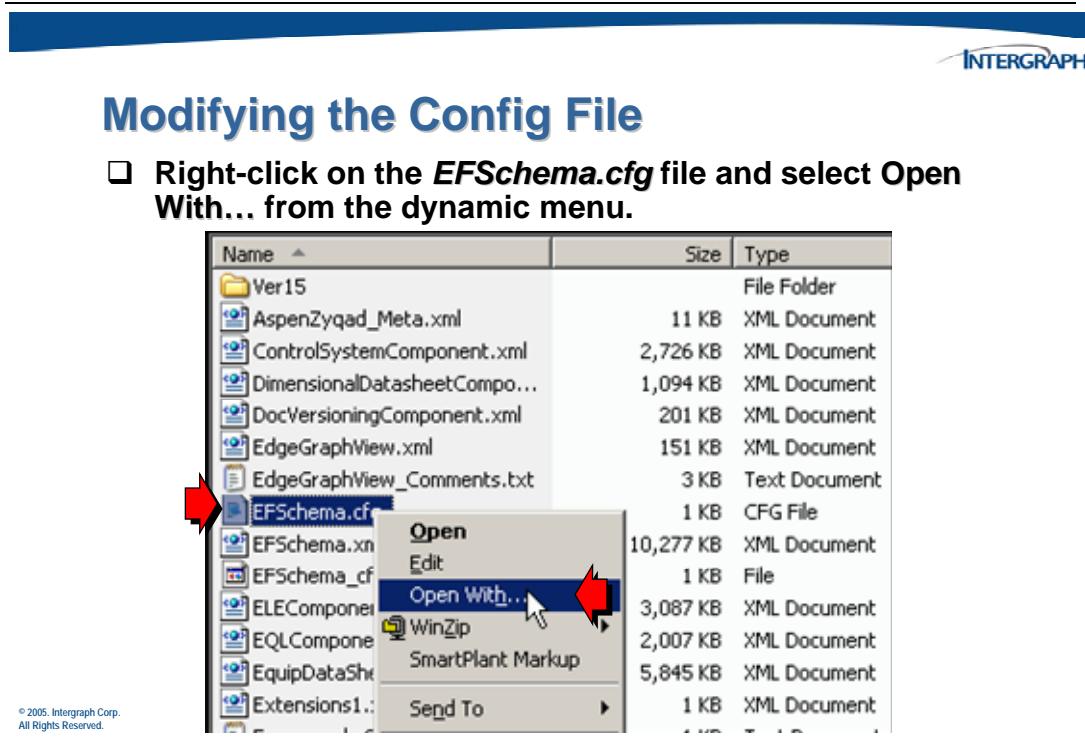


You will be prompted whether or not to save any changes that has been made to the schemas. The dependencies that have been set are stored in the header of each xml and will need to be saved.



7.3.1 Modifying the Config File

In the last section, a custom extensions file (container) was added to the configuration to contain the extensions for the authoring tools. Since the provided extension file, *Extensions1.xml*, was not used, it can be removed from the config file.

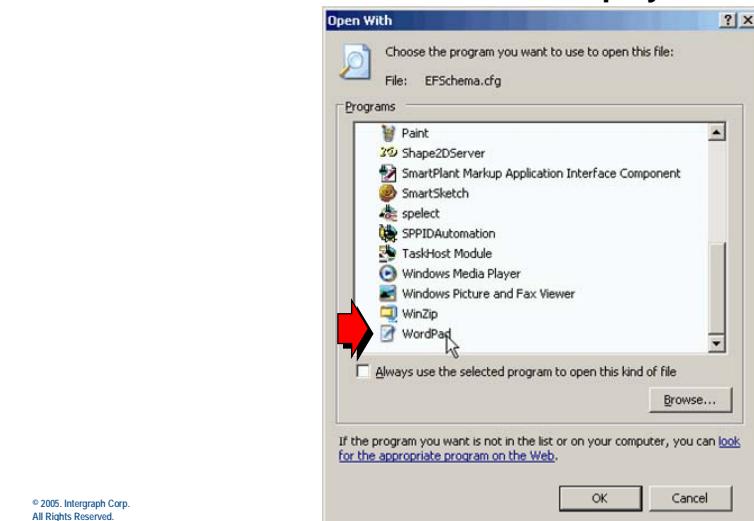


It is recommended that **WordPad** be used to make the modification to the config file so that it is easier to read when editing.



Modifying the Config File

- Choose WordPad from the displayed list.

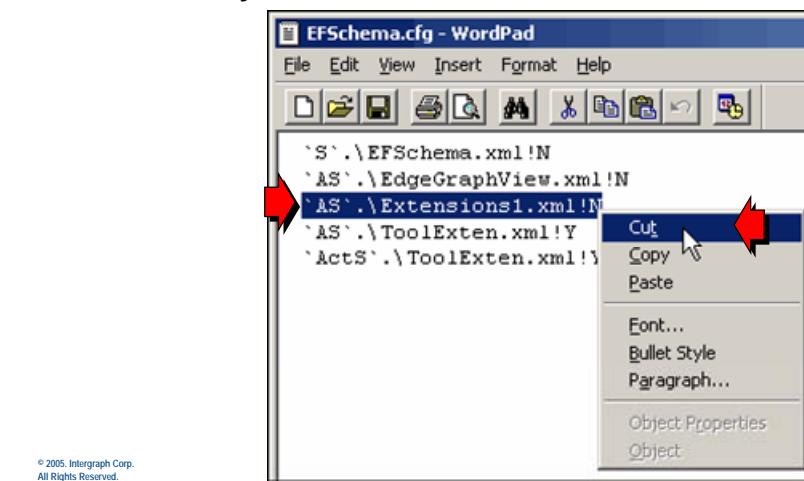


Remove the reference to **Extensions1.xml** from the config file since it will not be used. This will prevent any problems later when the schemas are loaded into the SPF Admin database.

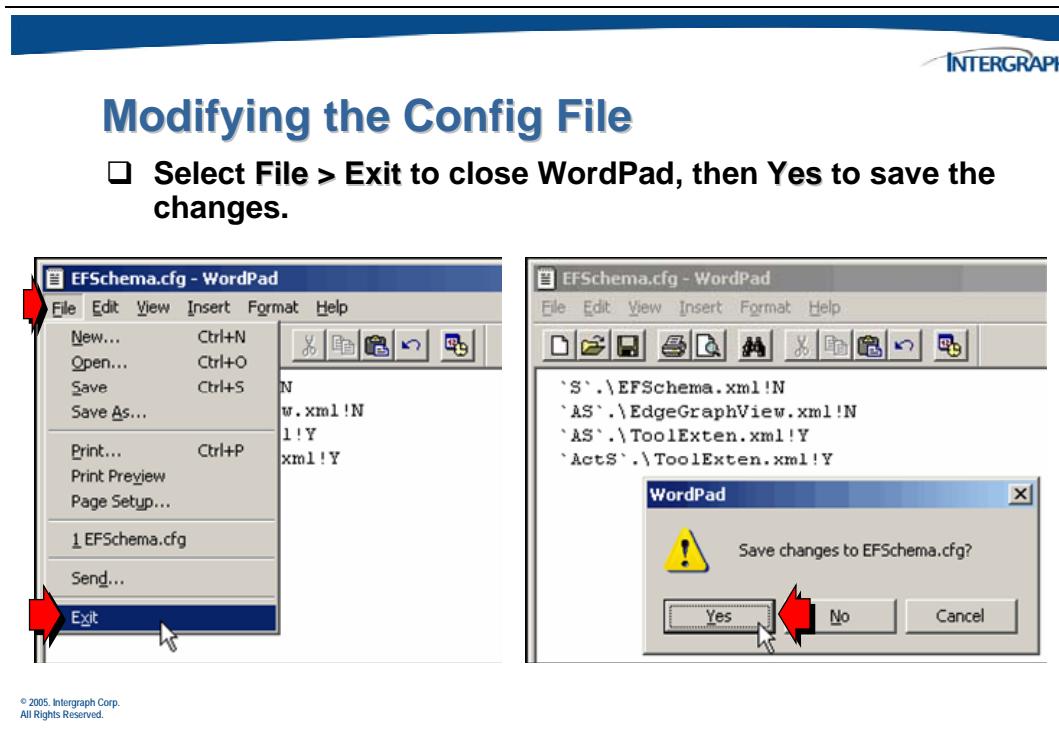


Modifying the Config File

- Highlight the **Extensions1.xml** entry and choose Cut from the dynamic menu.



After the modification has been made, exit from WordPad and save the changes.



7.4 Tool Schemas Overview

Each authoring tool defines a tool map schema file that determines the mapping between objects in the tool map schema and the SmartPlant schema. For example, the Drawing class in SmartPlant P&ID maps to the PIDDrawing class in the SmartPlant schema. These map files are used during publish and retrieve to convert tool data to SmartPlant data and vice versa.



Tool Map Schemas

Tool map schemas describe the data in the authoring tool databases before it is transformed into the format prescribed by the SmartPlant schema. Tool map schemas also define the mapping between objects in the tool map schema and the SmartPlant schema.

When a document is published, the authoring tool software converts tool map schema objects into SmartPlant schema objects.

When a file is retrieved into an authoring tool, the authoring tool converts the SmartPlant schema objects into tool map schema objects.

© 2005, Intergraph Corp.
All Rights Reserved.



Tool Map Schemas

For every tool map schema used by the application there should be a **ToolSchema** object (in the schema for each plant).

Each **ToolSchema** object should have a **Name** that identifies which of the tool map schemas this object applies to.

Each **ToolSchema** object should be tied to the **Tool** object.

Each **ToolSchema** object should indicate whether the tool map schema is publish only, retrieve only or publish and retrieve.

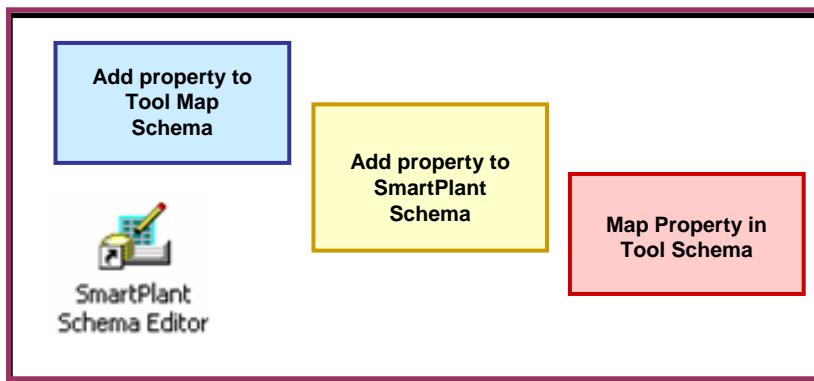
© 2005, Intergraph Corp.
All Rights Reserved.

7.5 Modifying the SmartPlant Schema

Once the new enumerated values have been added to the authoring tool, these values must also be added to the SmartPlant Schemas via the Tool extension schema (ToolExten.xml). Unlike the tool map schema, it is NOT critical that the value added match what has been added to the tool meta schema (data base). When you perform the mapping later, that is the procedure that will match the value in the master schema to a value in the tool schema.



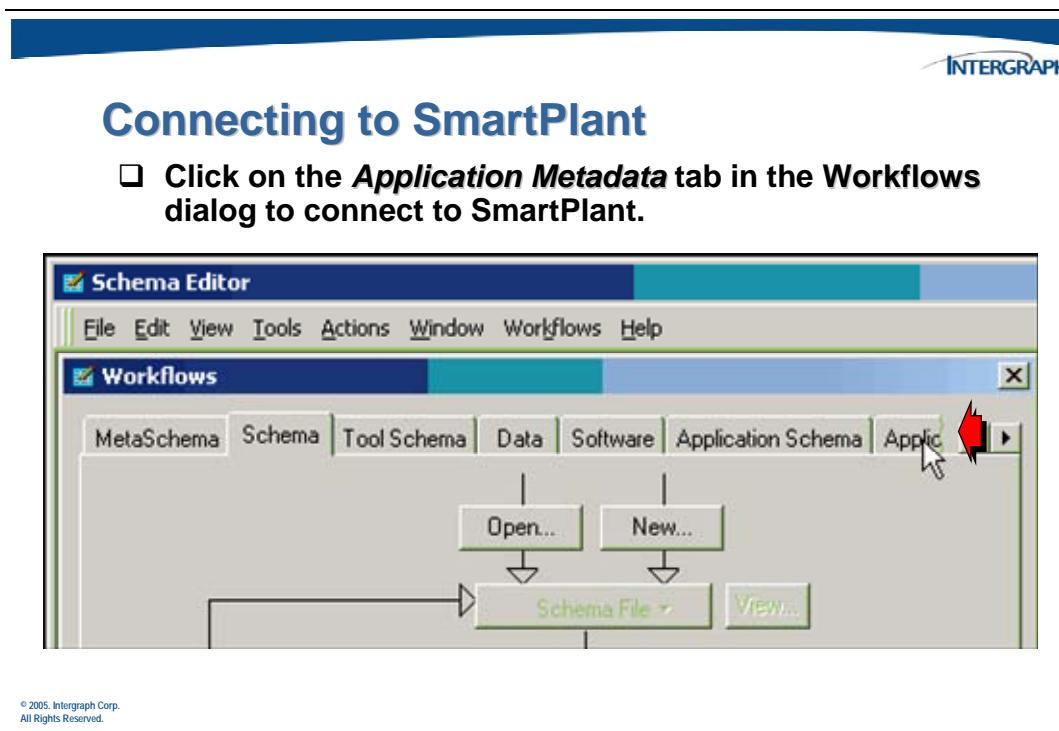
Mapping with the Schema Editor



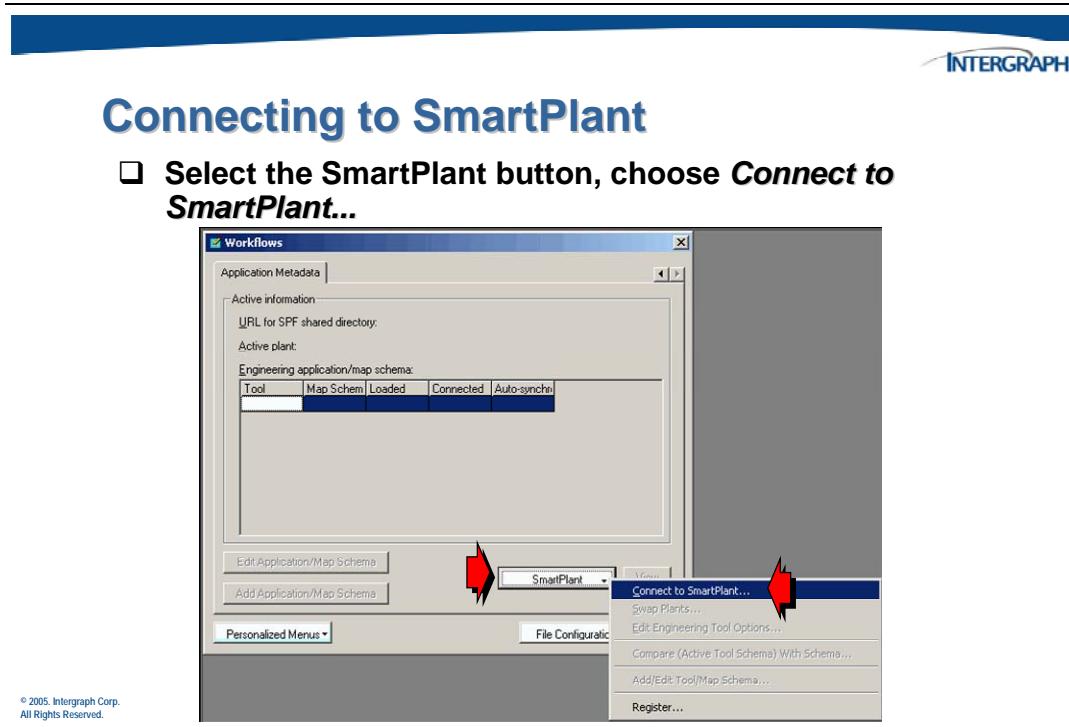
7.5.1 Connecting to SmartPlant

The steps necessary to modify the extension schema involves using the schema editor to connect to SmartPlant and then synchronize the tool meta schema with the tool map schema. This causes the tool meta adapter to read the contents of the tool meta schema and automatically add any new definitions to the corresponding tool map schema. This is called synchronizing the schema.

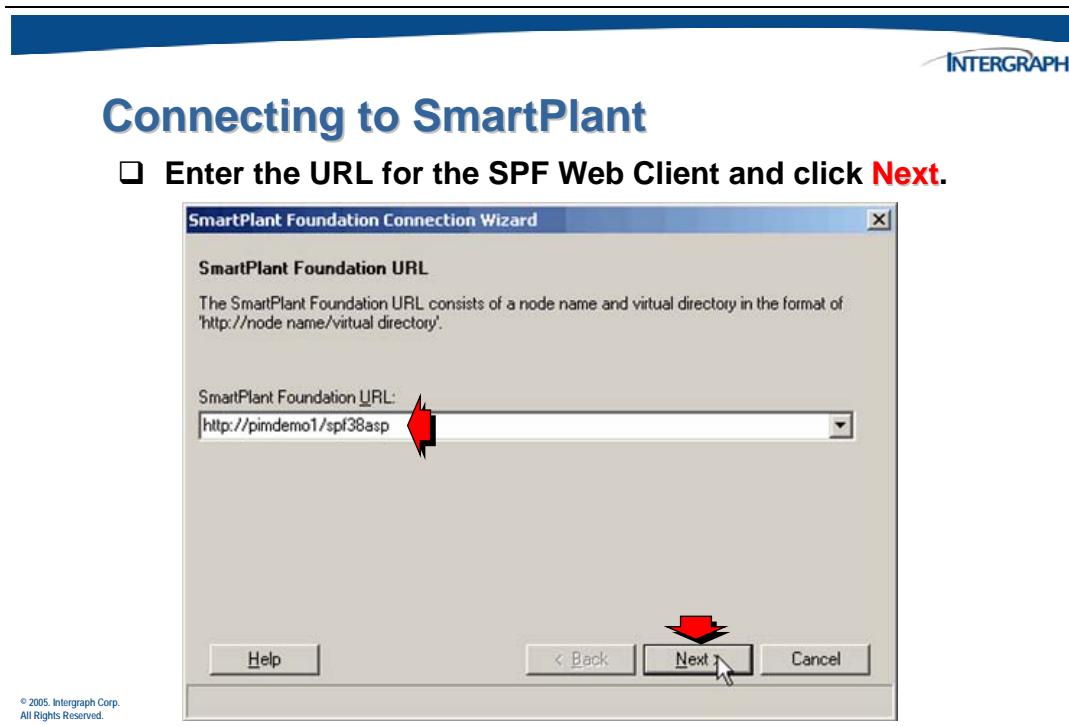
Begin by starting the schema editor and selecting the **Application Metadata** tab.



The *Application Metadata* dialog will display.



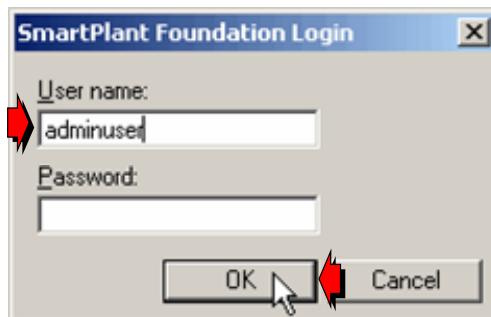
The *SmartPlant Foundation Connection Wizard* will be displayed.



In the *SmartPlant Foundation URL* field, type the node name and virtual directory of the SmartPlant Foundation database with which you want to connect. Use the following format: http://<SPFServer>/<VirtualDirectory>. For example, **http://pimdemo1/spf38asp**.

Connecting to SmartPlant

- Enter a valid SPF **User name** and **Password** to log on to the SPF server from the schema editor.



© 2005, Intergraph Corp.
All Rights Reserved.

For the next step, you will need to know the SmartPlant Foundation plant configuration that you will use for mapping.

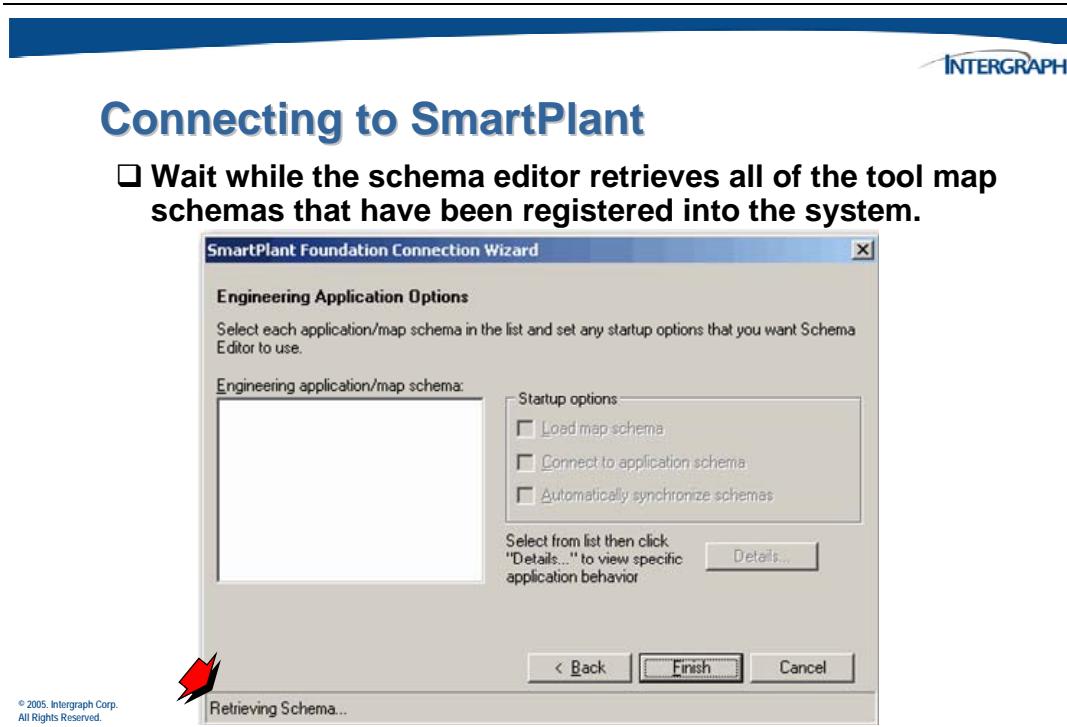
Connecting to SmartPlant

- Select the plant configuration to be used for the schema mapping and click **Next**.



© 2005, Intergraph Corp.
All Rights Reserved.

The Schema Editor will go and find the SmartPlant schema and all **registered** tool map schemas.



The list control on this form will have an entry for each ToolSchema object in the parsed schema and shows the Names from the Tool and ToolSchema objects (presuming the Tool is registered for the plant).

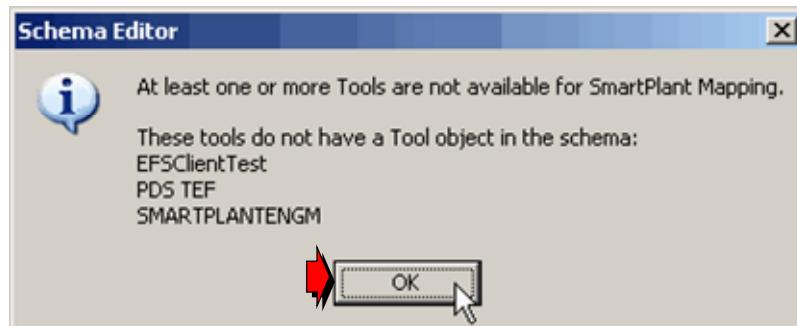
When this form is displayed, a check is made for each tool registered to the plant. This check determines whether there is a Tool object for the registered tool and whether there is at least one ToolSchema object for that tool in the schema.

Any problems detected during this check are reported back to the user. If the software finds any tool map schemas that have not been registered (that is the tool has not been registered) an informational dialog will display.



Connecting to SmartPlant

- When a Schema Editor informational window displays, click OK to continue.



© 2005, Intergraph Corp.
All Rights Reserved.

In the above example, the *EFSClientTest* tool has registered with SmartPlant for this plant but no Tool object exists for it. A list of available tool map schemas will be displayed.



Connecting to SmartPlant

- Select the tool map schema to be loaded and click Finish.



© 2005, Intergraph Corp.
All Rights Reserved.

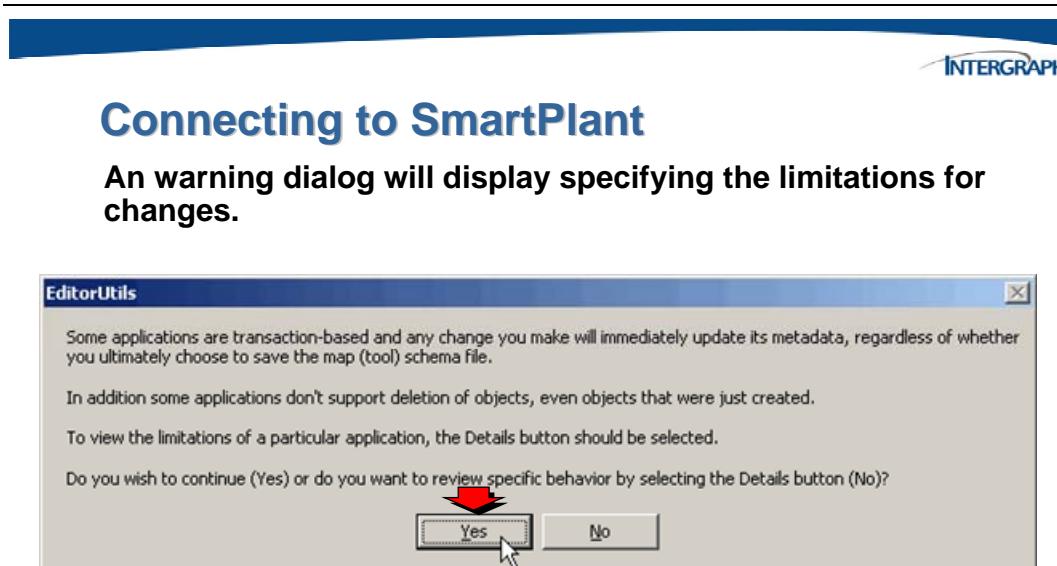
Selecting a Tool/ToolSchema entry from the list control enables one (or more) of the toggle boxes on the right of the dialog.

The toggle boxes are used to specify what actions are to be performed for that Tool/ToolSchema combination.

Selecting the “Load map schema” toggle will trigger the parsing of the tool map schema file. If this is the only toggle selected, then the metadata adapter will not be called to Connect (and therefore no map schema based on the application metadata will be generated and no synchronization will occur). This mode of operation is referred to as “disconnected”.

If the user selects the “*Load map schema*” button, then the “*Connect to application schema*” button will be automatically selected. Likewise, unselecting the “*Connect to application schema*” button will also unselect the “*Load map schema*” button

If the “*Connect to application schema*” radio button is selected, then not only will the tool map schema file be parsed but the metadata adapter will also be called to Connect to its metadata data store and generate a tool map schema based on its application metadata.

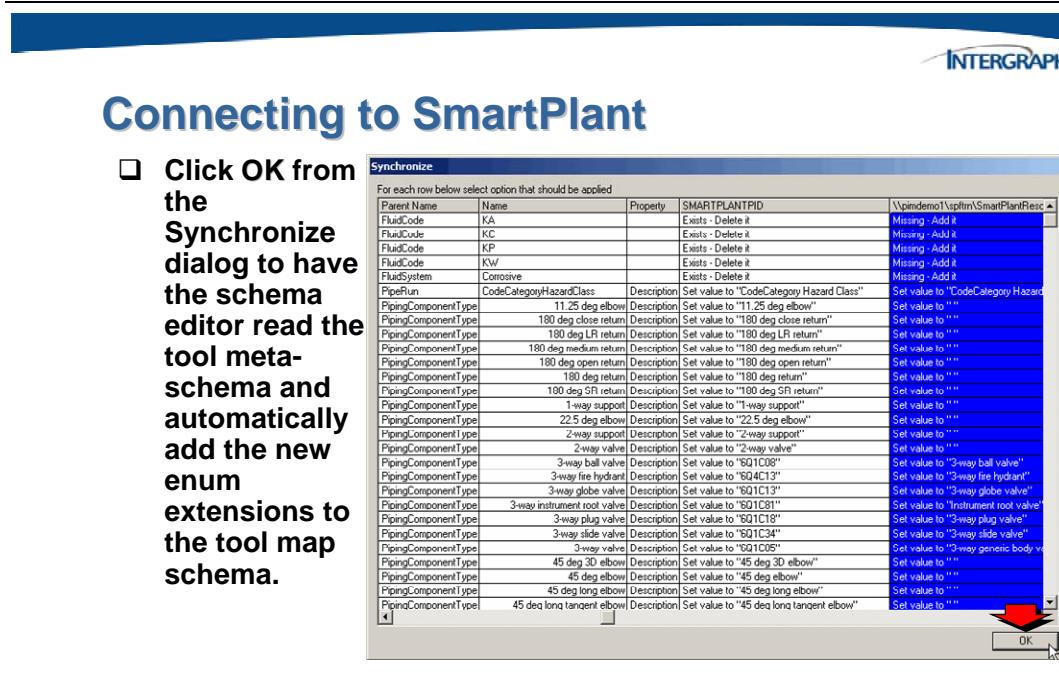


© 2005, Intergraph Corp.
All Rights Reserved.

After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema.

The differences between the two map schemas are displayed in the **Synchronize** form. For each difference the user must select (each one has a system-determined pre-selection) which synchronization action to perform

Finally, a *Synchronize* dialog will display.



© 2005, Intergraph Corp.
All Rights Reserved.

Synchronization Form Columns

- Class** – the class definition (ClassDef) for the object that is different
- Name** – the Name of the object that is different.
- Property** – if the object exists in both map containers, but one or more properties are different, than the name of a property that is different will display in this column
- Generated map schema** – the option to take if the generated map schema is to take precedence
- Parsed map schema** – the option to take if the parsed map schema is to take precedence



Connecting to SmartPlant

Zoomed in view of the Synchronize dialog

Synchronize				
For each row below select option that should be applied				
Parent Name	Name	Property	SMARTPLANTPID	
FluidCode	KA	Exists - Delete it	Missing - Add it	
FluidCode	KC	Exists - Delete it	Missing - Add it	
FluidCode	KP	Exists - Delete it	Missing - Add it	
FluidCode	KW	Exists - Delete it	Missing - Add it	
FluidSystem	Corrosive	Exists - Delete it	Missing - Add it	
PipeRun	CodeCategoryHazardClass	Description	Set value to "CodeCategory Hazard Class"	Set value to "CodeCategory Hazard Class"

7.5.2 Verifying the SmartPlant Schema

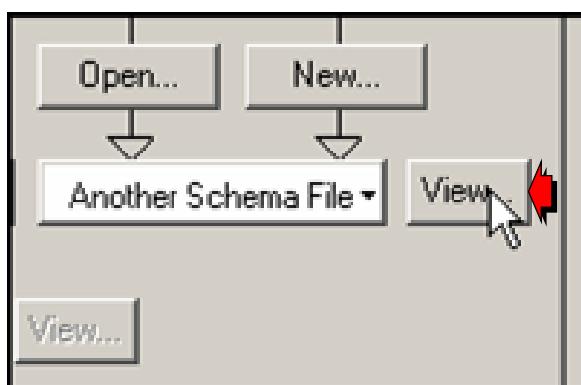
After the Schema Editor synchronizes the SmartPlant schema with the tool map schema, you can view the contents of the SmartPlant schema to verify that the enumerated list extensions that were added to the tool meta schema are not present in the SmartPlant schema.

Since multiple schema files have been opened, you will use the **Another Schema File > View** button instead of the **Schema File > View** button above and to the left of this area of the Workflows dialog.

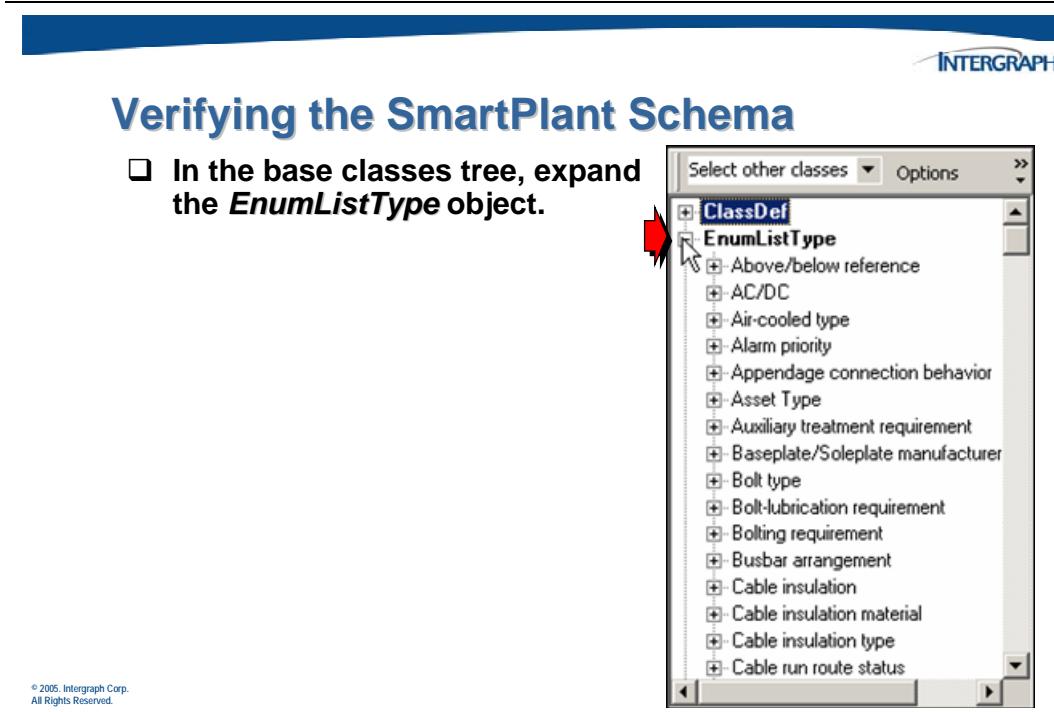


Verifying the SmartPlant Schema

- In the **Workflows** dialog box, select the **View** button (next to **Another Schema File**) to view (and edit) the schema files.



In the *Data Dictionary Manager*, extensions to the *Fluidsystems Select List* were made. Make sure that these values are not in the SmartPlant schema before you start any modifications (additions).

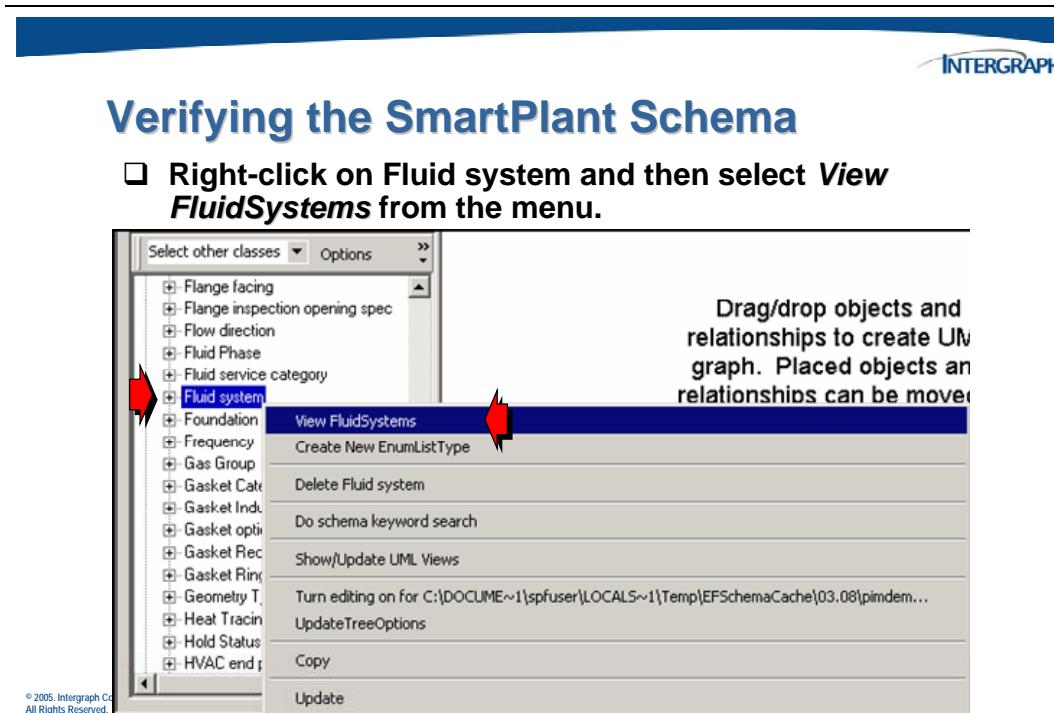


Verifying the SmartPlant Schema

- In the base classes tree, expand the *EnumListType* object.

© 2005. Intergraph Corp.
All Rights Reserved.

Scroll down in the tree until you locate the existing *Fluid system* EnumList.

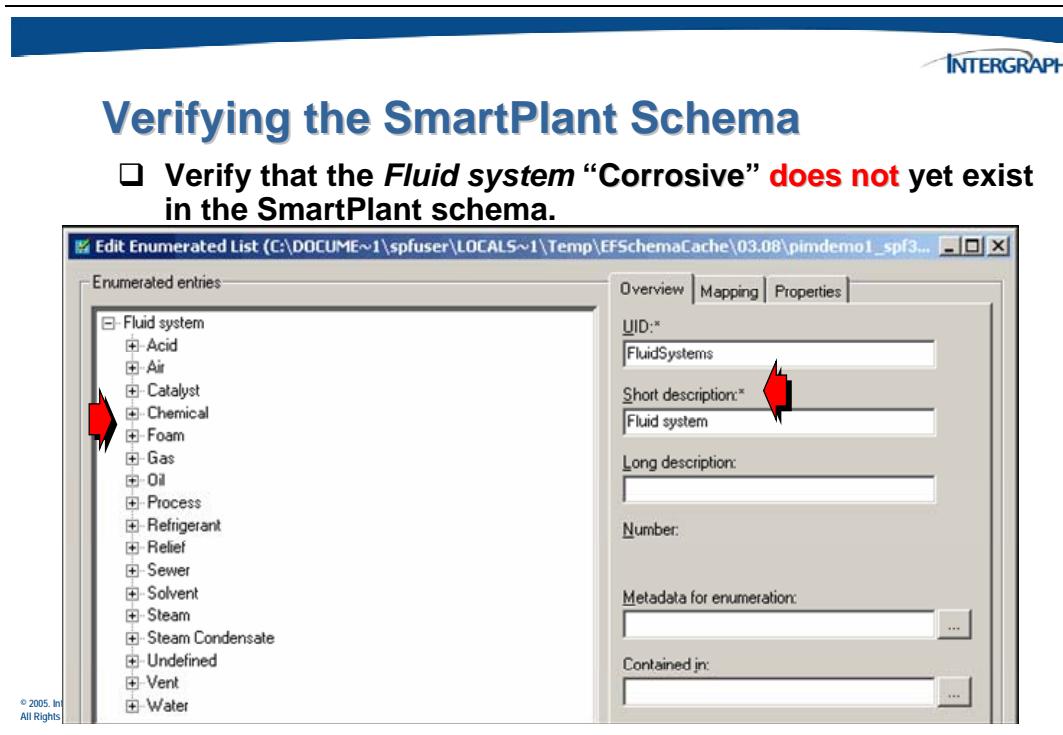


Verifying the SmartPlant Schema

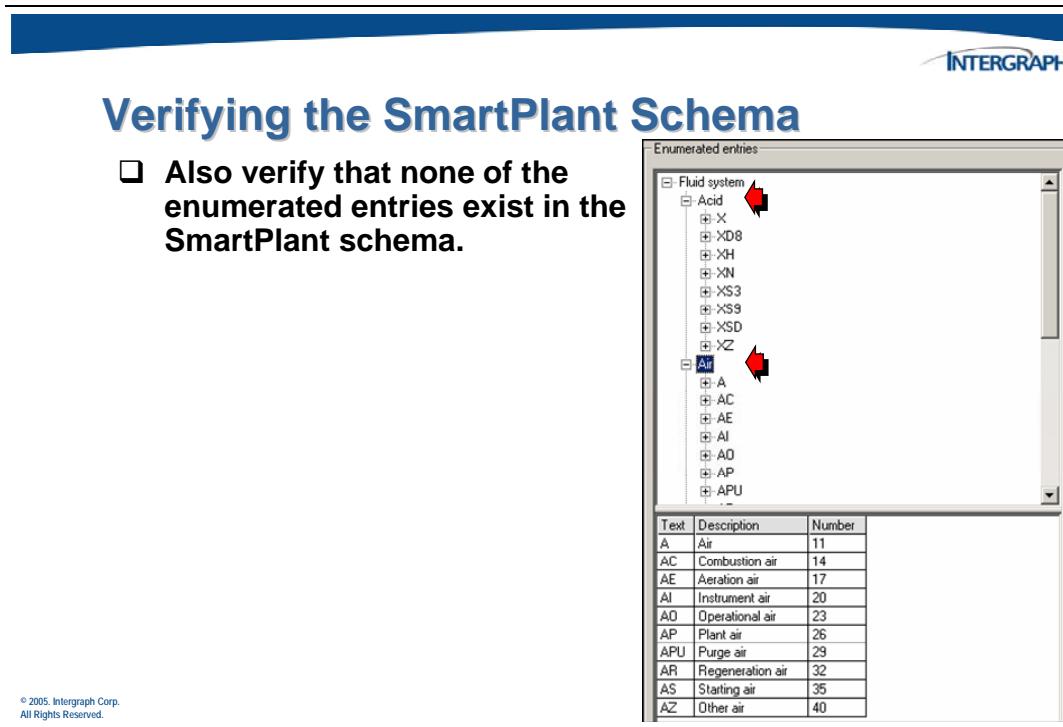
- Right-click on Fluid system and then select *View FluidSystems* from the menu.

© 2005. Intergraph Corp.
All Rights Reserved.

The *Edit Enumerated List* dialog will be displayed.



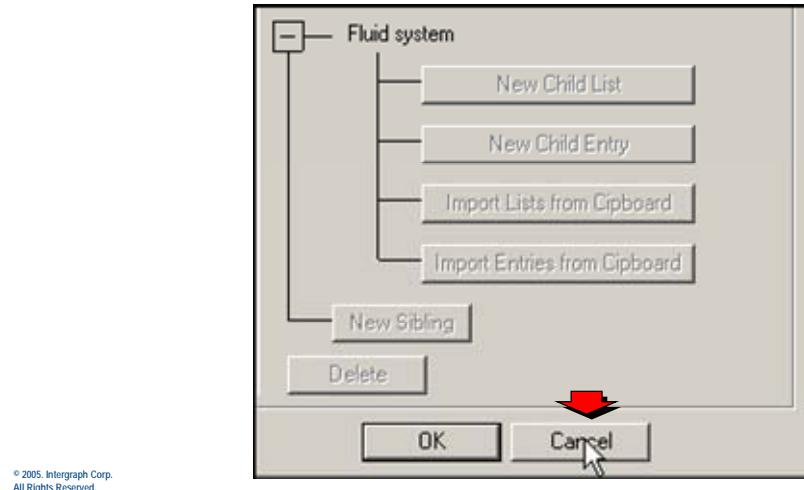
Expand the “child” enumerated lists to view the existing enumerated entries.





Verifying the SmartPlant Schema

- Cancel out of the edit dialog since no changes or additions were made.



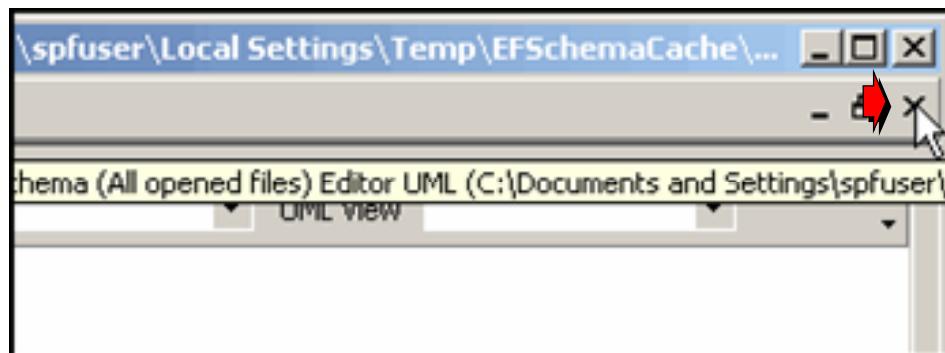
© 2005, Intergraph Corp.
All Rights Reserved.

Once the edit dialog has been closed, close out of the view window.



Verifying the SmartPlant Schema

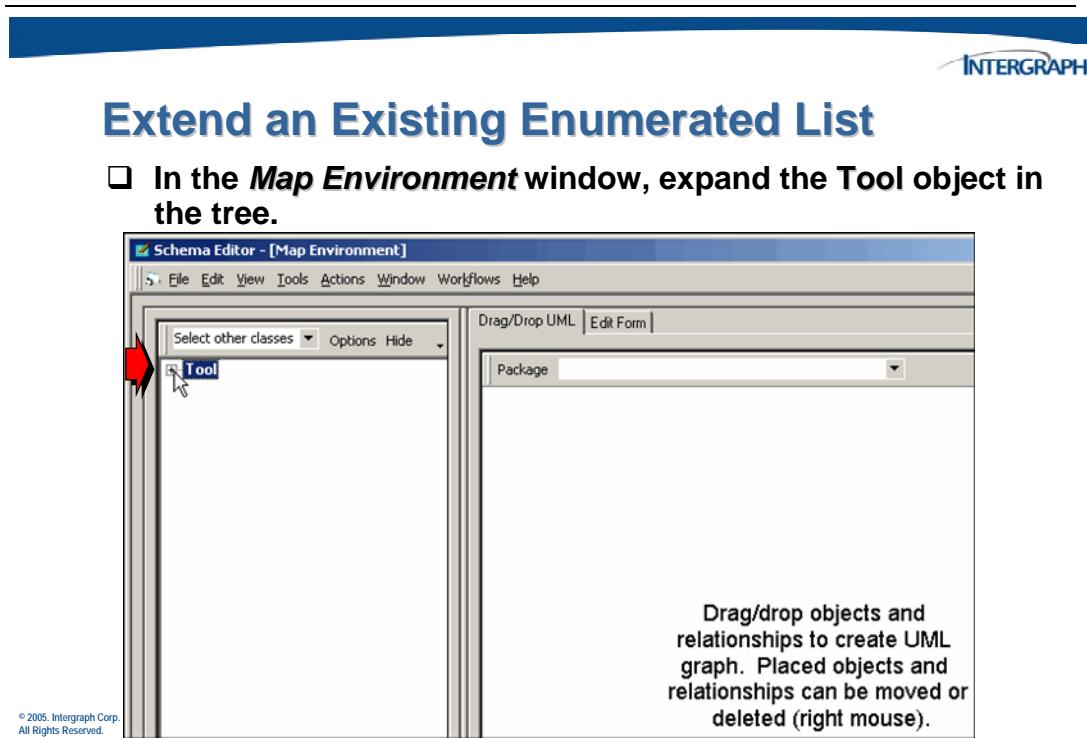
- Click the X to close out of the view window.



© 2005, Intergraph Corp.
All Rights Reserved.

7.5.3 Extending an Existing Enumerated List

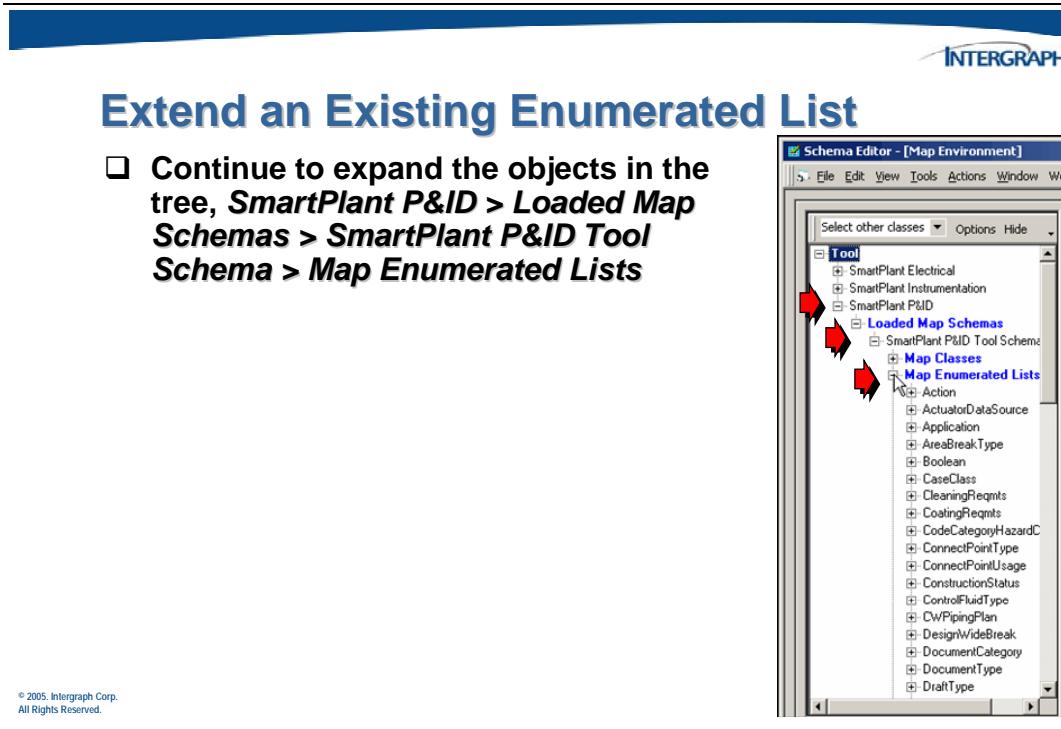
When the view window is closed, the *Map Environment* window will still be displayed. Use the Map Environment window to extend an existing enumerated list in the SmartPlant schema as well as perform the mapping to the proper tool map schema.



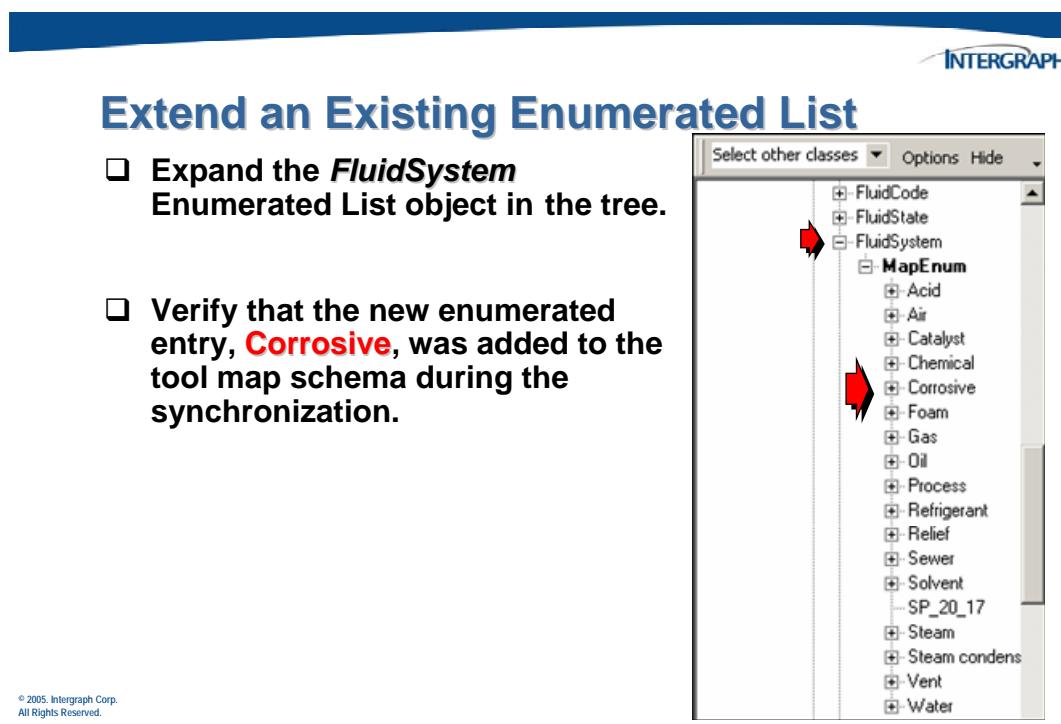
The Map Environment Tree displays each application (Tool object) for which the “Load map file” option was selected.

When an application is expanded, shows the tool map schema (ToolSchema object) selected for that application (only one map schema per application can be used). When the tool map schema is expanded, it shows the map classes, map enumerated lists, map unit of measure lists and map relationship definitions for that tool map schema

This will display the available tool map schemas in the tree view.



Locate the existing *FluidSystem MapEnum* list in the tree.



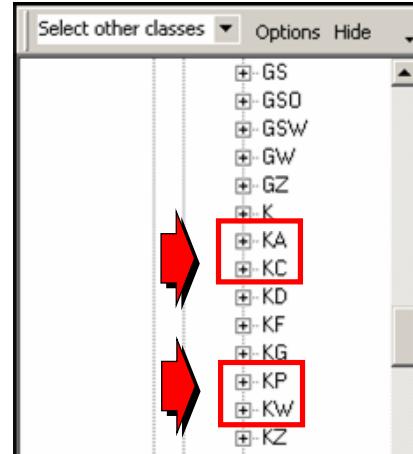
The reason that the entry for **Corrosive** is displayed is because the software read the list entries that were created in the *Data Dictionary Manager* and then automatically added these entries to the tool map schema during the earlier synchronization step. This occurred when the schemas were opened by the connect process.

Scroll down in the tree and locate the *FluidCode MapEnum* list.

Extend an Existing Enumerated List

- Expand the *FluidCode Enumerated List* object in the tree.**

- Verify that the new enumerated entries were added to the tool map schema during the synchronization.**

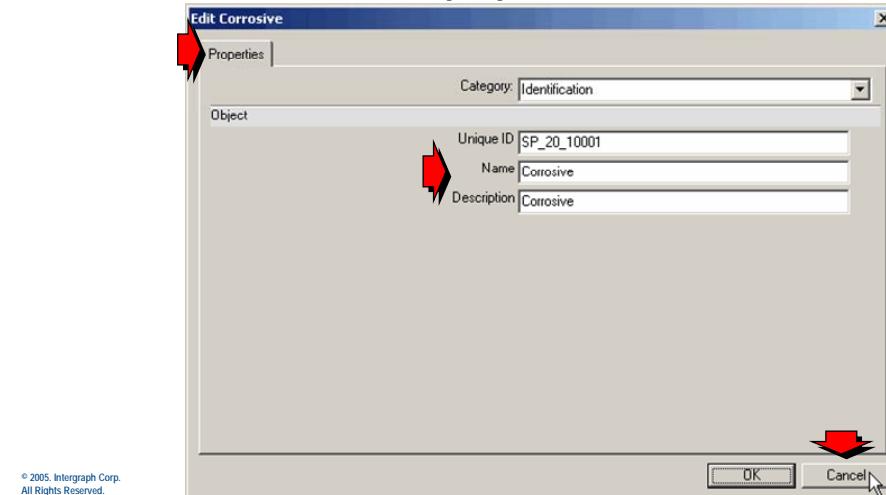


Again, the reason that these entries are displayed is because the software read these during synchronization.



Extend an Existing Enumerated List

- ❑ Right-click on **Corrosive** in the tree and select the **Edit** command to view it's properties.

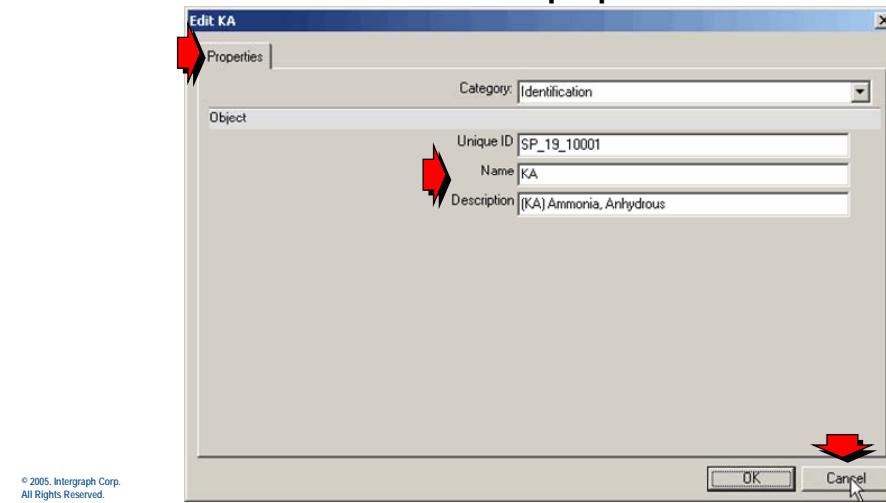


The information displayed in the properties dialogs can be correlated with the information stored in the *Enumerations* and *Codelists* Oracle tables viewed earlier.



Extend an Existing Enumerated List

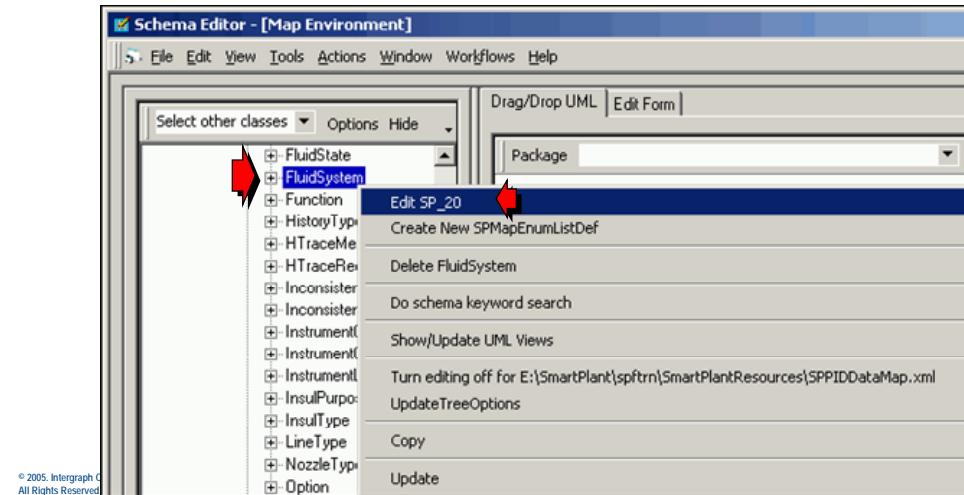
- ❑ Right-click on **KA** (for example) in the tree and select the **Edit** command to view it's properties





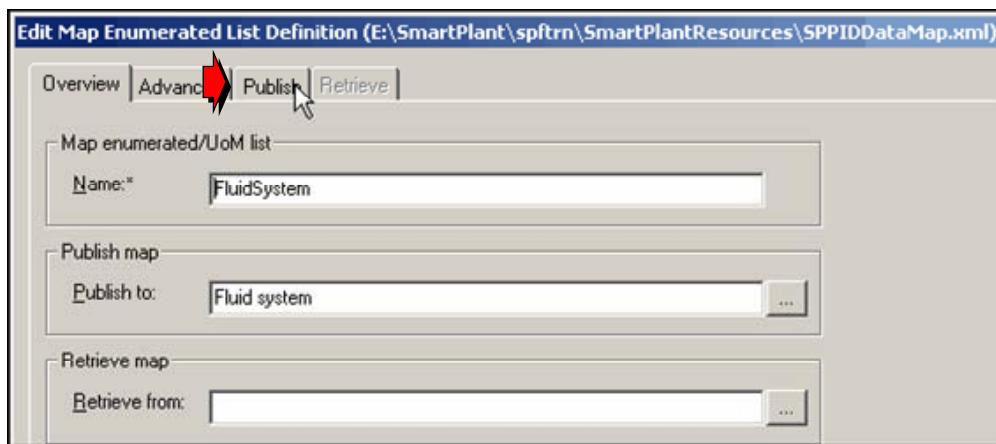
Extend an Existing Enumerated List

- Choose the **FluidSystem Map Enumerated List** and select **Edit** from the pop-up menu.



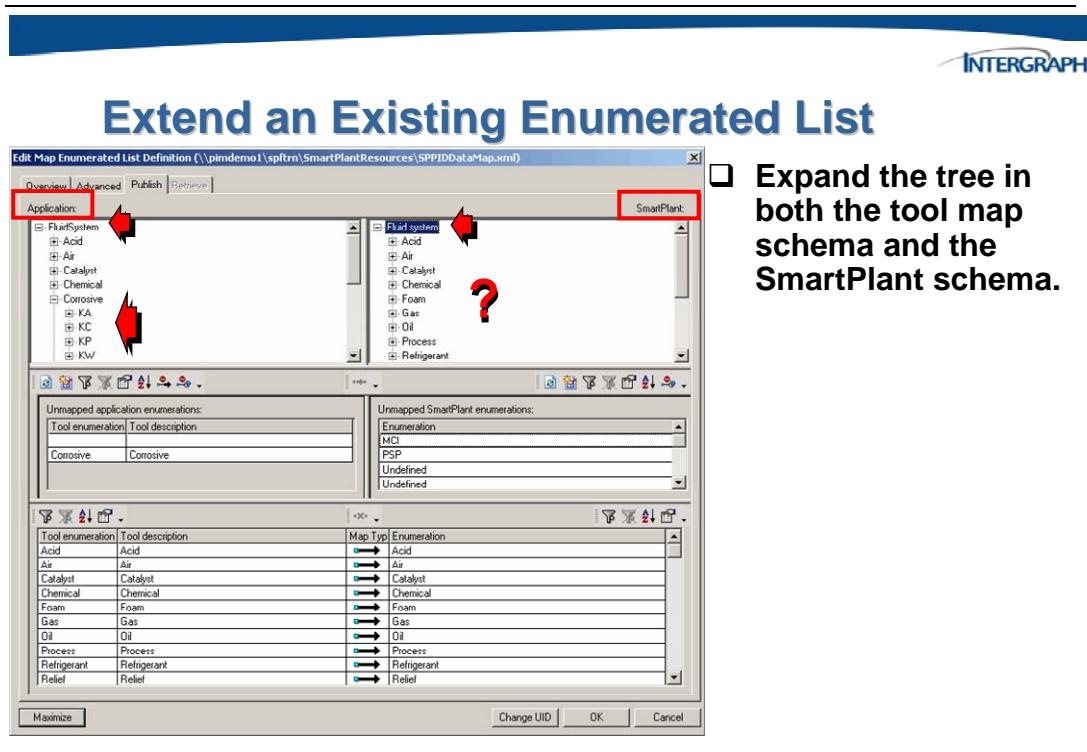
Extend an Existing Enumerated List

- From the **Edit Map Enumerated List Definition** dialog, click on the **Publish** tab.



The edit form for map class definitions contains the following tabs:

- Overview** – shows the *Name*, the mapping option and the publish and retrieve view definitions
- Publish** – shows the publish information for the map class definition
- Retrieve** – shows the retrieve information for the map class definition

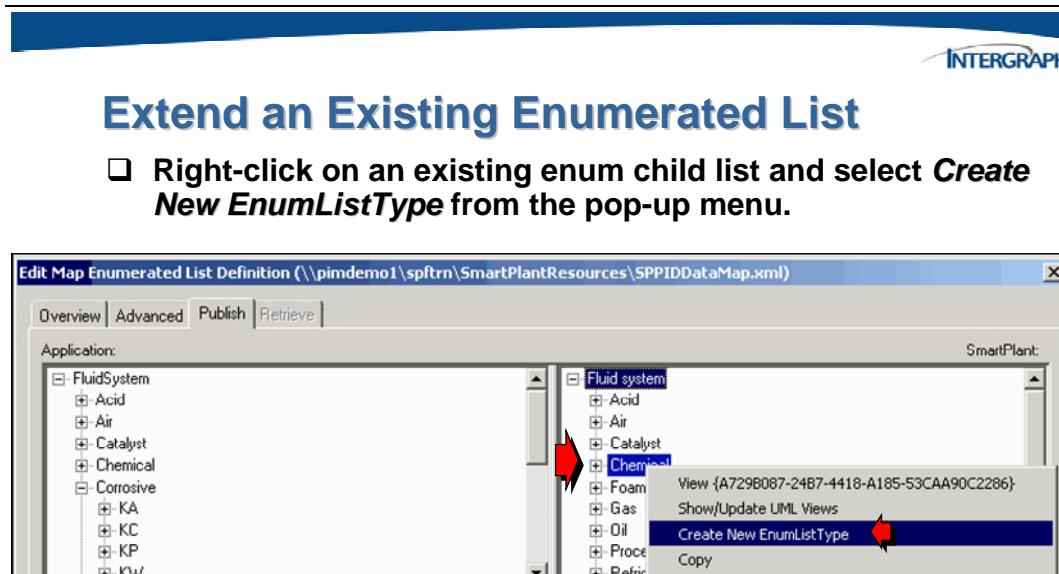


The Publish dialog has four controls:

- Upper left tree control shows tool map schema information
- Upper right tree control shows SmartPlant information
- Middle control shows unmapped map properties and unmapped SmartPlant view properties
- Lower control shows map properties that have been mapped and the SmartPlant properties to which they map

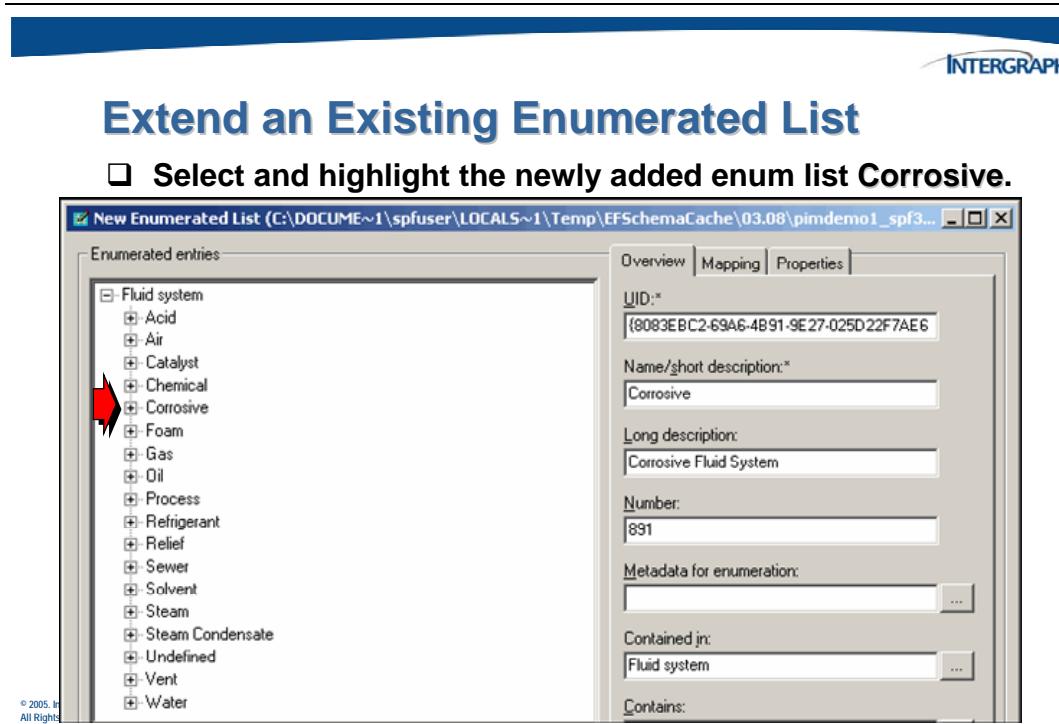
- Expand the tree in both the tool map schema and the SmartPlant schema.**

Note that the value for **Corrosive** does not exist in the SmartPlant schema, so it must be added.

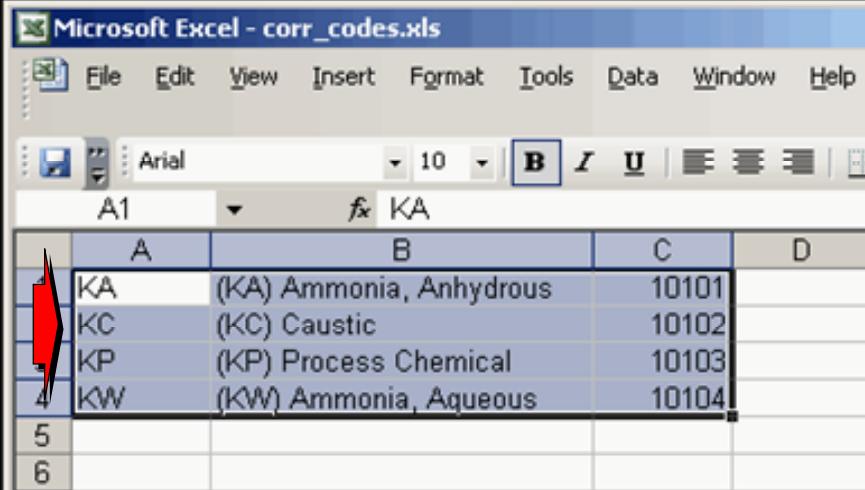


© 2005, Intergraph Corp.
All Rights Reserved.

Enter the necessary information to create a new child enumerated list (within Fluid system).



New enumerated entries can be first entered in an Excel spreadsheet.



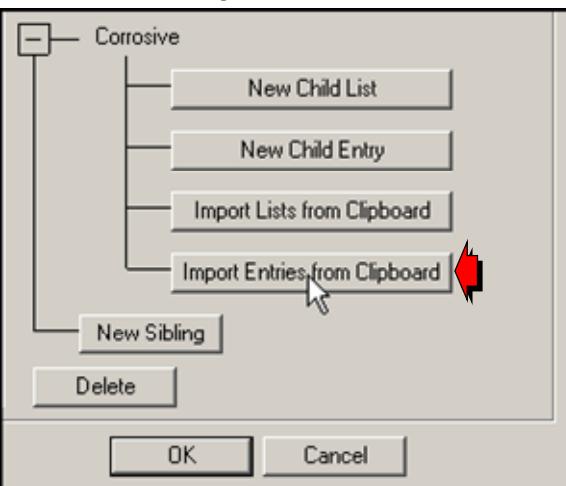
Extend an Existing Enumerated List

Copy (control-c) the new enum entries to the clipboard.

A	B	C	D
KA	(KA) Ammonia, Anhydrous	10101	
KC	(KC) Caustic	10102	
KP	(KP) Process Chemical	10103	
KW	(KW) Ammonia, Aqueous	10104	
5			
6			

© 2005, Intergraph Corp.
All Rights Reserved.

Minimize Excel and return back to the *New Enumerated List* dialog.



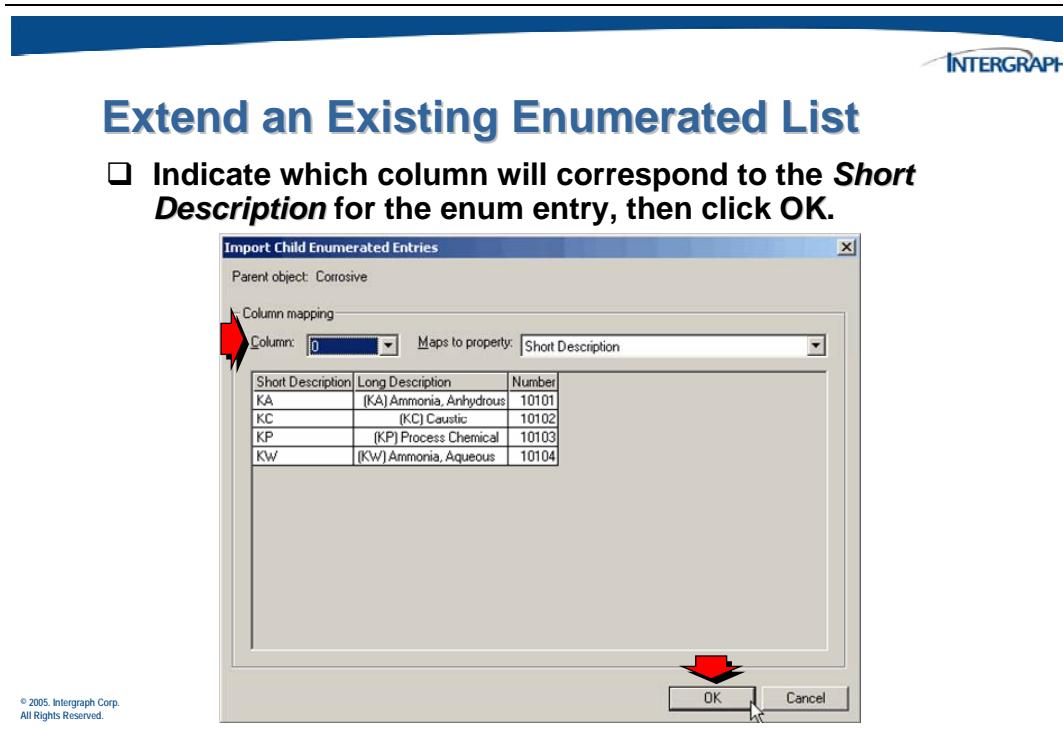
Extend an Existing Enumerated List

Select the Import Entries from Clipboard button on the bottom of the *Edit Map Enumerated List Definition* dialog.

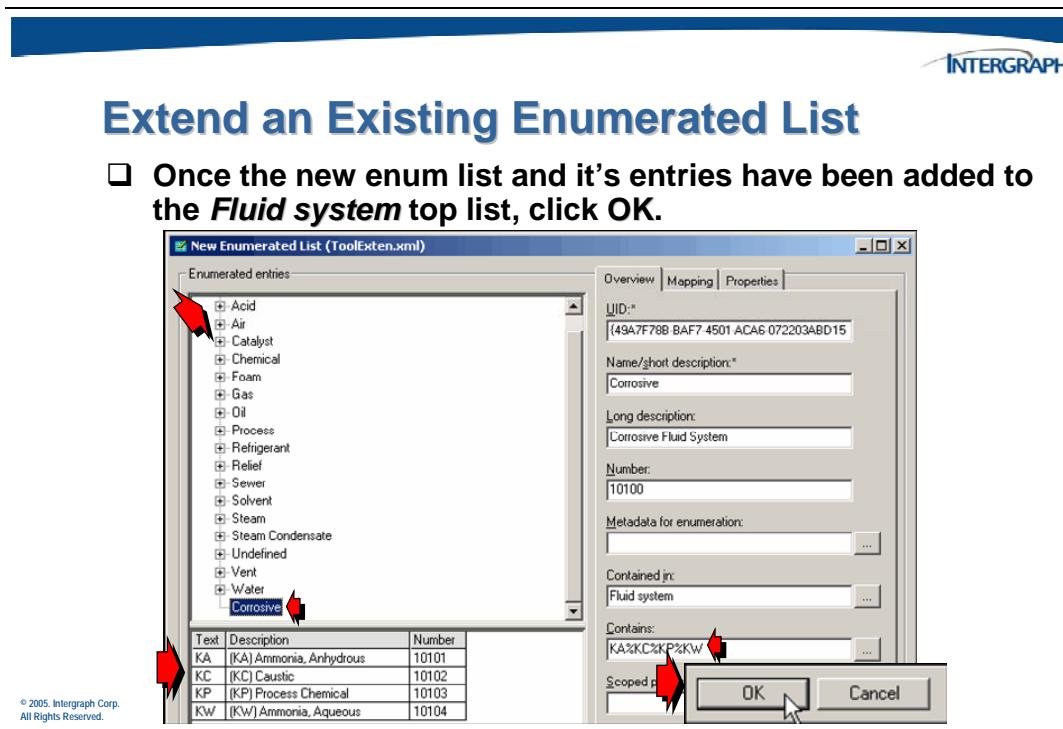
- Corrosive
 -
 -
 -
 -
 -
-

© 2005, Intergraph Corp.
All Rights Reserved.

The Schema Editor will prompt for orientation information in order to import the entries.



Verify that the new **Fluid system - Corrosive** values have been imported correctly.



7.5.4 Mapping an Existing Enumerated List

After you define the new enumerated list in the SmartPlant schema, you will do the mapping for publish and retrieve using the Map Environment.

Listed below are the mapping icons and their functions.



Define Schema Mapping

Click **To**



Refresh – refreshes the display of the map properties



Create application property – displays the create/edit form for an application property (SPMapPropertyDef) and allows user to add an application property to the map schema



Filter application properties – allows the user to specify criteria that filters the set of displayed application properties in the unmapped (middle) control



Unfilter application properties – allows the user to unfilter the application properties

© 2005, Intergraph Corp.
All Rights Reserved.



Define Schema Mapping

Click **To**



Add/delete displayed properties – allows user to specify properties (columns) to be displayed in the unmapped control



Sort map properties – allows user to specify sort criteria to be used for ordering unmapped application properties



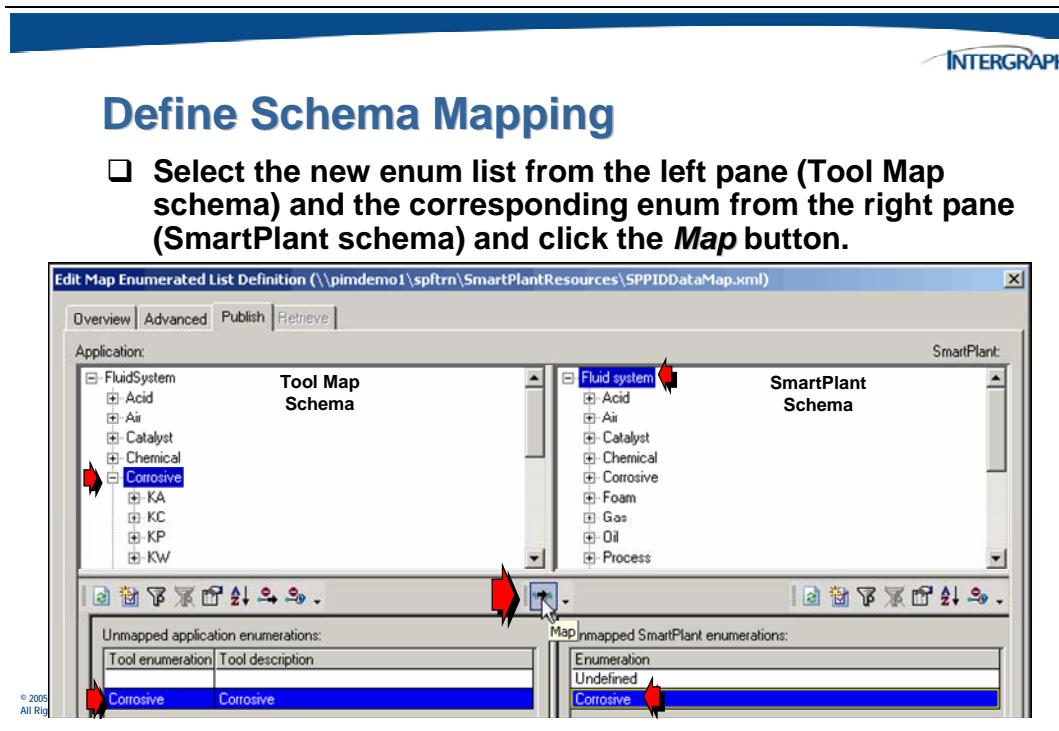
Ignore application properties – allows user to identify application properties to ignore



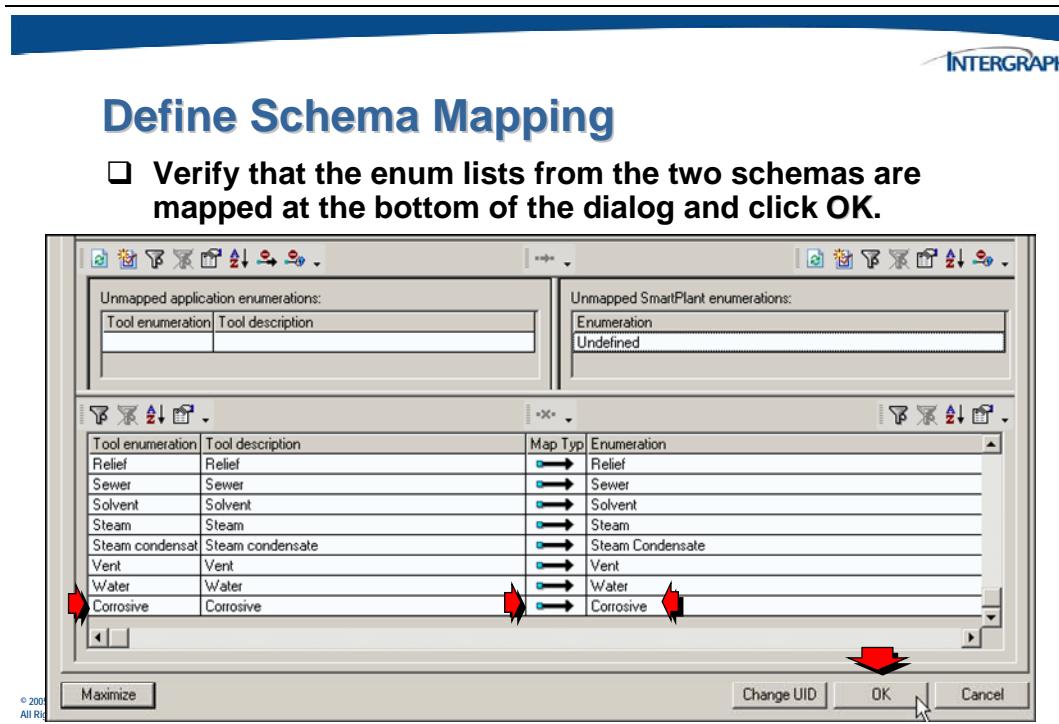
Set unmapped map properties – allows user to identify map properties that won't be mapped

© 2005, Intergraph Corp.
All Rights Reserved.

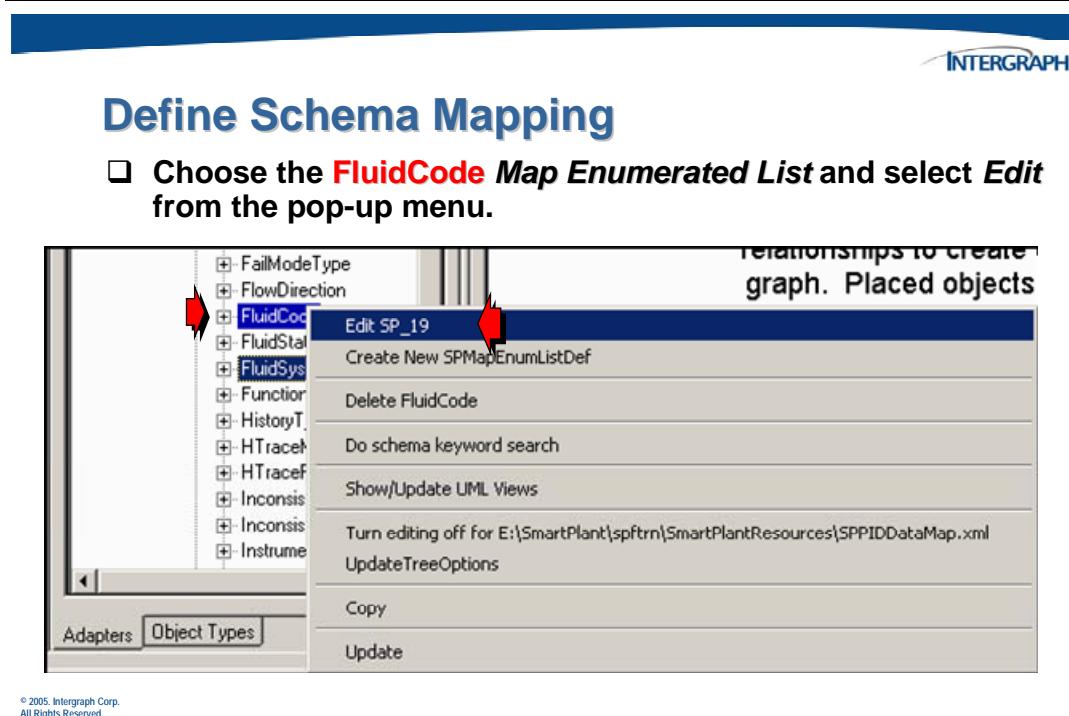
Clicking a map property (enum list/value) in the tree control selects it from either the unmapped or mapped control.



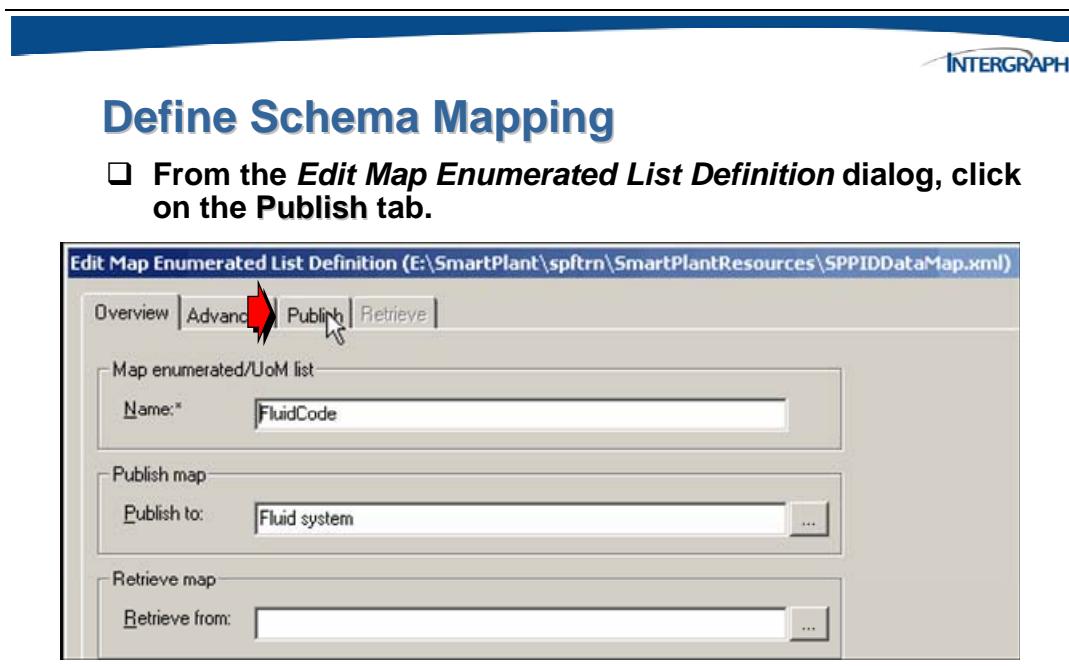
The **Map** button is used to map an unmapped application property (enum list/value) to a unmapped SmartPlant property (enum list/value) definition.



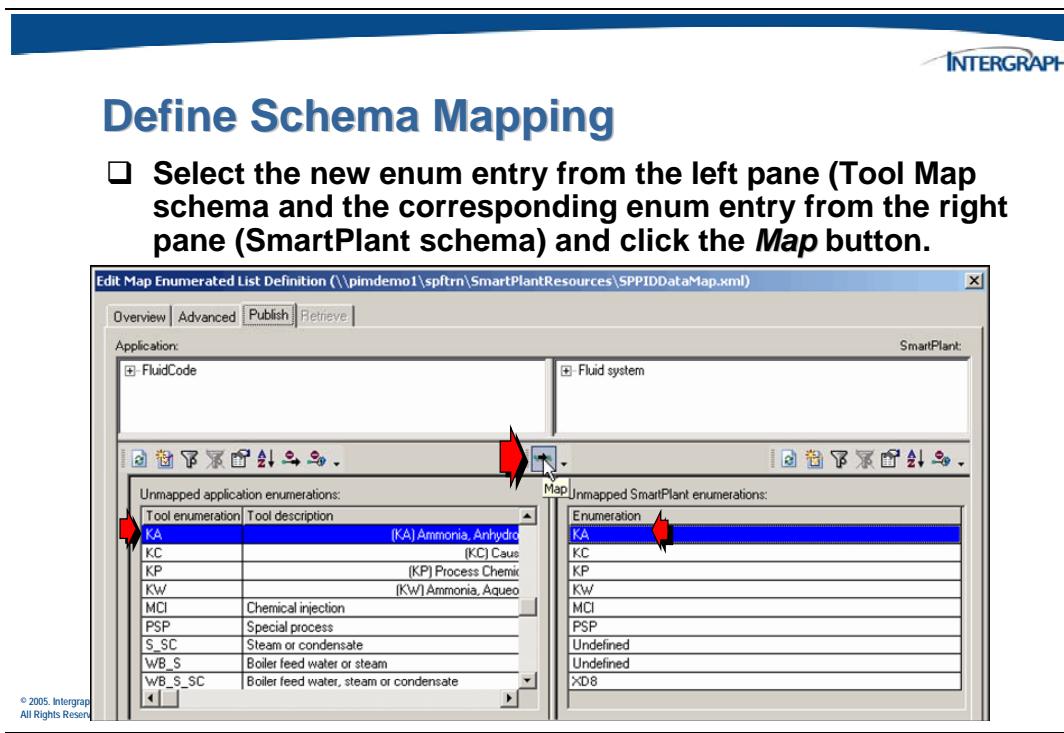
Since the new enumerated values were originally created in two different *Select Lists* in the *Data Dictionary Manager*, both enumerated lists will have to be mapped to **Fluid system** in the SmartPlant schema.



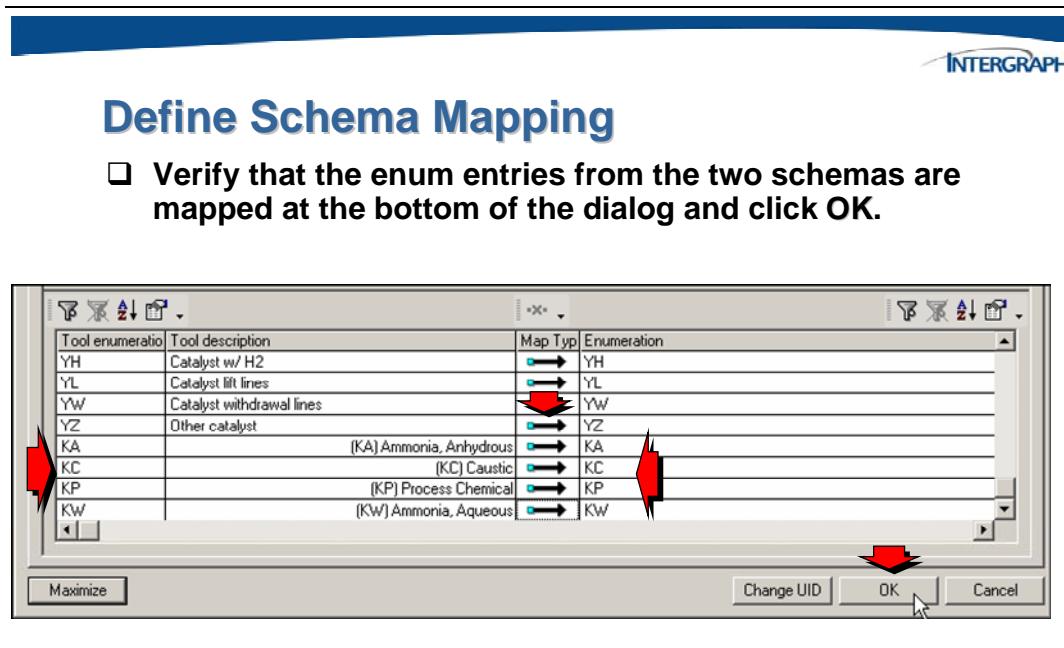
Repeat the mapping for the **FluidCode** enum entries.



When the *Publish* form displays, select the enumerated entries to be mapped.

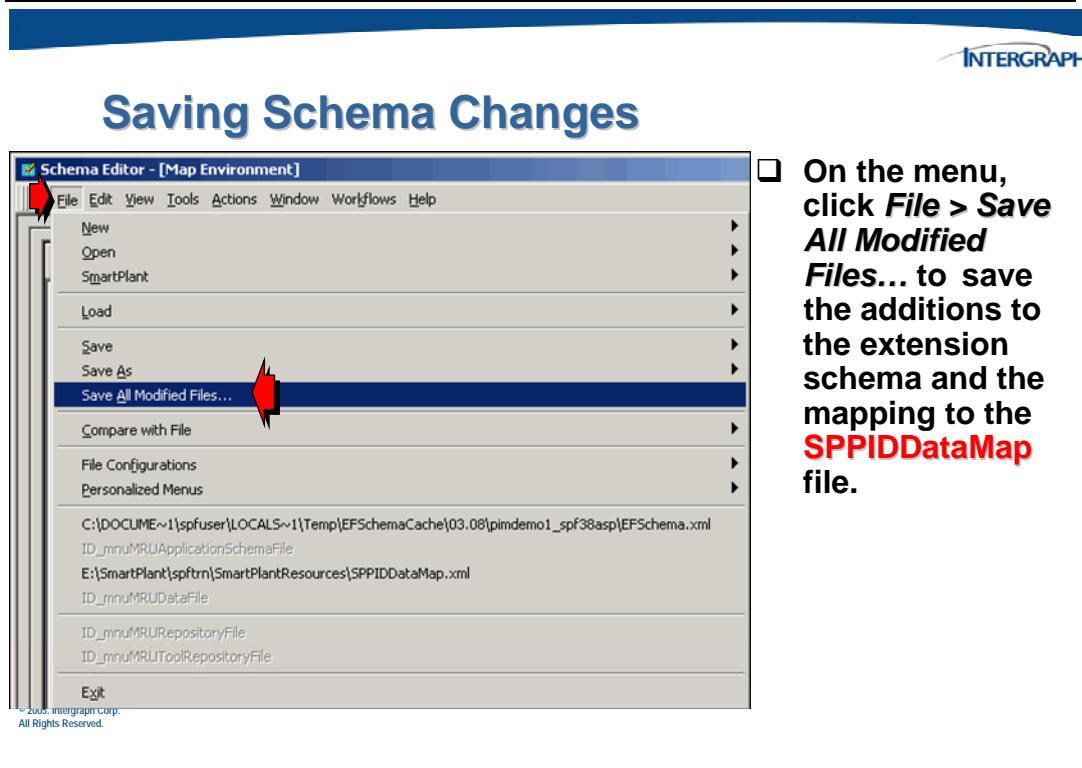


The **Unmap** button deletes the mapping relationship for each of the selected rows from the control.



7.5.5 Saving Schema Changes

At this point, the additions to the SmartPlant schema and the mapping information is stored in memory will need to be saved to the respective xml files.



You will be prompted to save the changes to the tool map schema.



Saving Schema Changes

- Verify the tool map schema name for the mapping changes and choose **Yes**.

© 2005, Intergraph Corp.
All Rights Reserved.

You will also be prompted to save the changes to the SmartPlant extension schema.

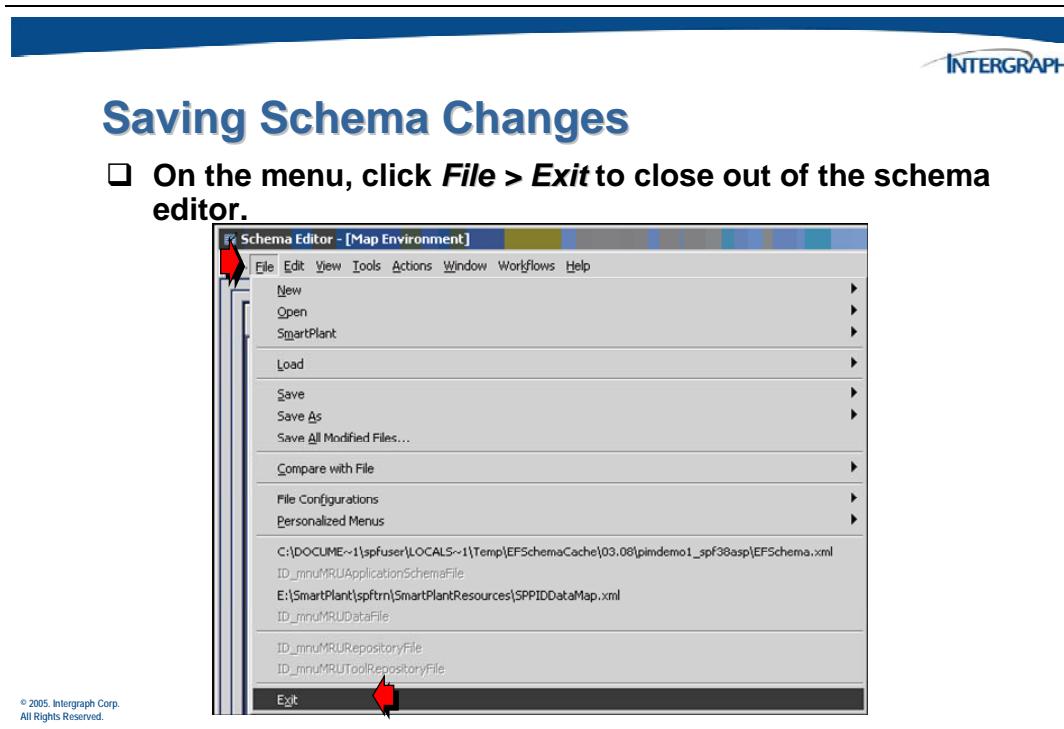


Saving Schema Changes

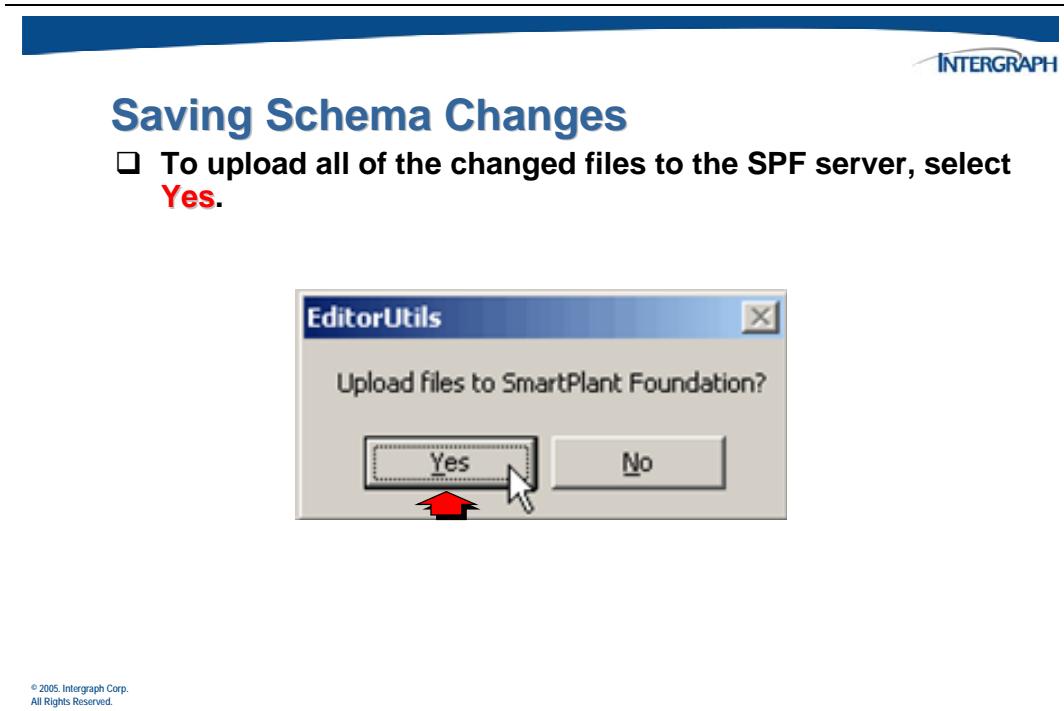
- Verify the extension schema name for the enum additions and choose **Yes**.

© 2005, Intergraph Corp.
All Rights Reserved.

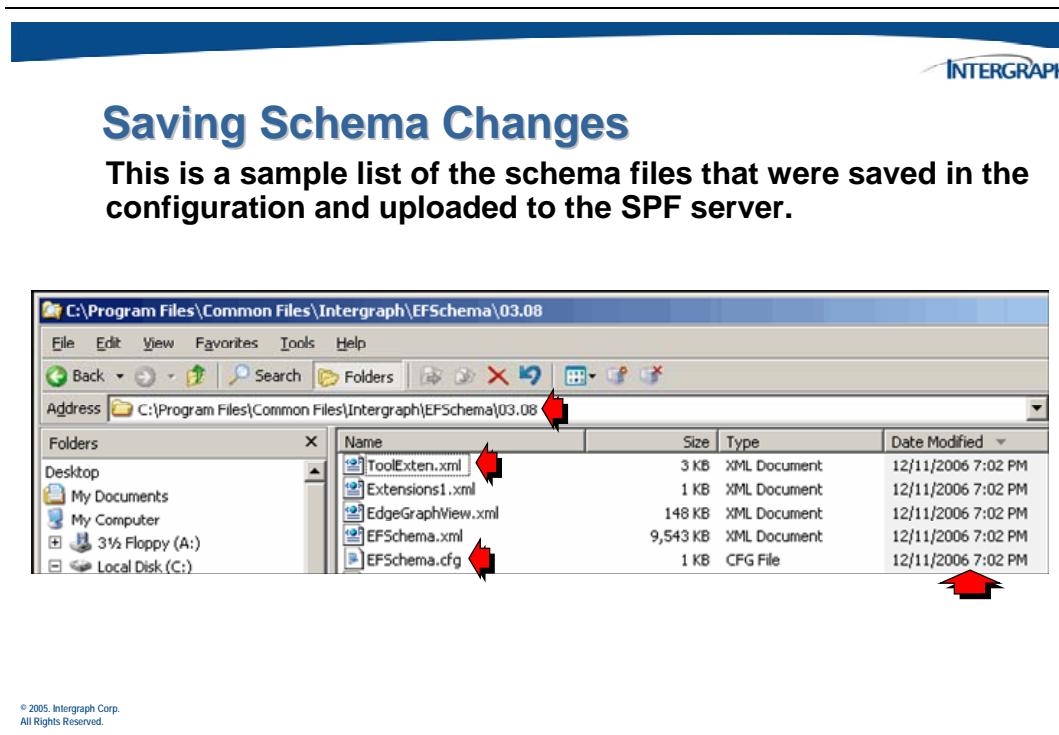
Once all of the schema files have been saved, exit the Schema Editor.



The schema files have been copied to a temporary location in order to perform the additions and mapping. The last step is to upload these files to the schema location on the SmartPlant Foundation server.



All schema files in the EFSchema configuration (EFSchema.cfg) will be copied to the SmartPlant Foundation server.



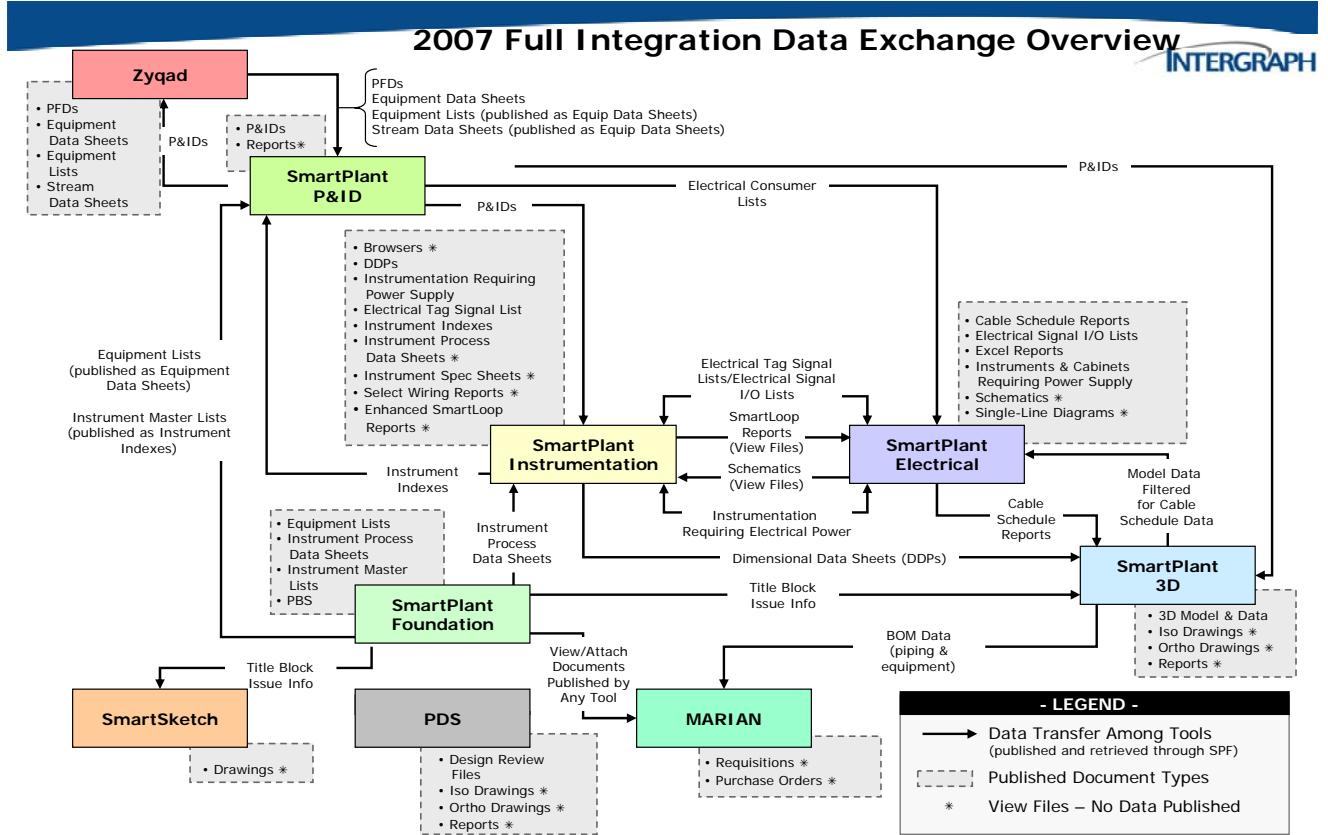
The screenshot shows a Windows File Explorer window with the title bar "C:\Program Files\Common Files\Intergraph\EFSchema\03.08". The address bar displays the same path. The left pane shows a "Folders" tree with icons for Desktop, My Documents, My Computer, 3½ Floppy (A:), and Local Disk (C:). The right pane is a grid view of files with columns for Name, Size, Type, and Date Modified. Red arrows point to several specific files: "ToolExten.xml", "Extensions1.xml", "EdgeGraphView.xml", "EFSchema.xml", and "EFSchema.cfg". All these files are XML documents except for "EFSchema.cfg" which is a CFG File. The date modified for all files is 12/11/2006 7:02 PM, and their sizes range from 1 KB to 9,543 KB.

Name	Size	Type	Date Modified
ToolExten.xml	3 KB	XML Document	12/11/2006 7:02 PM
Extensions1.xml	1 KB	XML Document	12/11/2006 7:02 PM
EdgeGraphView.xml	148 KB	XML Document	12/11/2006 7:02 PM
EFSchema.xml	9,543 KB	XML Document	12/11/2006 7:02 PM
EFSchema.cfg	1 KB	CFG File	12/11/2006 7:02 PM

© 2005, Intergraph Corp.
All Rights Reserved.

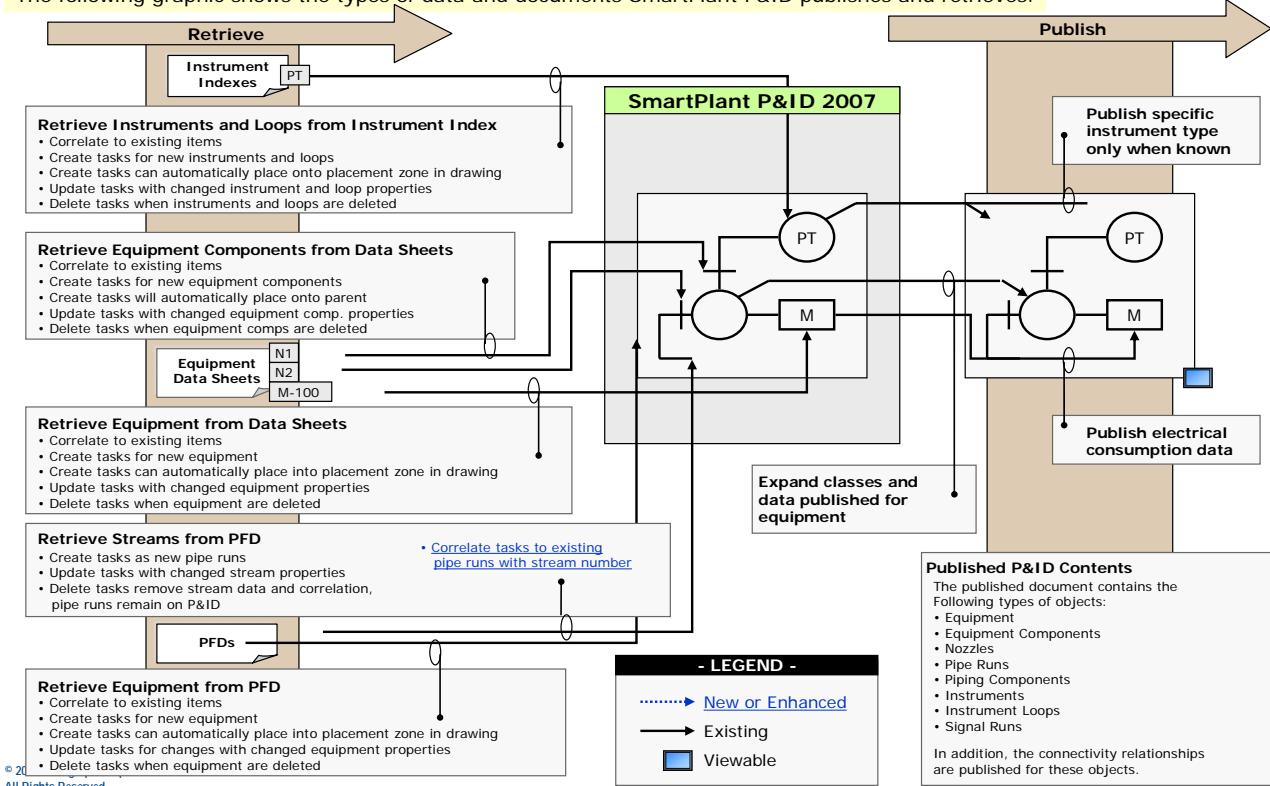
7.6 Published and Retrieved Document Types

Listed in this section are some diagrams to show the integration data exchange. The first diagram shows an overview for full integration. There are also data exchange diagrams that describe the out-of-the-box data exchange capabilities for each of the tools.



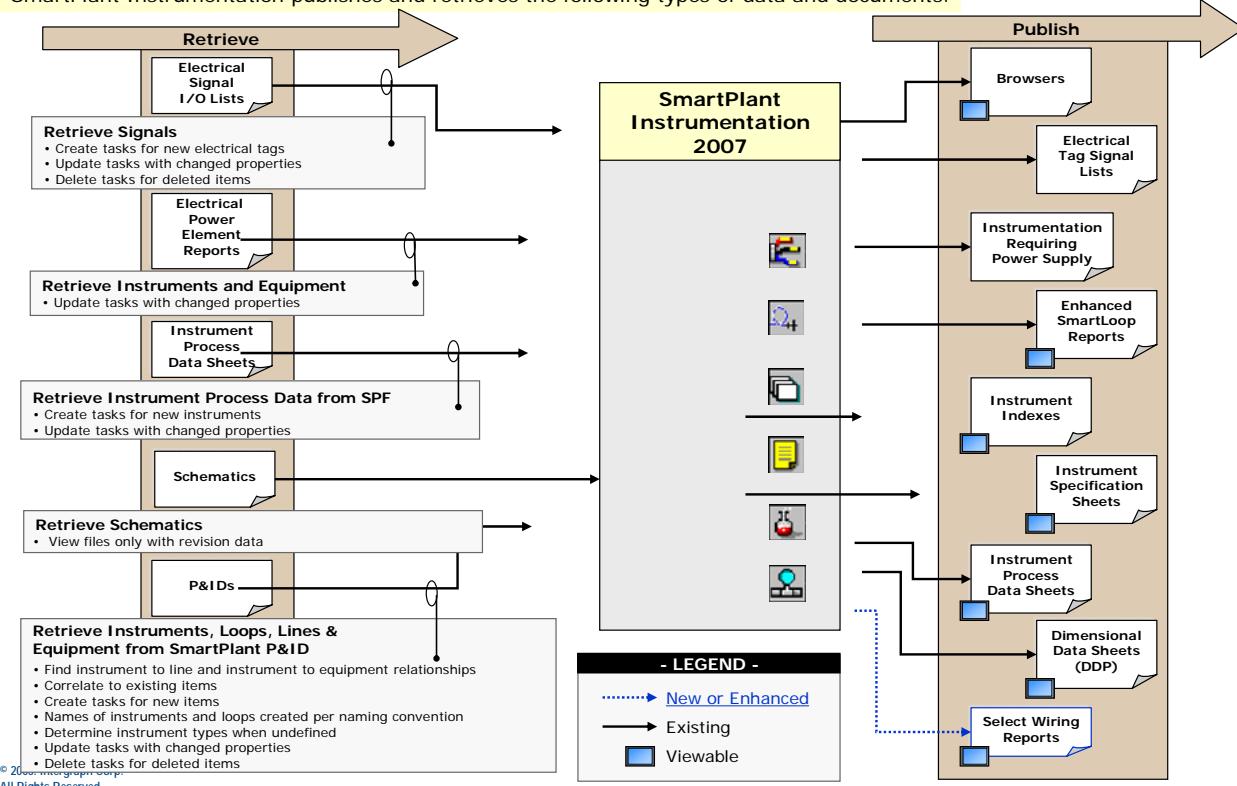
SmartPlant P&ID Data Exchange Example

The following graphic shows the types of data and documents SmartPlant P&ID publishes and retrieves.

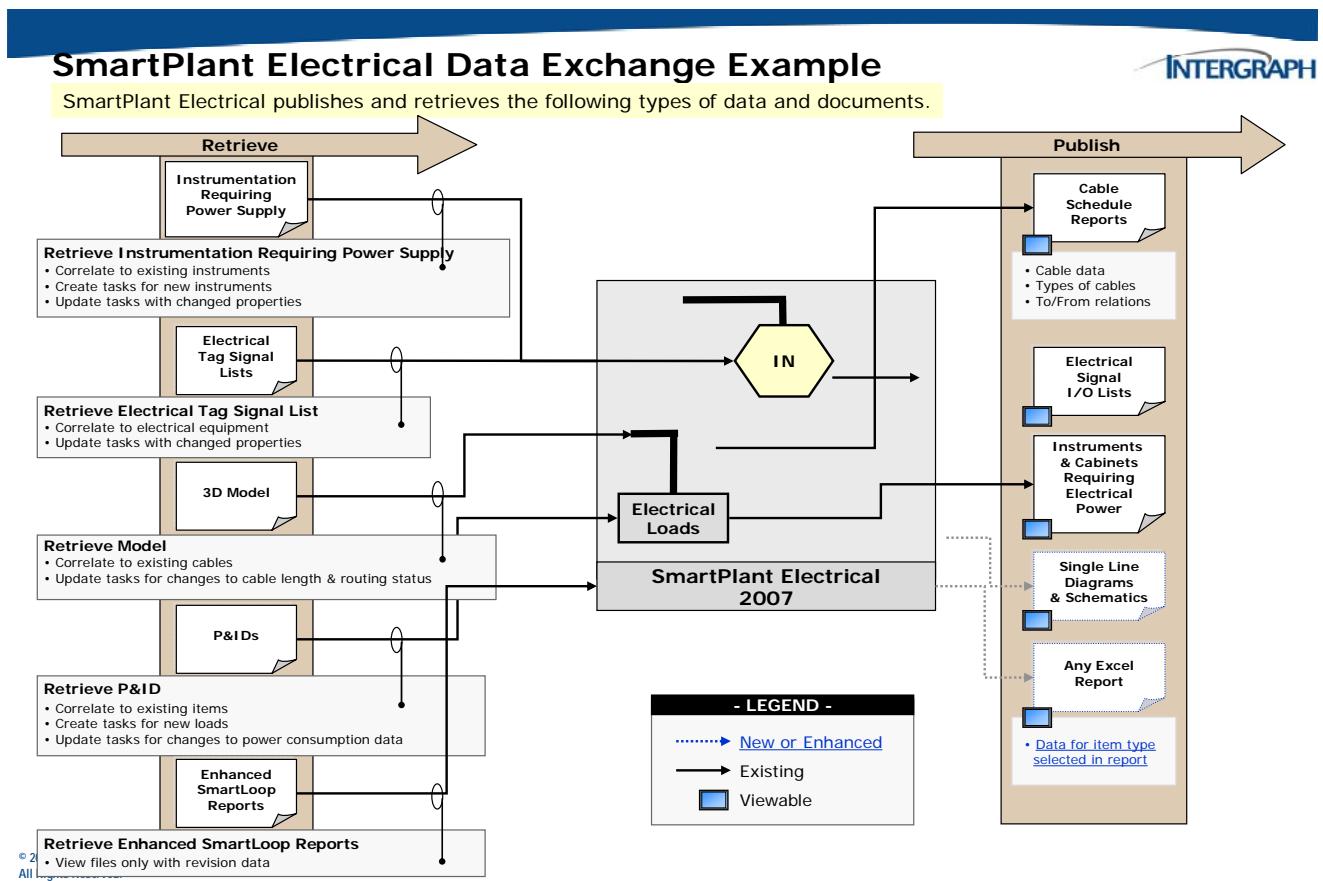


SmartPlant Instrumentation Data Exchange Example

SmartPlant Instrumentation publishes and retrieves the following types of data and documents.



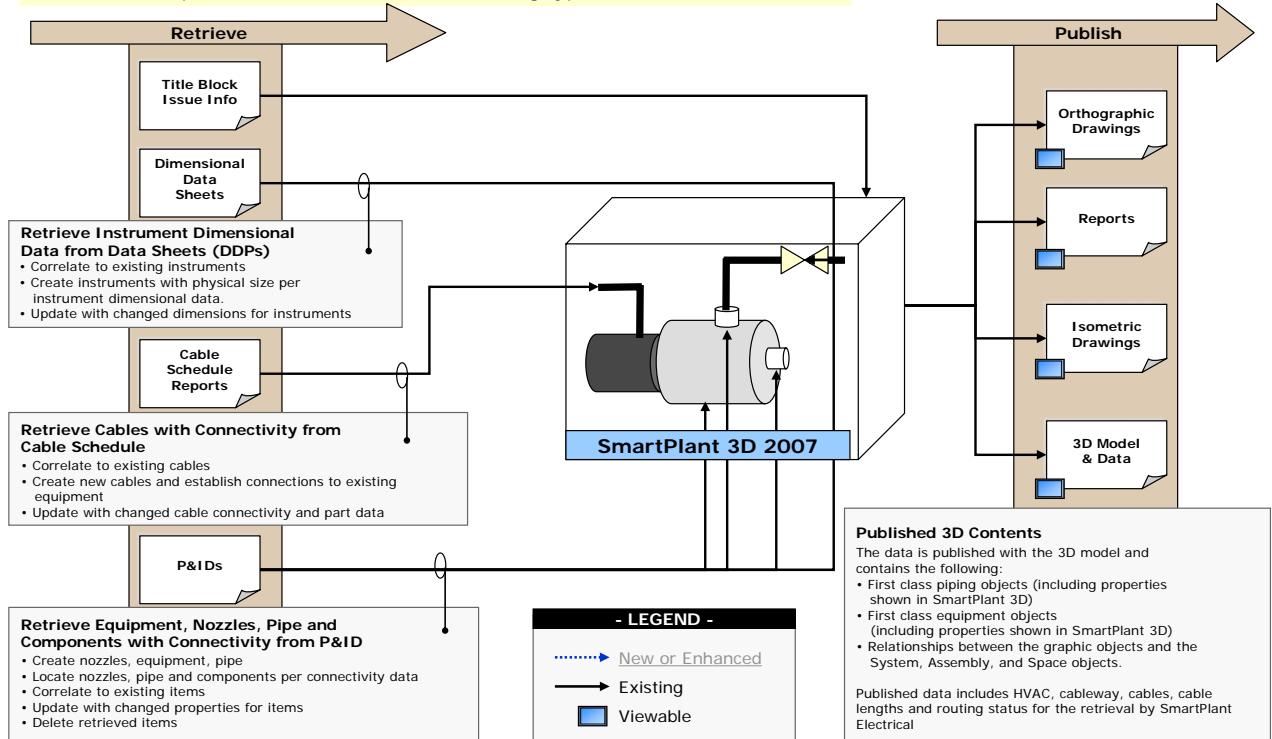
© 2006 Intergraph Corp.
All Rights Reserved.



SmartPlant 3D Data Exchange Example



SmartPlant 3D publishes and retrieves the following types of data and documents.

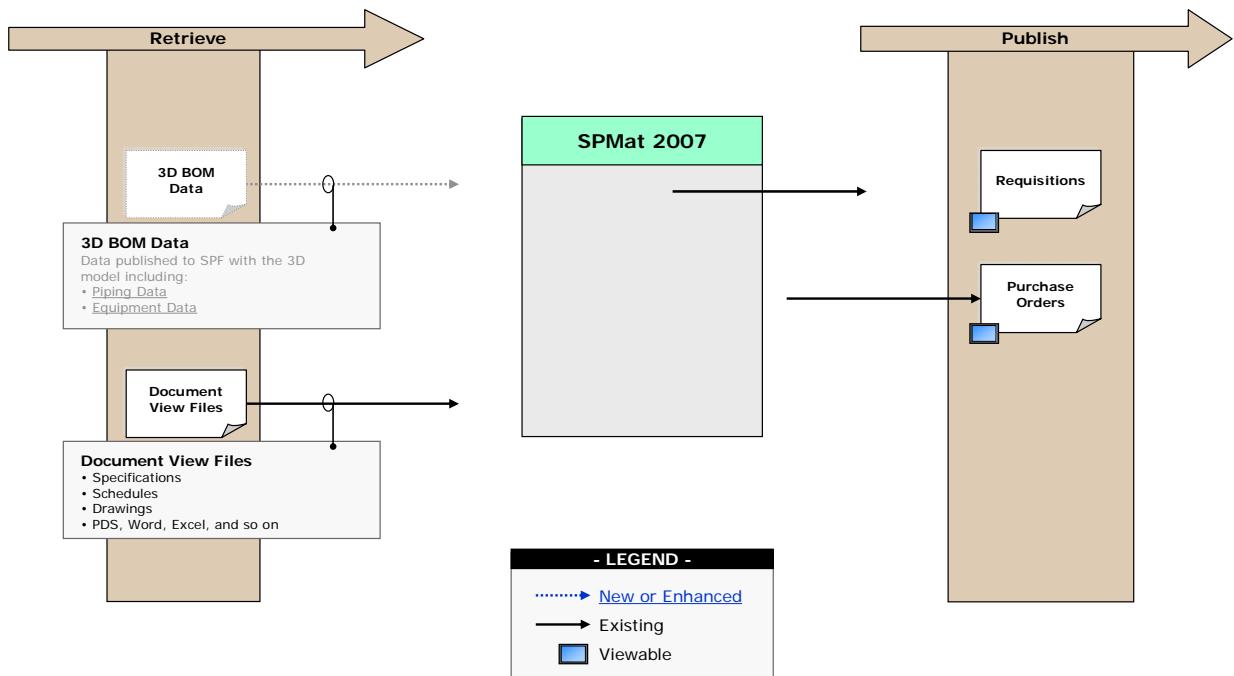


© 2005. Intergraph Corp.
All Rights Reserved.

SmartPlant Materials Data Exchange Example



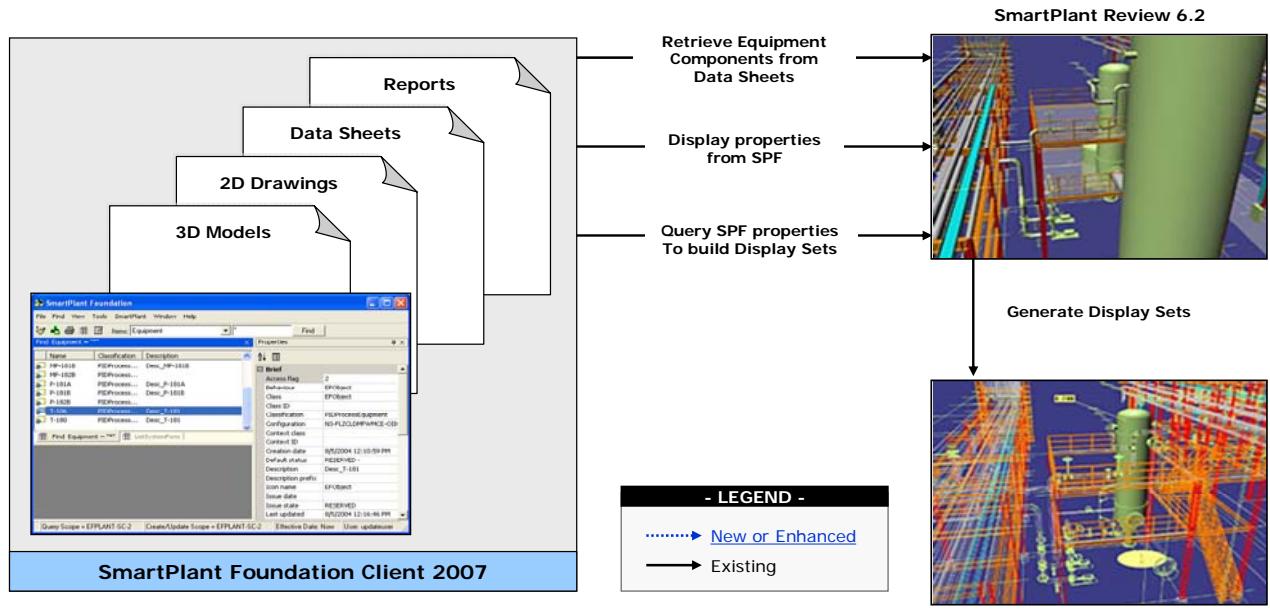
SmartPlant Materials publishes and retrieves the following types of data and documents.



SmartPlant Review Information Exchange Example



Data flows into and out of SmartPlant Review in the following way when it is used in an integrated environment.

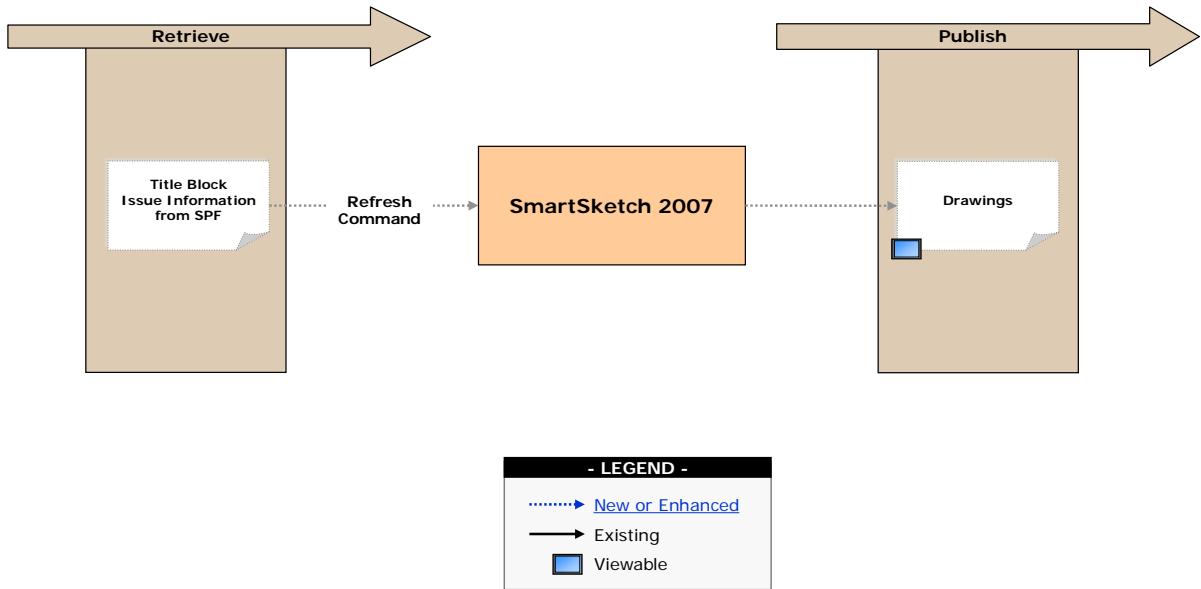


© 2005, Intergraph Corp.
All Rights Reserved.

SmartSketch Data Exchange Example

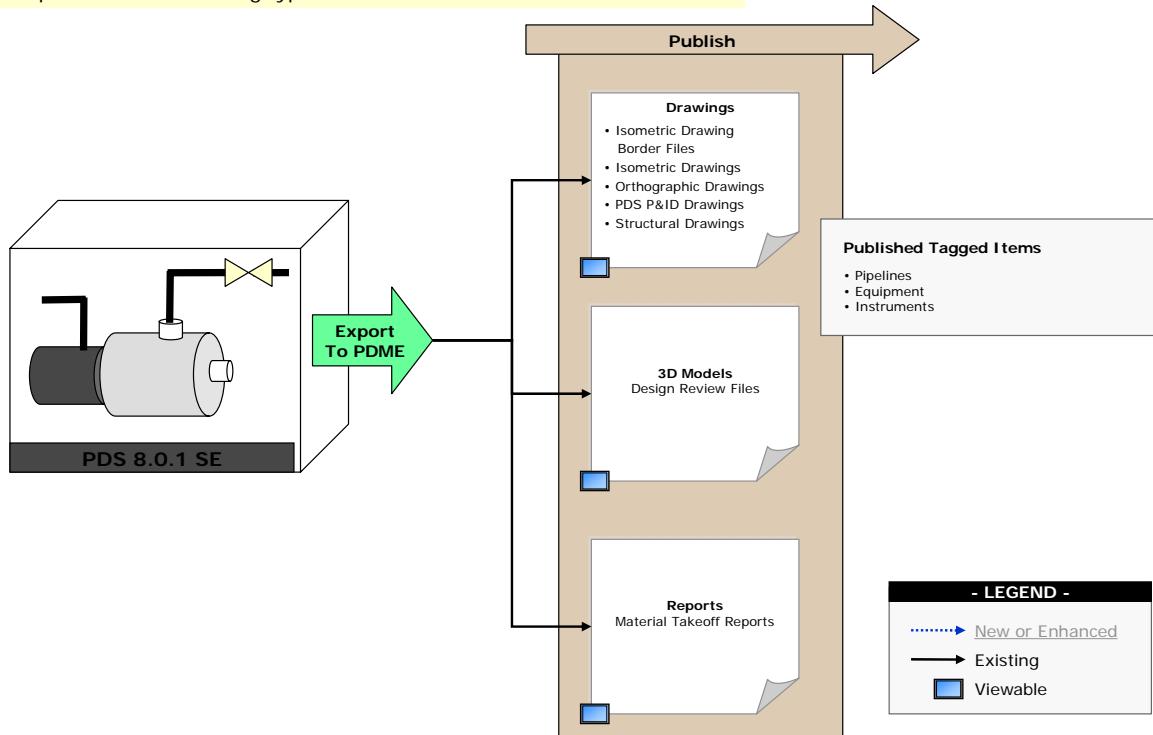


SmartSketch publishes and retrieves the following types of data and documents.



PDS Data Exchange Example

PDS publishes the following types of data and documents via Model Loader.



© 2005, Intergraph Corp.
All Rights Reserved.

7.7 SmartPlant Foundation Mapping Spreadsheets

SmartPlant Foundation mapping spreadsheets are available on Intergraph's eCustomer (<http://crmweb.intergraph.com>) under:

Download Software Updates > Products > SmartPlant Foundation > Technical Notes and Documentation

If you don't have a eCustomer login, you can request one at <https://crmweb.intergraph.com> then click on the *I'm a new user* link, fill out the form, and your eCustomer login will be provided.

7.8 Activity 1 – Extending the Authoring Tool

In this activity you will be extending a select list that will be used with an existing property. You will also use Oracle to determine the index numbers used for your codelist (select list) in order to verify the schema mapping in the next activity.

1. Log on to your operating system as *spfuser* with no password (if not already logged in). Optionally, use Microsoft Excel to open the following file:
D:\SPF_Training\Map_SpreadSheets\corr_codes.xls before starting Data Dictionary Manager.
2. Click *Start > All Programs > Intergraph SmartPlant engineering Manager > Data Dictionary Manager* to start the *SmartPlant Data Dictionary Manager*.
3. Add new entries to the **Fluid System** *select list* (picklist).
 - Click **Select Entry** on the left side of the application window.
 - In the *Select Entry* window, choose **Fluid System** from the list, scroll to the bottom and enter the following information:
 - Value – **Corrosive**
 - Short Value – **Corrosive**
 - Dependent Value – <blank>
 - In the *Select Entry* window, choose **Fluid Code** from the list, scroll to the bottom and enter the following information (you can copy and paste the Short Value from the excel spreadsheet):
 - Value – **KA**
 - Short Value – **(KA) Ammonia, Anhydrous**
 - Dependent Value – **Corrosive**
 - Select the **Add Row** command from the tool bar to add the next entry for **Fluid Code**:
 - Value – **KC**
 - Short Value – **(KC) Caustic**
 - Dependent Value – **Corrosive**
 - Select the **Add Row** command from the tool bar to add the next entry for **Fluid Code**:

- Value – **KP**
 - Short Value – **(KP) Process Chemical**
 - Dependent Value – **Corrosive**
- Select the **Add Row** command from the tool bar to add the last entry for **Fluid Code**:
- Value – **KW**
 - Short Value – **(KW) Ammonia, Aqueous**
 - Dependent Value – **Corrosive**
4. Save the changes to the SmartPlant Data Dictionary Manager.
- Click **File > Save**.
- Click **File > Exit**.

Determining the P&ID Data Dictionary Codelist ID

5. Click **Start > All Programs > Oracle – OraClient10g_home1 > Enterprise Manager Console** to start the *Oracle Enterprise Manager* or use the shortcut **Oracle Enterprise Manager Console** from the Desktop.
- In the *Oracle Enterprise Manager Console* window, expand **Databases** and **SPFDB**. In the *Database Connect Information* dialog enter the following information to log in:
- Username – **system** (you may have to key in s two times)
 - Password – **oracle**
- Click **OK** to log in to the Oracle database.
6. View the tables in the PID data dictionary for our SmartPlant P&ID plant user.
- In the tree view, click + next to Schema, and then the **EFPLANTSC1PIDD** entry, which is our oracle user.
- Locate the **Enumerations** table, right click and select **View/Edit Contents...** to review the table.
- Find *Fluid Code* in the Name column. Note the ID number assigned to the *Fluid Code* entry as well as the DependsOnId. Write both here _____ / _____.
- Close the *Enumerations* table **without** making any changes.

- Locate the **Codelists** table, right click and select **View/Edit Contents...** to review the table. Choose the *Display all rows* option.
 - Sort the table by CODELIST_NUMBER in descending order and find the entries you added earlier (in the CODELIST_TEXT column).
 - KW
 - KP
 - KC
 - KA
 - Note the CODELIST_INDEX number assigned to each entry and write them below:
 - KW _____
 - KP _____
 - KC _____
 - KA _____
 - Click **Close** to return to the list of tables.
7. Exit from Oracle without making any changes to the database.
- Click **File > Exit** to close and exit the *Oracle Enterprise Manager*.
8. Continue to activity 2 of this chapter.

7.9 Activity 2 – Mapping with the Schema Editor

The goal of this activity is to give you the opportunity to create custom enumerated in the SmartPlant schema and synchronize with the SmartPlant P&ID tool map schema. Finally, you will perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in). Verify that the correct location is set for the SmartPlant Schema file via *Server Manager*.
2. Click *Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor* to start the *Schema Editor*.
3. Open the SmartPlant schema using the *EFSchema.cfg* configuration file.
 - Click *File Configurations > Open Configuration....*
 - Browse to **C:\Program Files\Common\Intergraph\EFSchema\03.08** in the *Open Configuration File* dialog box.
 - Select **EFSchema.cfg** file and click **Open**.
4. Create a new extension schema file that will contain all of the custom extension modeling statements.
 - In the *Workflows* dialog box, click the *New* button above the *Another Schema File* button.
 - When the *New Another Schema File* dialog box appears, enter the name **ToolExten.xml** and click **Save**.
 - When the *Set Container Dependencies* dialog box displays, select the master **EFSchema.xml** file and **EdgeGraphView.xml**, then click **OK**.
5. Set the active schema file to be modified during the modeling edit sessions.
 - Click the *Another Schema File* button in the *Workflows* dialog box.
 - Select the *Set Active Schema File...* command
 - When the *Set Active File* dialog box displays, select the **ToolExten.xml** file and click **OK**.

6. Edit the configuration to be used for both the master schema and the extension schema during the modeling/mapping edit sessions.
 - Click the **Another Schema File** button in the *Workflows* dialog box.
 - Select the **Edit Configuration...** command.
 - When the *Edit Configuration* dialog box displays, make the **EFSchema.xml** file not editable by clicking in the **Editable?** column and toggling the setting to **No**. Set all other xml files to not editable.
 - Make the **ToolExten.xml** file editable by clicking in the **Editable?** column and toggling the setting to **Yes**.
 - Verify that the *Active schema file* is **ToolExten.xml**.
 - Close the *Edit Configuration* window (click **X**).
7. Save this configuration so that it can be used in future mapping sessions.
 - In the *Workflows* dialog box, click the **File Configurations** button in the lower right corner of the application window.
 - Select the *Save As Configuration File* command and save using the name **EFSchema.cfg**.
8. Exit the Schema Editor and save your work.
 - Select the **File > Exit** command from the menu
 - When prompted to save changes to the other xml files, click **Yes**.
 - Edit the **EFSchema.cfg** file with WordPad and remove the **Extensions1.xml** entry.

Creating SmartPlant Schema Mapping

9. Re-start the *Schema Editor* by selecting **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor**.
10. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the **SmartPlant** button and click **Connect to SmartPlant...**

11. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with
 - Click **Next**
 - Select the tool map schema to be loaded, *SMARTPLANTPID/SmartPlant P&ID Tool Schema*
 - Enable the **Load map schema** toggle
 - Enable the **Connect to application schema** toggle
 - Click **Finish**
12. When the *Synchronize* dialog displays, confirm that the new select list values have been imported by the meta data adapter and click **OK**. You are now in the **Map Environment** window.
13. View the master schema using the *Schema Editor* to verify that the new enumerated values DO NOT YET EXIST.
 - In the main menu, click the **View > Schema (All)** command to open another window on top of the *Map Environment* window.
 - When the *View Schema* dialog box appears, select the *View* tab, choose the **Tree/Drag-Drop UML** option.
 - Click **OK**
14. View the **Fluid system** *EnumListType* to make sure that these values are not in the SmartPlant schema.
 - Expand the **EnumListType** objects in the base classes tree.
 - Right-click on **Fluid system** and select **View FluidSystems** from the dynamic menu
 - Verify that the value **Corrosive** does not yet exist
 - Verify that none of the child values for *Corrosive* exist (KA, KC, KP, KW)
 - Cancel** the edit dialog and **Close (X)** the view window

15. Open Microsoft Excel (if not already open) and copy the contents of **corr_codes.xls** to the clipboard. The path is D:\SPF_Training\Map_SpreadSheets.

	A	B	C	D
1	KA	(KA) Ammonia, Anhydrous	10101	
2	KC	(KC) Caustic	10102	
3	KP	(KP) Process Chemical	10103	
4	KW	(KW) Ammonia, Aqueous	10104	
5				
6				

16. Add new entries to an existing Enumerated List in the SmartPlant schema to correspond to new entries that have been added to a tool map schema during the synchronization.
- Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant P&ID Tool Schema entries are displayed.
 - Expand the **Map Enumerated Lists**, **FluidSystem** and **Corrosive** objects. Verify that the entries were added to the SmartPlant P&ID Tool Map Schema as a result of the synchronization with the SPPID application meta schema.
 - Right-click on the FluidSystem entry in the tree and choose **Edit SP_20** from the dynamic menu.
 - What does *SP_20* represent? _____
 - Where does the number originate from? _____
 - Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog. Then right-click on an existing enum **child** list in the SmartPlant control and choose **Create New EnumListType** from the dynamic menu.
 - In the *New Enumerated List* dialog box, enter the following information:
 - Name/short description – **Corrosive**
 - Long description – **Corrosive Fluid System**
 - In the *New Enumerated List* dialog, select the **Import Entries from Clipboard** button.
 - Verify the contents are correct with what was displayed in Excel

- Click **OK** to perform the import
- In the *New Enumerated List* dialog, verify that the new child entries were added correctly to **Corrosive**. Click on *Corrosive* in the displayed tree.
- Click the **OK** button to close the *New Enumerated List* dialog.

Defining Schema Mapping

17. Highlight the *Corrosive* MapEnum in the **Application** upper tree control and the *Fluid system* enumerated list in the **SmartPlant** upper tree control in order to perform the schema mapping.
 - Make sure that **Corrosive** is highlighted in the middle control in the application section and that **Corrosive** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPPID tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *Corrosive* maps to *Corrosive* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Enumerated List Definition* dialog box.
18. Locate the *FluidCode* Map Enumerated List definition in the tree and perform the schema mapping.
 - Right-click on the *FluidCode* entry in the tree and choose **Edit SP_19** from the dynamic menu.
 - What does *SP_19* represent? _____
 - Where does the number originate from? _____
 - Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog.
 - Make sure that **KA** is highlighted in the middle control in the application section and that **KA** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPPID tool map schema and the SmartPlant schema.
 - Verify in the bottom control that **KA** maps to **KA** in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Repeat the mapping for the other FluidCode unmapped entries; KC, KP, KW
 - Click **OK** to close the *Edit Map Enumerated List Definition* dialog box.

19. Save the changes to the all schema files.
 - From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **Yes**.
20. Use Windows Explorer to verify that all the files were uploaded to **C:\Program Files\Common\Intergraph\EFSchema\03.08** with the latest time and date.
21. When you are finished with this activity, you may take a short break until the other students have finished.

C H A P T E R

8

Mapping with SmartPlant P&ID

8. Mapping with SmartPlant P&ID

In the last chapter, the concept of extending the delivered schema was introduced along with using the SmartPlant Schema Editor to perform the necessary mapping. In the example, new enumerated values were added to existing enumerated lists. The process involved using the Data Dictionary Manager to add the desired Select List entries to the SPPID application. The next step was to use the Schema Editor and its SPPID meta data adapter to read the new enumerated additions and automatically synchronize those additions with the SPPID tool map schema. The final steps were to add the new entries to the SmartPlant Schema and map the two schemas together using the schema editor Map Environment.

In this chapter, more additions to the SmartPlant P&ID application and tool map schema will be covered in details. Two new properties will be added and mapped, with one property being a textual property type and the other utilizing a brand new enumerated list.

Two different methods for making these changes will be demonstrated in this chapter.



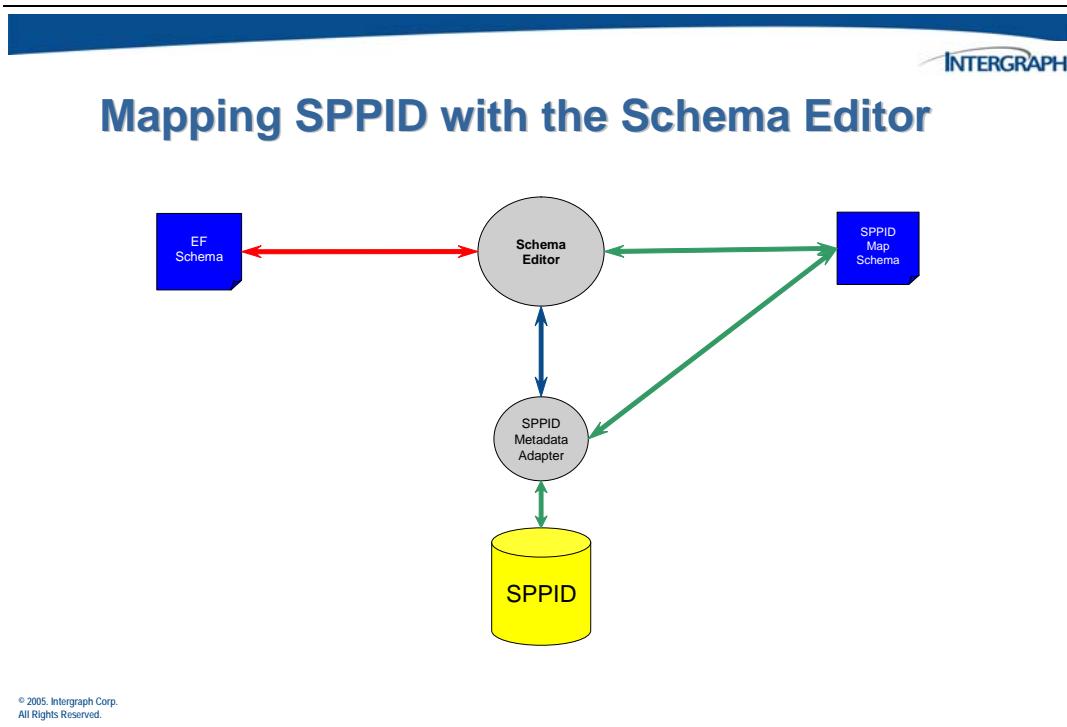
Mapping SPPID with the Schema Editor

There are two methods for extending the delivered schema:

- Make any necessary additions in the SPPID Meta-Schema using the Data Dictionary Manager**
 - this is the recommended process for adding new properties

- Make any changes using the schema editor, which will allow you to add those changes to the SPPID Meta-Schema, the SPPID tool map schema and the SmartPlant Schema**
 - this is the recommended process for adding a new enumerated list

The figure below shows the function of the metadata adapter with the schema editor.

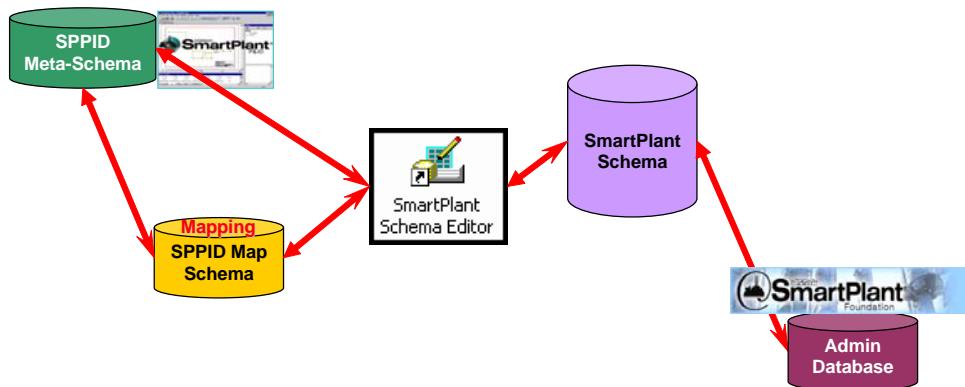


As in the previous example, the desired change will first be made in the SmartPlant P&ID application using the Data Dictionary Manager tool. There is a specific reason for approaching the problem in this manner. The property to be added could be used by several different types of objects in SmartPlant P&ID; Instruments, Instrument Loops, Nozzles, Pipe Lines, etc. Rather than add the same property multiple times, if the new property will be used as a “global” *Plant Item* item type property, then it can be added to the Plant Item table in the data dictionary but be available to other classes through inheritance. The *PlantItem* item type is a classification that provides common relationships and attribution for all the types of objects that may exist independently of any other object. This item type is derived from the *ModelItem* class. The subclasses of *ModelItem* include all those types whose existence is not dependent on another.

Once the new property has been added with the Data Dictionary Manager, the schema editor will be used to read this new property from the application database which is also called the tool meta schema. This is accomplished by a part of the software known as the metadata adapter. The task of the metadata adapter is to extract information from the meta schema and automatically update the contents of the tool map schema so that the meta schema and the tool map schema are synchronized.



Mapping SPPID with the Schema Editor



© 2005, Intergraph Corp.
All Rights Reserved.

- Changes are added to the **SPPID Meta schema** via the *Data Dictionary Manager* (new property defined).
- Any changes are automatically added to the **SPPID Map Schema** when the schema editor synchronizes the schema files.
- The same changes get added to the **SmartPlant Schema** using the schema editor *Map Environment*.
- Any unmapped changes to the schemas can be mapped using the schema editor in preparation for publish/retrieve operations.

This will allow for the smooth flow of published data from SmartPlant P&ID into the SmartPlant Foundation database.

8.1 Adding Properties to the Meta schema

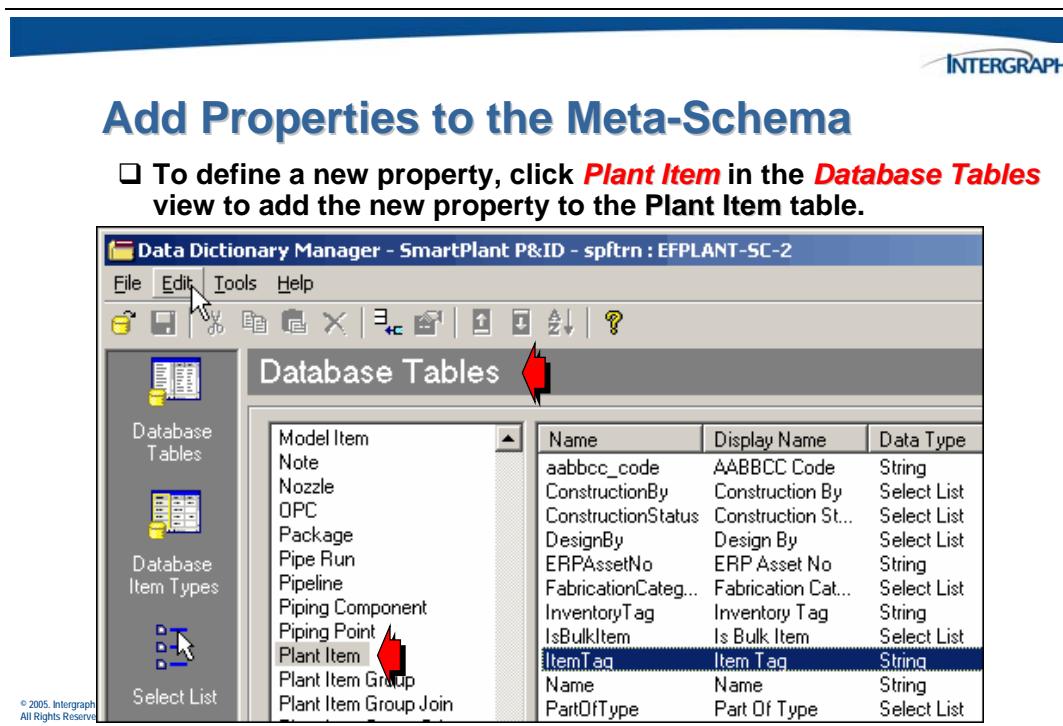
The procedure for adding properties to the authoring tools differs from tool to tool. The following example describes the procedure for adding a new property to the SmartPlant P&ID data model using SmartPlant Data Dictionary Manager (delivered with SmartPlant Engineering Manager). For more information about adding properties to SmartPlant P&ID, see the SmartPlant Data Dictionary Manager documentation.

To add a property to SmartPlant P&ID, open Data Dictionary Manager by clicking **Start > All Programs > Intergraph SmartPlant Engineering Manager > Data Dictionary Manager**.

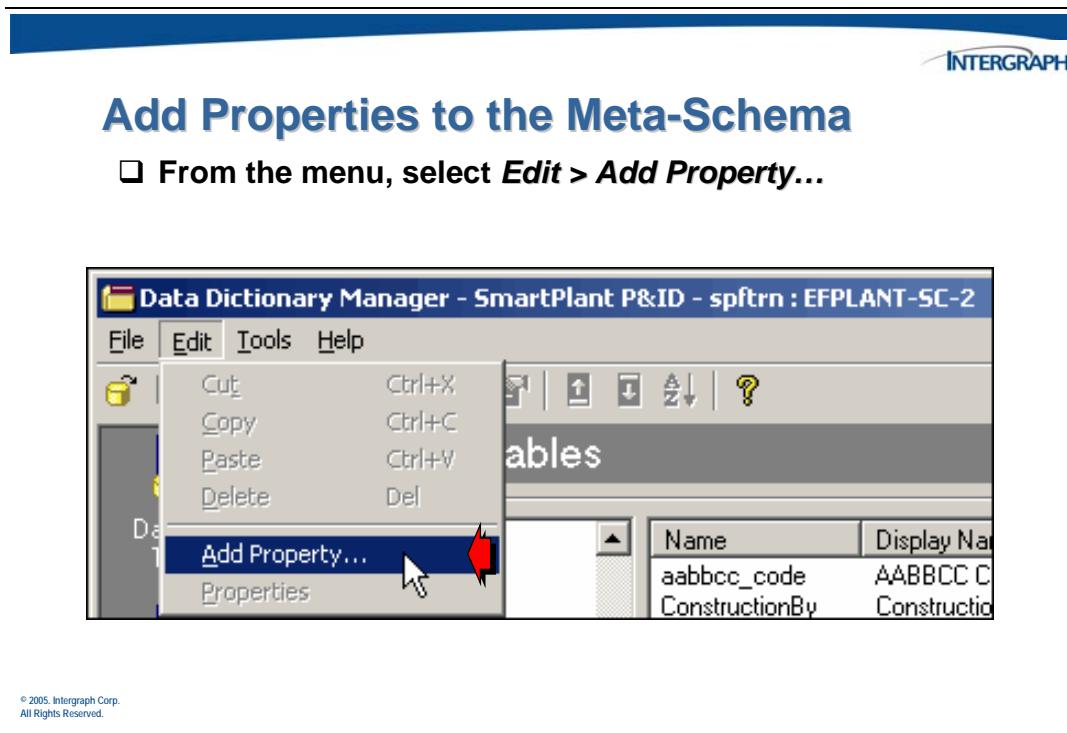
The following example describes how to add a new property to the SmartPlant P&ID data model. This property includes the following:

- A simple property without an enumerated list

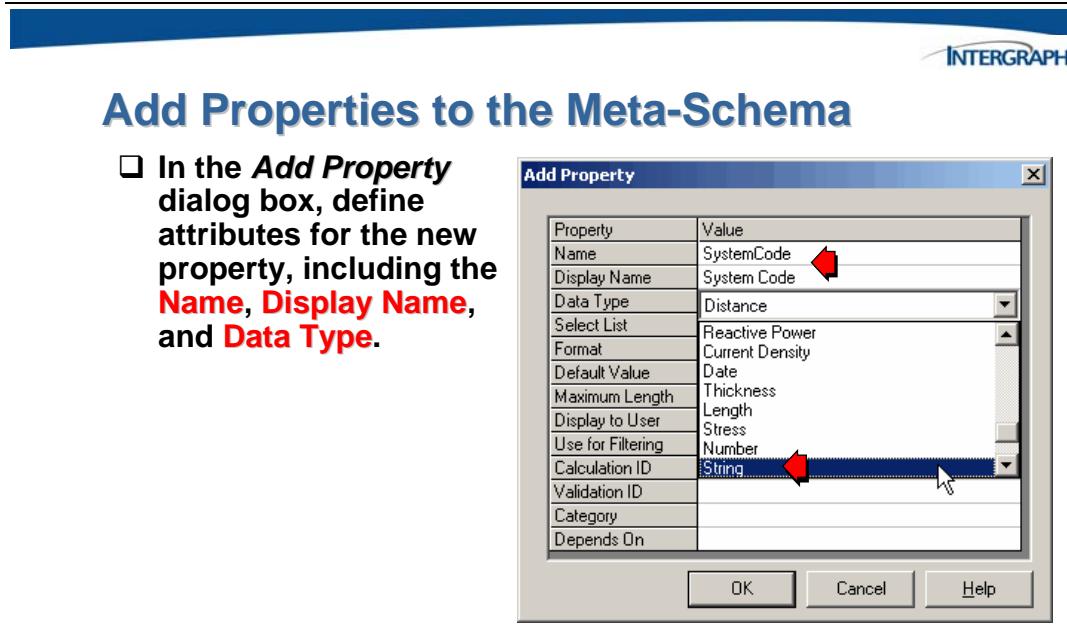
When you open Data Dictionary Manager, the **Database Tables** view is already selected. To select the database table to which you want to add a property, click the name of the table in the list.



After you select the table that you want, you can add a property to that table.



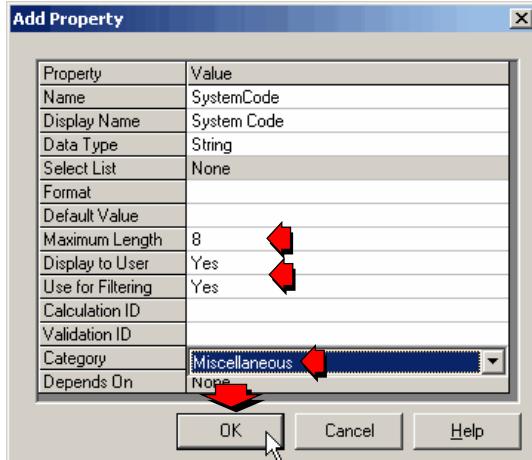
The **Add Property** dialog box appears. Define information for the new property in the **Add Property** dialog box.



Add Properties to the Meta-Schema

- Continue to define attributes for the new property, including the **Maximum Length**, **Display**, **Filtering**, and **Category**.

- Click **OK** to add the new property to the Plant Item table.

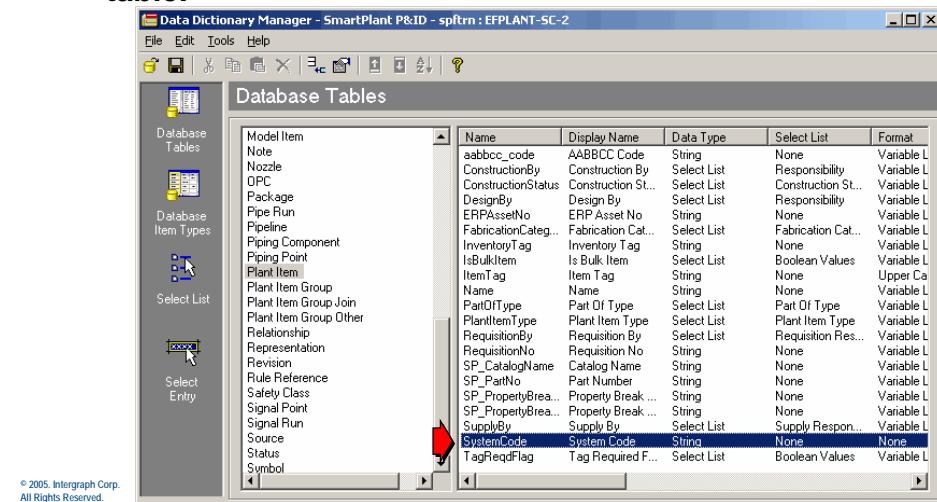


© 2005, Intergraph Corp.
All Rights Reserved.

When you click **OK**, the list of properties for the *Plant Item* table updates with the new property.

Add Properties to the Meta-Schema

The new **SystemCode** property displays in the Plant Item table.

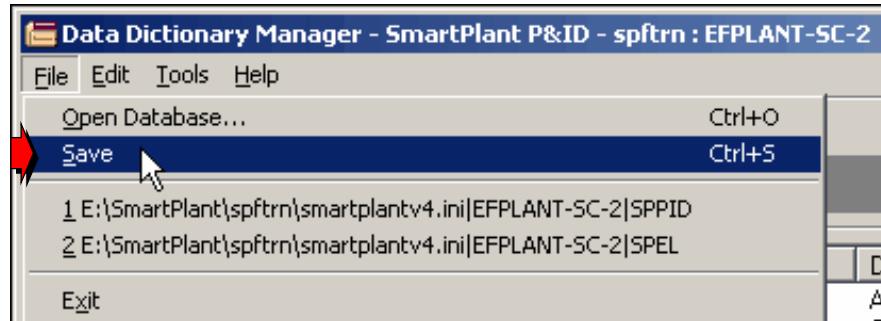


© 2005, Intergraph Corp.
All Rights Reserved.

After you add the property, save your changes to the SmartPlant P&ID meta schema.

Add Properties to the Meta-Schema

- From the menu, select **File > Save**.

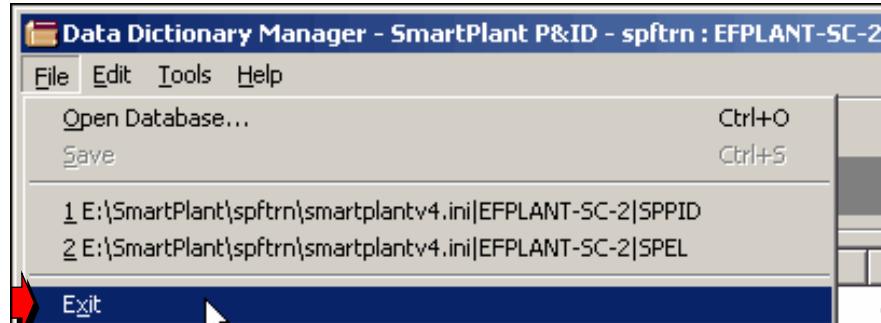


© 2005, Intergraph Corp.
All Rights Reserved.

Click the **File > Exit** command to close out of Data Dictionary Manager.

Add Properties to the Meta-Schema

- Select **File > Exit** from the menu.



© 2005, Intergraph Corp.
All Rights Reserved.

8.2 Performing a Schema Analysis

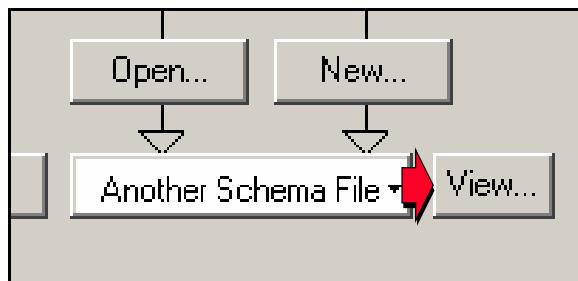
Before adding a new property to the SmartPlant schema, use the Schema Editor to view and analyze the available class definitions. The key to this analysis is to determine the best interface definition to use to expose any new properties that will be added to the extended schema. In the following example, the SystemCode property will need to be exposed by an interface that is realized by a majority if not all of the needed class definitions.

To begin the analysis, start the schema editor and open the schema view to be used to perform this analysis.

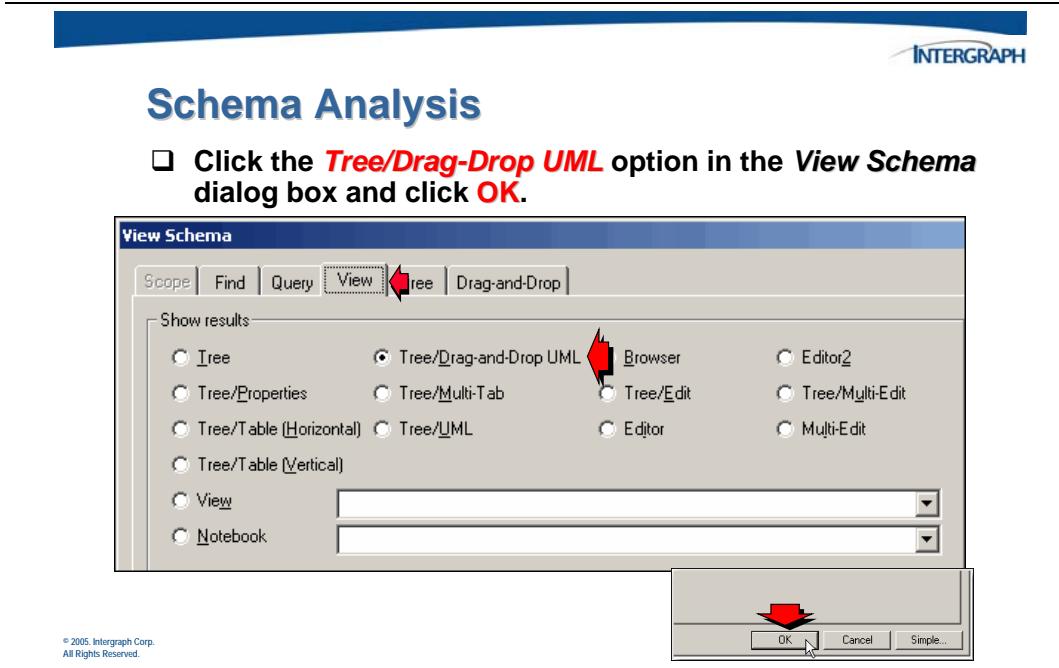


Schema Analysis

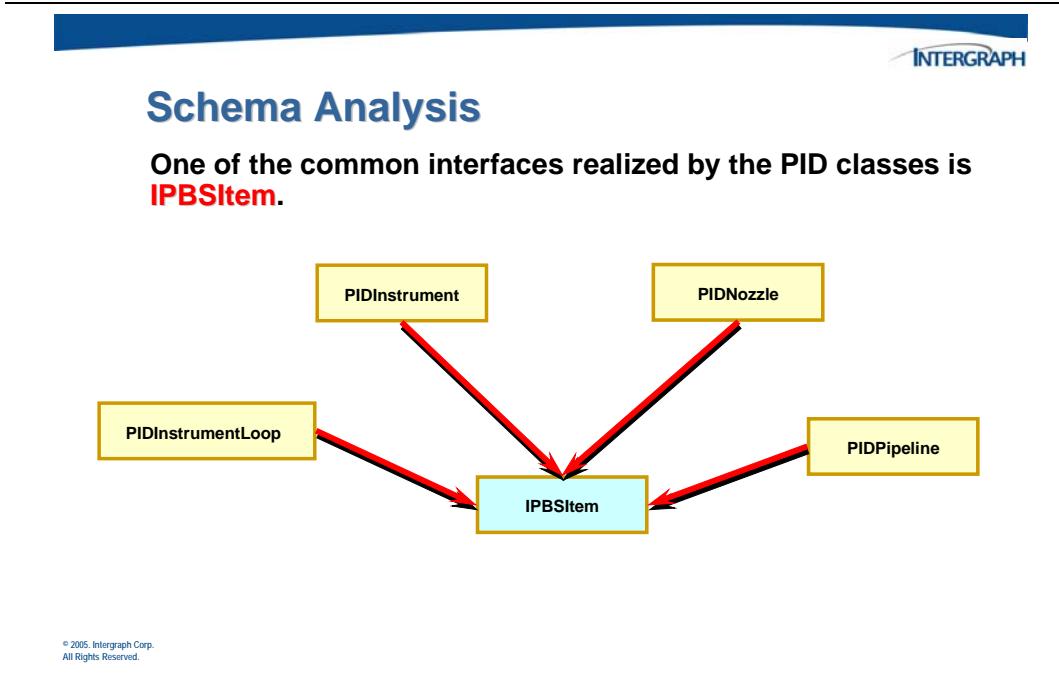
- Select the **View** button to the right of the **Another Schema File** button.



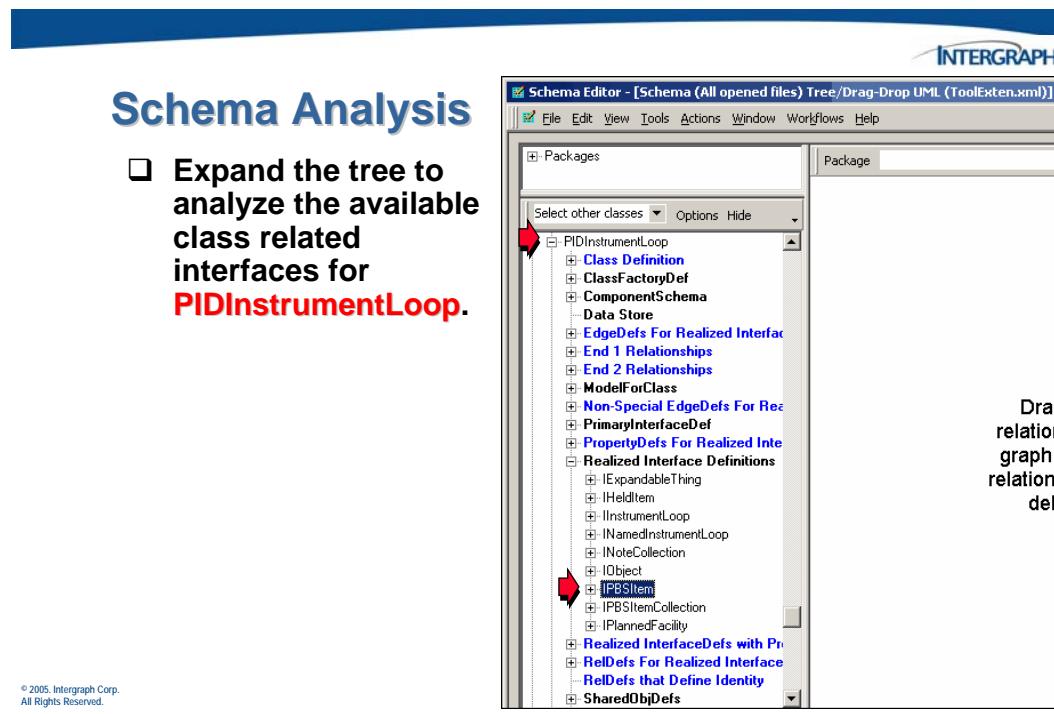
Click the **View** tab in the *View Schema* dialog box. For the following example, the **Tree/Drag-Drop UML View** will be used to perform the necessary schema analysis.



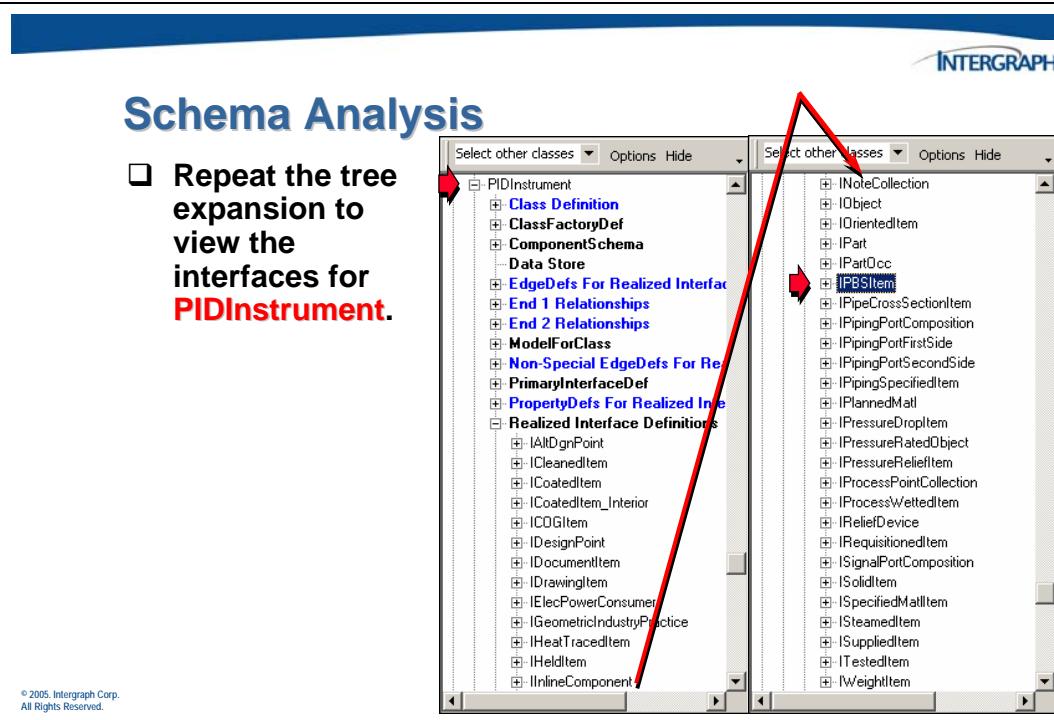
The SmartPlant schema class definitions to be reviewed will correspond to the tables listed in the Data Dictionary Manager. Here is a list of some of the classes (class def's) to be displayed: **PIDInstrumentLoop**, **PIDInstrument**, **PIDNozzle** and **PIDPipeline**.



Expand the *PIDInstrumentLoop* class in the tree to view the realized interfaces. The **IPBSItem** interface is a common interface realized by many classes and is a possibility.



Next, expand the *PIDInstrument* class in the tree to view the realized interfaces. Again, the **IPBSItem** interface is a possibility.



Repeat the expansion for the *PIDNozzle* class to view the realized interfaces. Note that **IPBSItem** is a possibility.

The screenshot shows the Schema Analysis interface with two main panes. The left pane displays the tree structure for the **PIDNozzle** class, which includes nodes for Class Definition, EdgeDefs For Realized Interface, and Realized Interface Definitions. The right pane shows a list of realized interfaces, with the **IPBSItem** interface highlighted by a red arrow. The title bar of the window reads "Schema Analysis" and the Intergraph logo is visible in the top right corner. A copyright notice at the bottom left states: "© 2005, Intergraph Corp. All Rights Reserved."

Finally, expand the tree to view the realized interfaces for *PIDPipeline*. Again, you will see that **IPBSItem** is listed.

The screenshot shows the Schema Analysis interface with two main panes. The left pane displays the tree structure for the **PIDPipeline** class, which includes nodes for Class Definition, EdgeDefs For Realized Interface, and Realized Interface Definitions. The right pane shows a list of realized interfaces, with the **IPBSItem** interface highlighted by a red arrow. The title bar of the window reads "Schema Analysis" and the Intergraph logo is visible in the top right corner. A copyright notice at the bottom left states: "© 2005, Intergraph Corp. All Rights Reserved."

In reviewing the SmartPlant schema, a common interface that can be used to expose a new property to many classes is IPBSItem. This knowledge is necessary before proceeding to the next step of extending and mapping the SmartPlant schema.



Adding a New SmartPlant Property

Add a Property
to Tool Map
Schema

Add a Property
to SmartPlant
Schema

© 2005, Intergraph Corp.
All Rights Reserved.

In the above figure, the process of adding a new property to the tool map schema will be automated by the tool metadata adapter.

8.3 SmartPlant Schema Additions

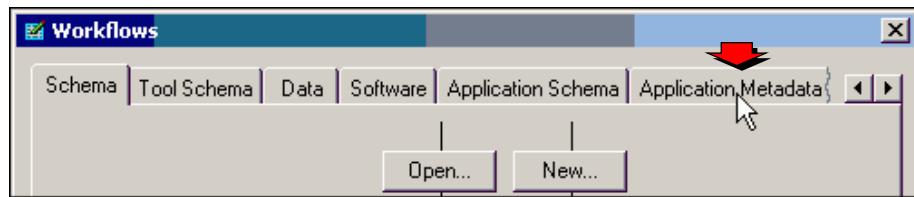
Before you can map a property from the tool map schema to the SmartPlant schema and back, you must use the Schema Editor to add the new property to the SmartPlant schema.

Start the Schema Editor and connect to SmartPlant.

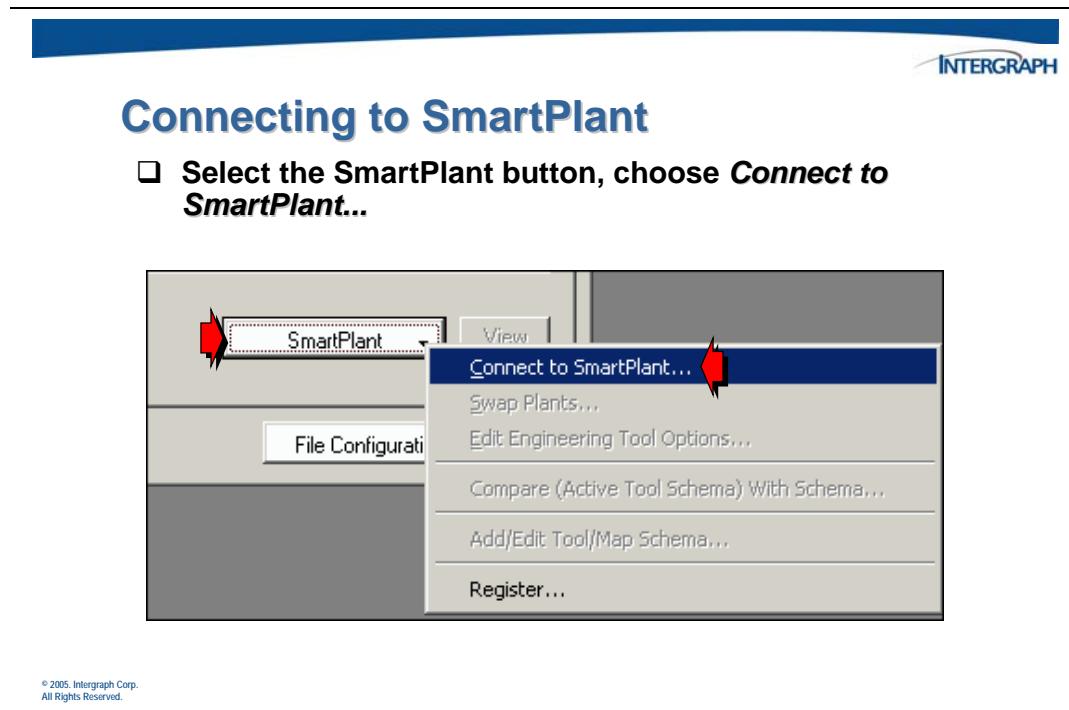


Connecting to SmartPlant

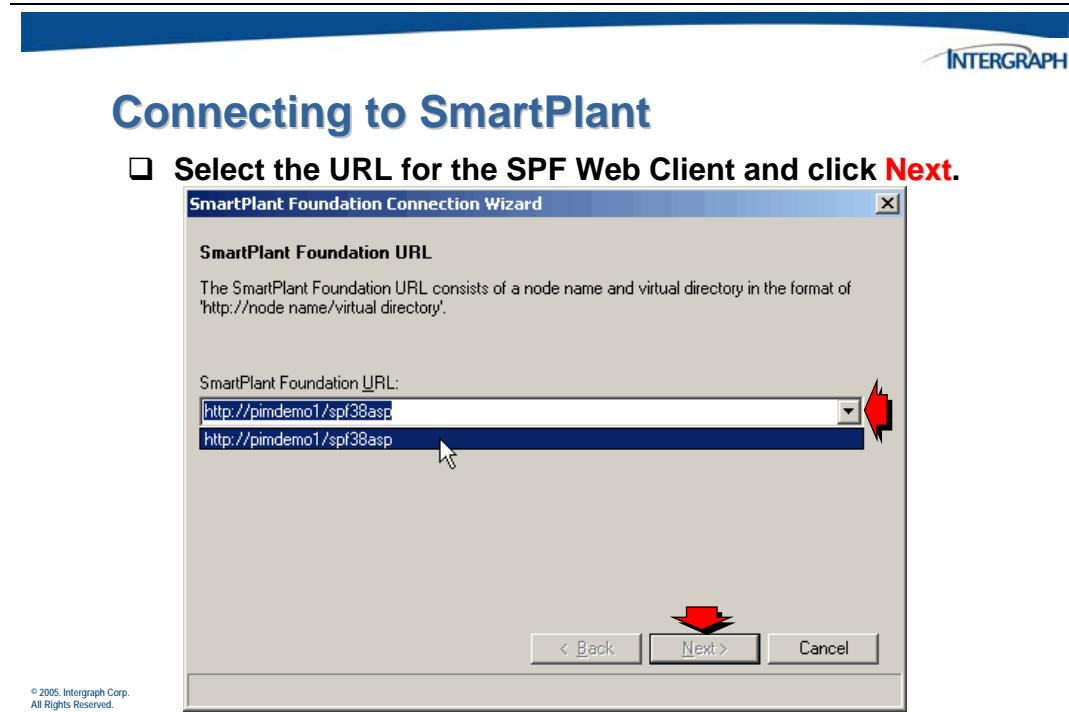
- In the **Workflows** dialog box, click the **Application Metadata** tab to connect to the SmartPlant Map Environment.



The *Application Metadata* dialog will display.



The *SmartPlant Foundation Connection Wizard* will be displayed.

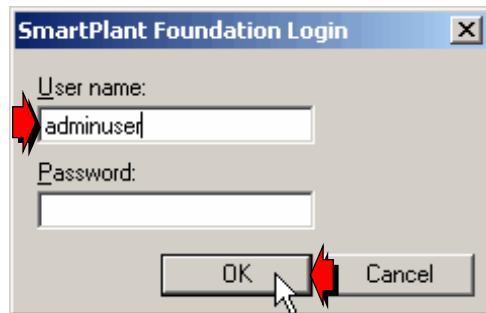


In the *SmartPlant Foundation URL* field, select the SmartPlant Foundation database with which you want to connect and click **Next**.



Connecting to SmartPlant

- Enter a valid SPF **User name** and **Password** to log on to the SPF server from the schema editor.



© 2005, Intergraph Corp.
All Rights Reserved.

Select the SmartPlant Foundation plant configuration that you will use for schema extension and mapping.



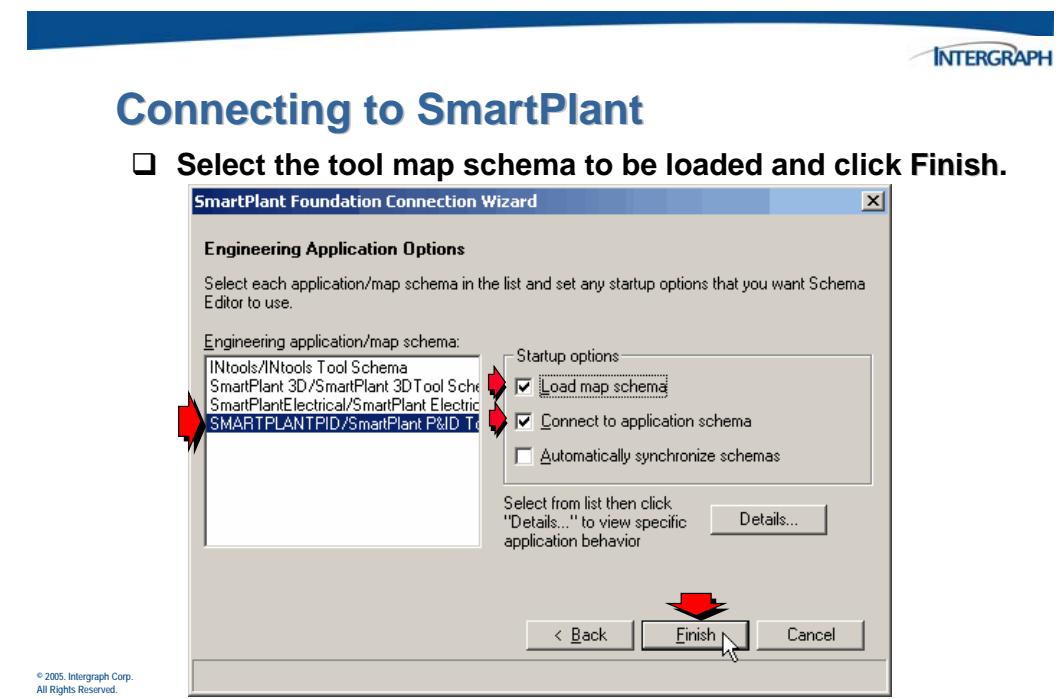
Connecting to SmartPlant

- Select the plant configuration to be used to perform the schema mapping and click **Next**.

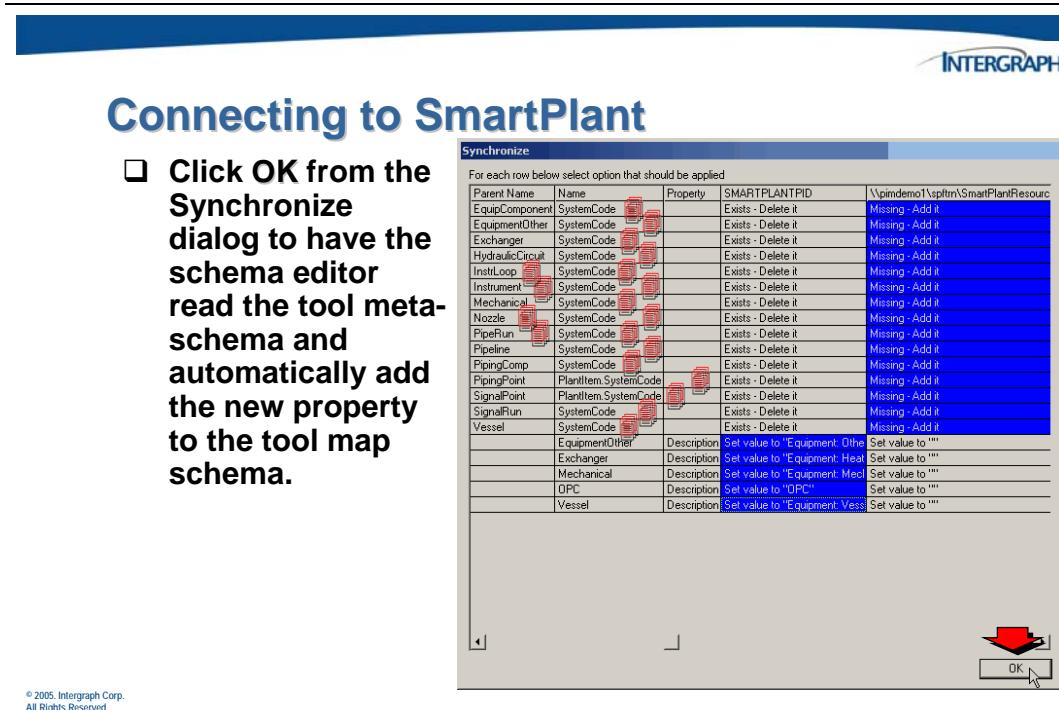


© 2005, Intergraph Corp.
All Rights Reserved.

The Schema Editor will find the SmartPlant schema and all **registered** tool map schemas.



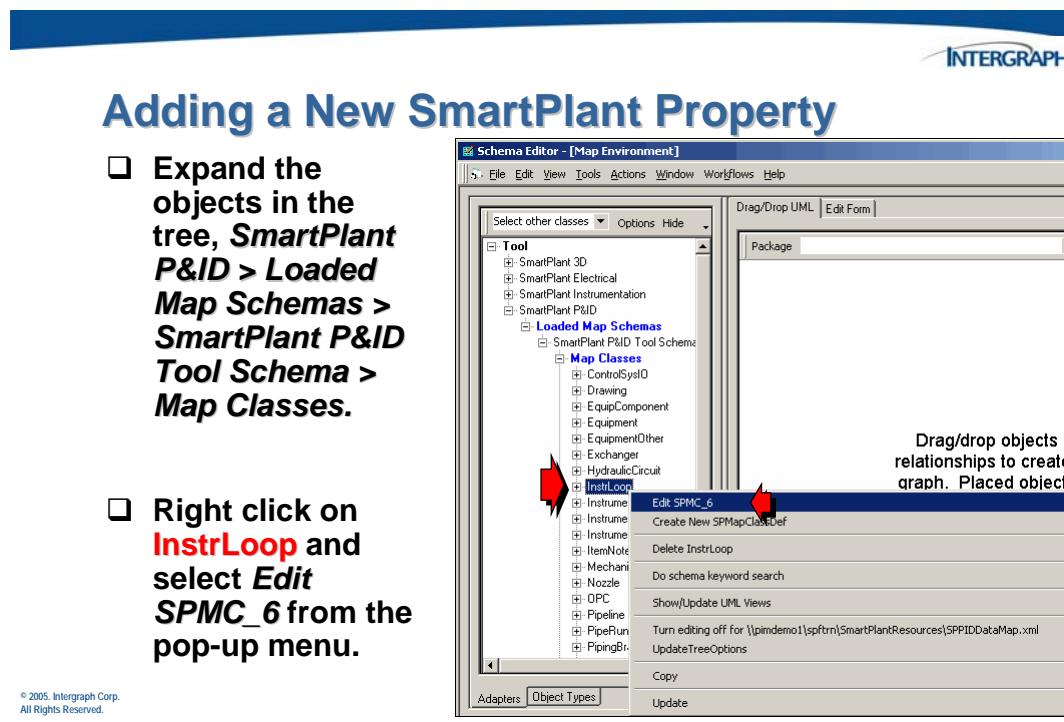
After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema. The differences between the two schemas are displayed in the **Synchronize** dialog.



During the connection process, a parsing of the tool map schema file will occur and the metadata adapter will also connect to its metadata data store (application tool database) and generate a tool map schema based on its application metadata. In the above example, you will note all the instances of **SystemCode** have been parsed for many of the SmartPlant P&ID classes.

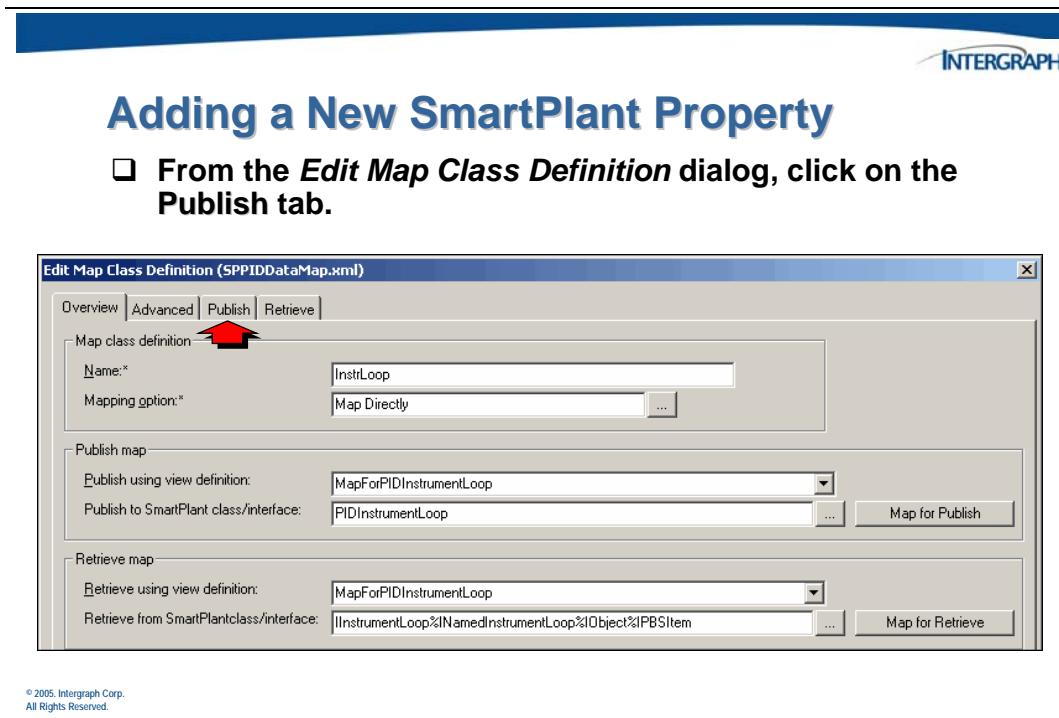
8.3.1 Adding a New SmartPlant Property

Use the Map Environment window to add a new property in the SmartPlant schema as well as perform the mapping to the SmartPlant P&ID tool map schema. When the tool map schema is expanded, it shows the map classes, map enumerated lists, map unit of measure lists and map relationship definitions for that tool map schema.

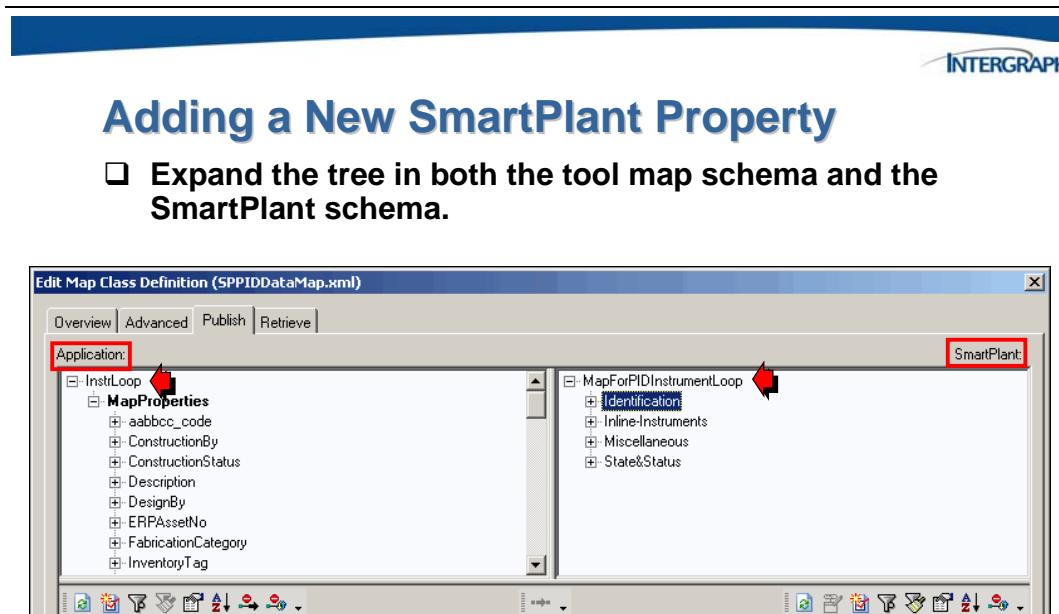


Locate the existing *InstrLoop* Map Class in the tree.

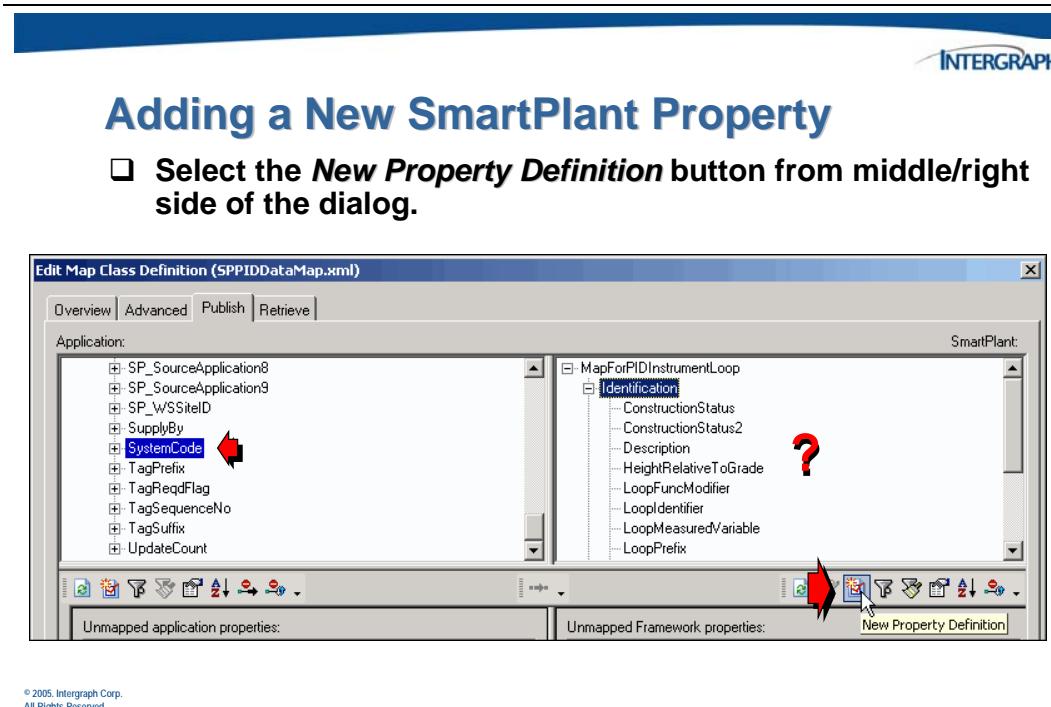
The *Edit Map Class Definition* dialog will be displayed.



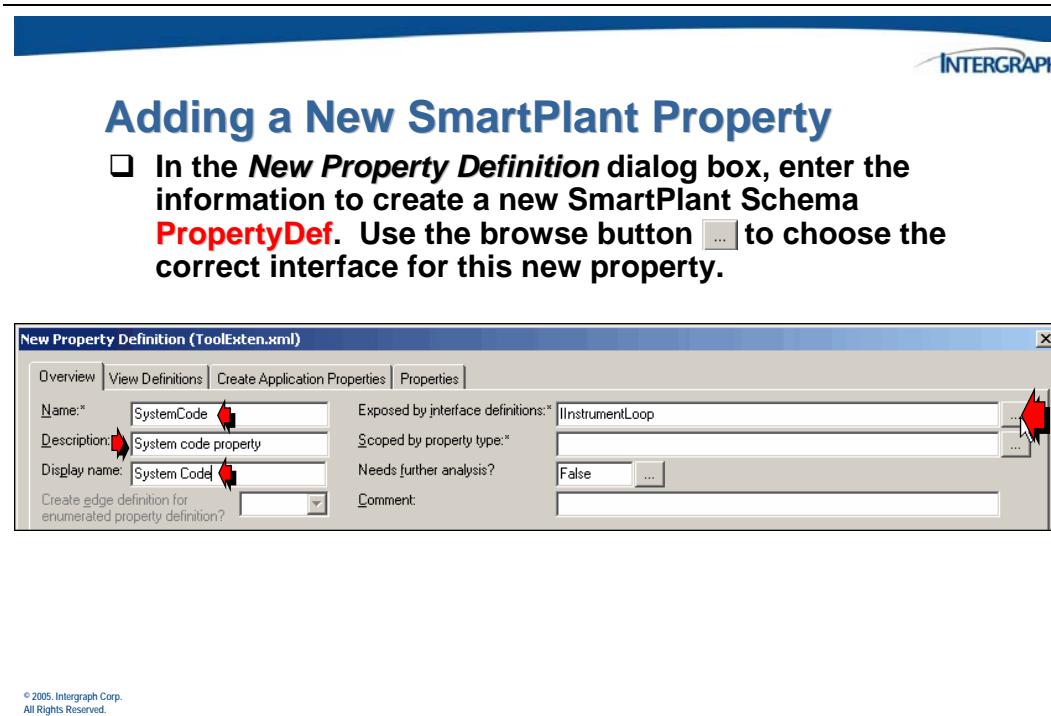
When the *Publish* tab is selected, the contents of the tool map schema will be displayed on the left side of the window and the SmartPlant schema contents will be displayed on the right side.



The Schema Editor will create a dynamic *View Definition* that will be used to display the SmartPlant schema properties. Selecting the mapped class will show all of the property definitions contained in the view definition.

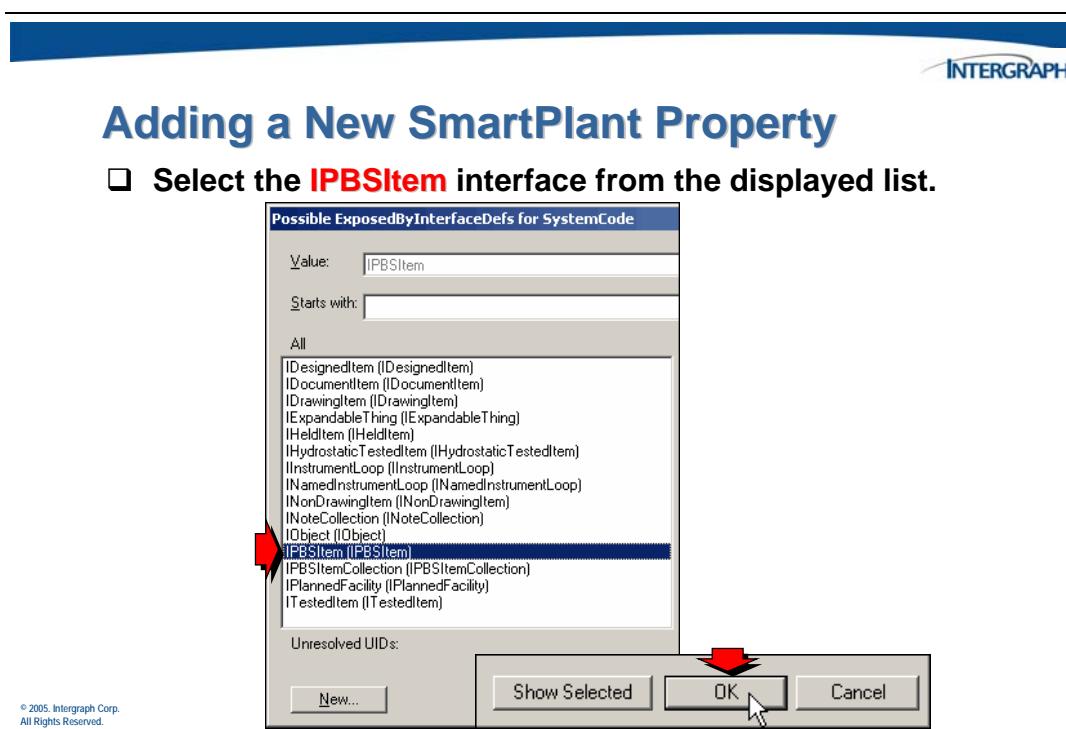


The *New Property Definition* dialog box will display.



The following describe the fields on the *New Property Definition* dialog box:

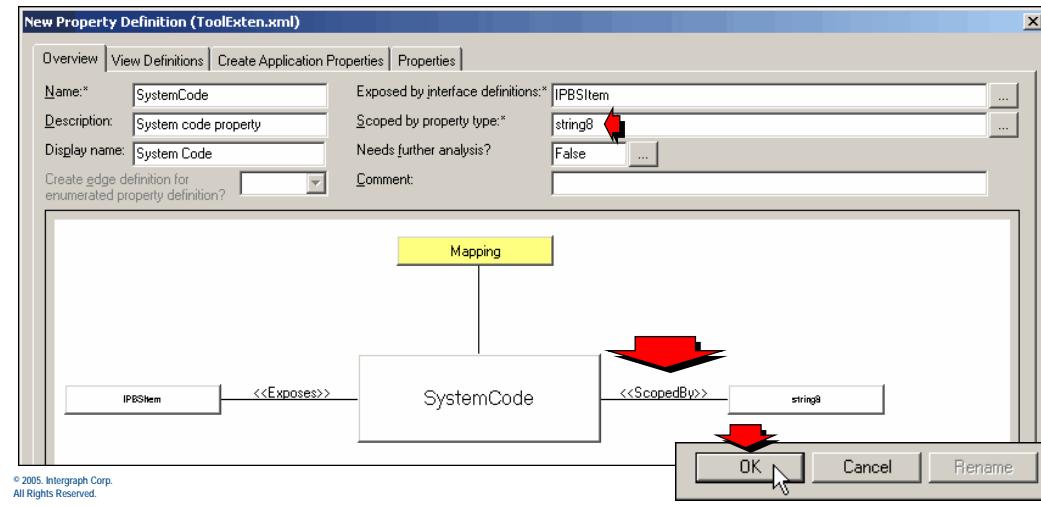
- Name** – Specifies the name of the new property definition.
- Description** – Specifies a description for the property definition.
- Display Name** – Specifies the display name for the property definition. The display name is used wherever the property definition appears in the user interface. If no display name is defined, the name is used instead.
- Exposed by interface definitions** – Specifies the interface definitions that expose this property definition.





Adding a New SmartPlant Property

- Set the **Scoped by property type** and click **OK** to finish the new property definition.



- **Scoped by property type** – Specifies whether the property definition is scoped by an enumerated list or a unit of measure (UoM) list.

Click **OK** to complete the property definition creation.

8.3.2 Mapping New Properties

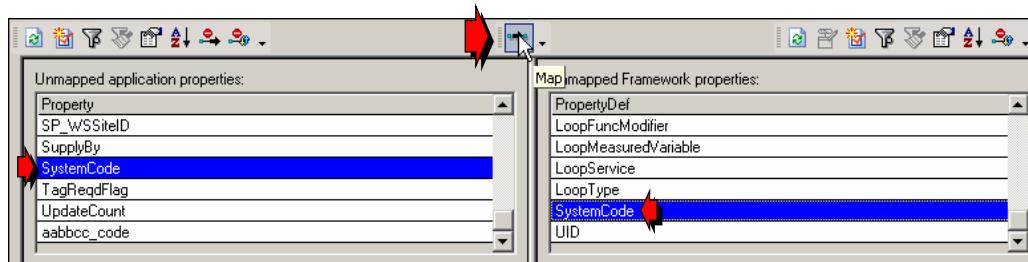
In the previous section, a new PropertyDef called **SystemCode** with a property type of *string* was added to the SmartPlant schema. Before defining mapping for this new PropertyDef using the Map Environment, the PropertyDef had to be added to the tool schema. This was done by the metadata adapter when the meta schema and the tool map schema were synchronized.

Clicking a map property in the tree control selects it from either the unmapped or mapped control.



Mapping New Properties

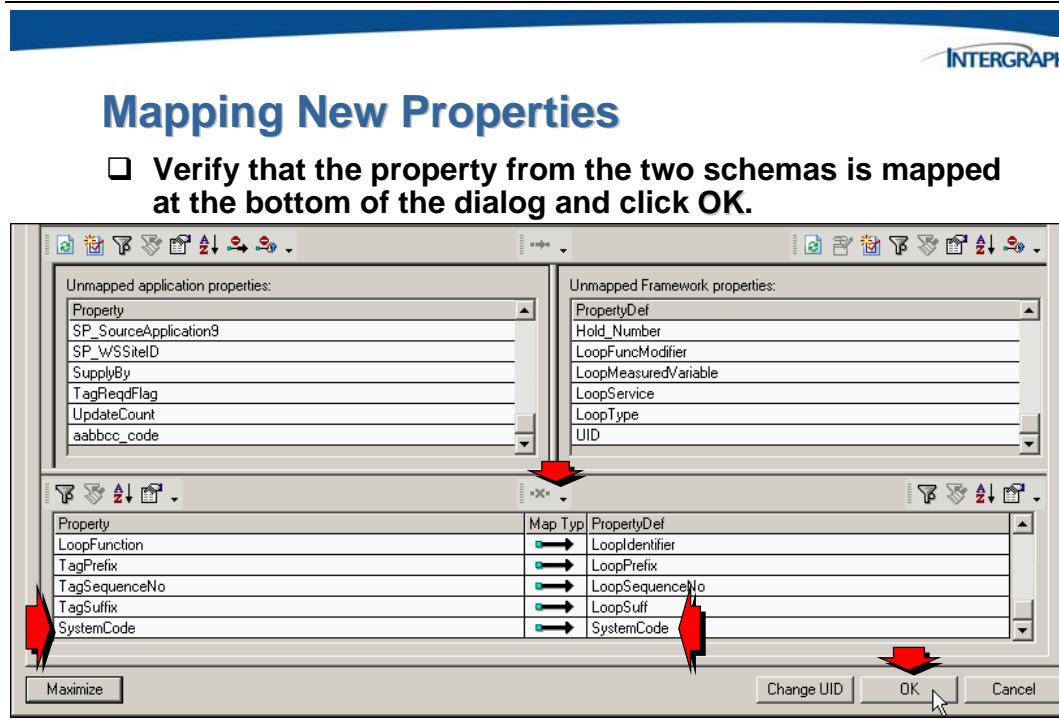
- Select the new **MapProperty** from the left pane (Tool Map schema and the corresponding **PropertyDef** from the right pane (SmartPlant schema) and click the *Map* button.



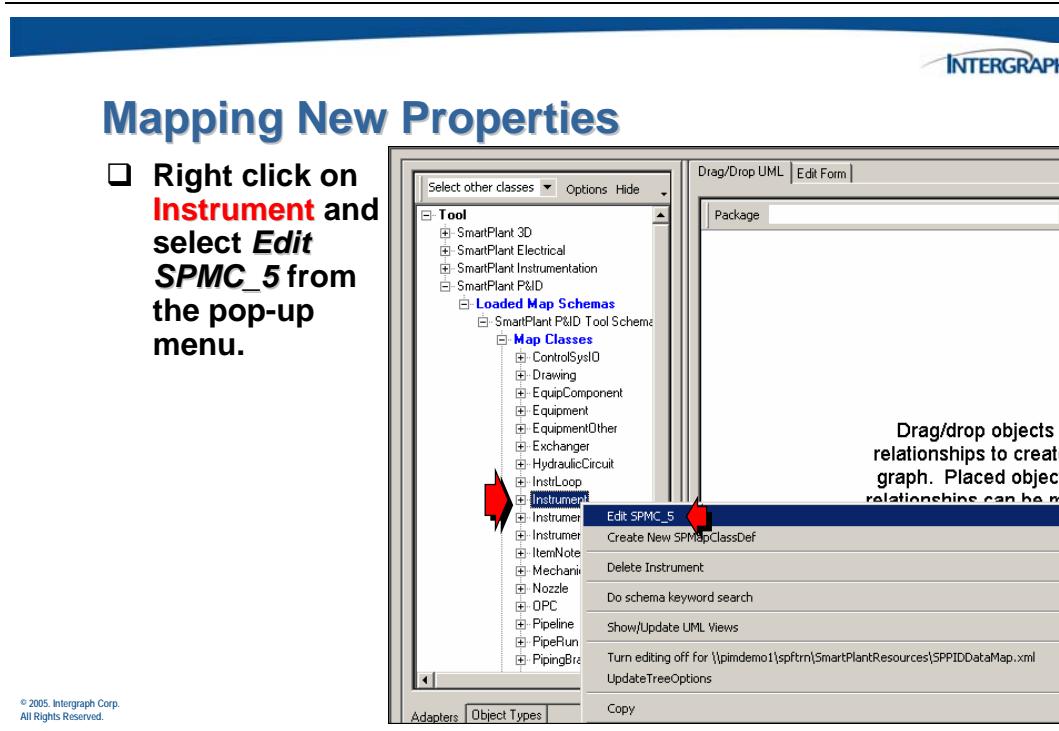
© 2005. Intergraph Corp.
All Rights Reserved.

The **Map** button is used to map an unmapped application property to an unmapped SmartPlant property definition.

The results of the mapping will be shown in the bottom control. The **Unmap** button will delete the mapping relationship for any of the selected rows in this control.



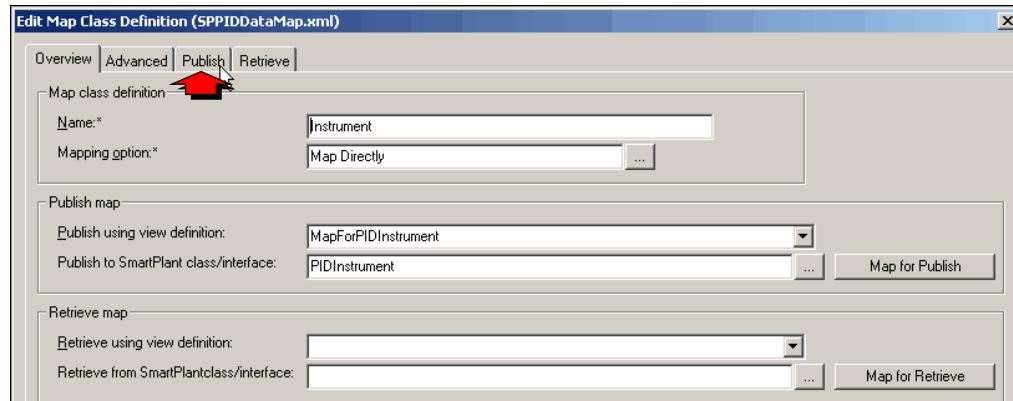
Since multiple P&ID classes inherited the **SystemCode** property, additional mapping can be performed.





Mapping New Properties

- From the *Edit Map Class Definition* dialog, click on the Publish tab.



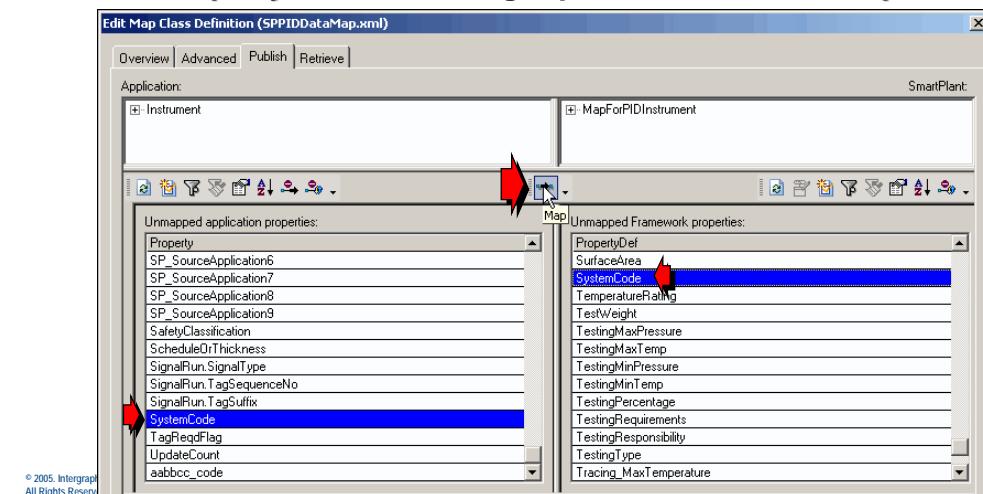
© 2005, Intergraph Corp.
All Rights Reserved.

In the middle control of the *Edit Map Class Definition* dialog, the unmapped SystemCode properties will be displayed.

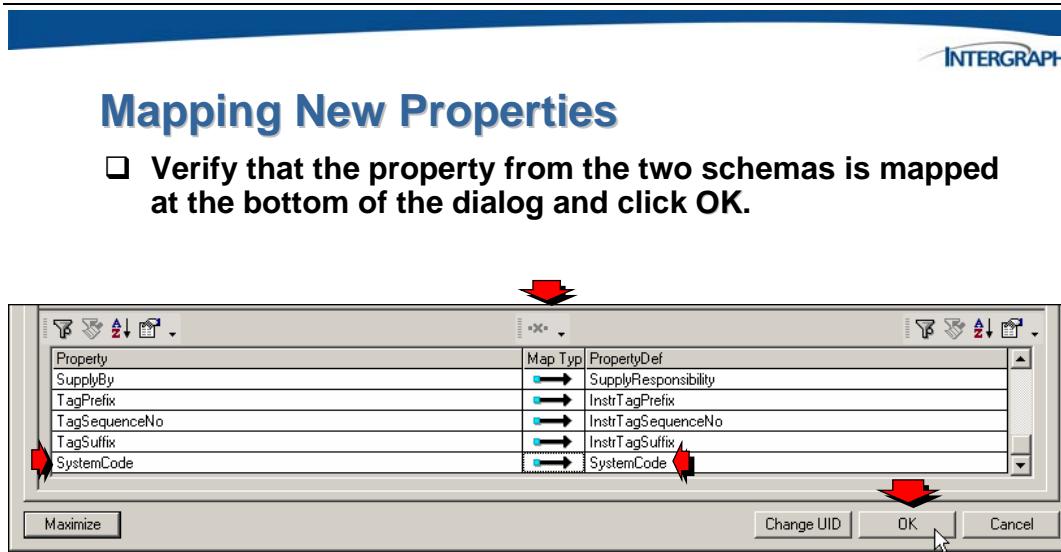


Mapping New Properties

- Select the new MapProperty from the left pane and the new PropertyDef from the right pane and click the *Map* button.

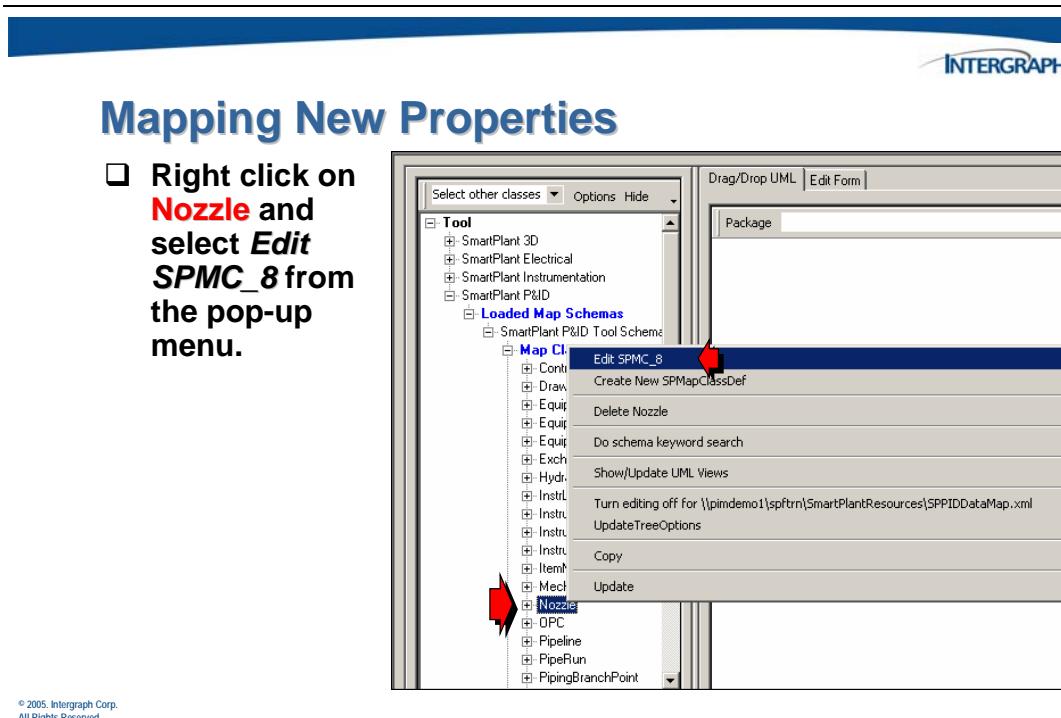


Once the properties have been mapped, they will display in the bottom control.



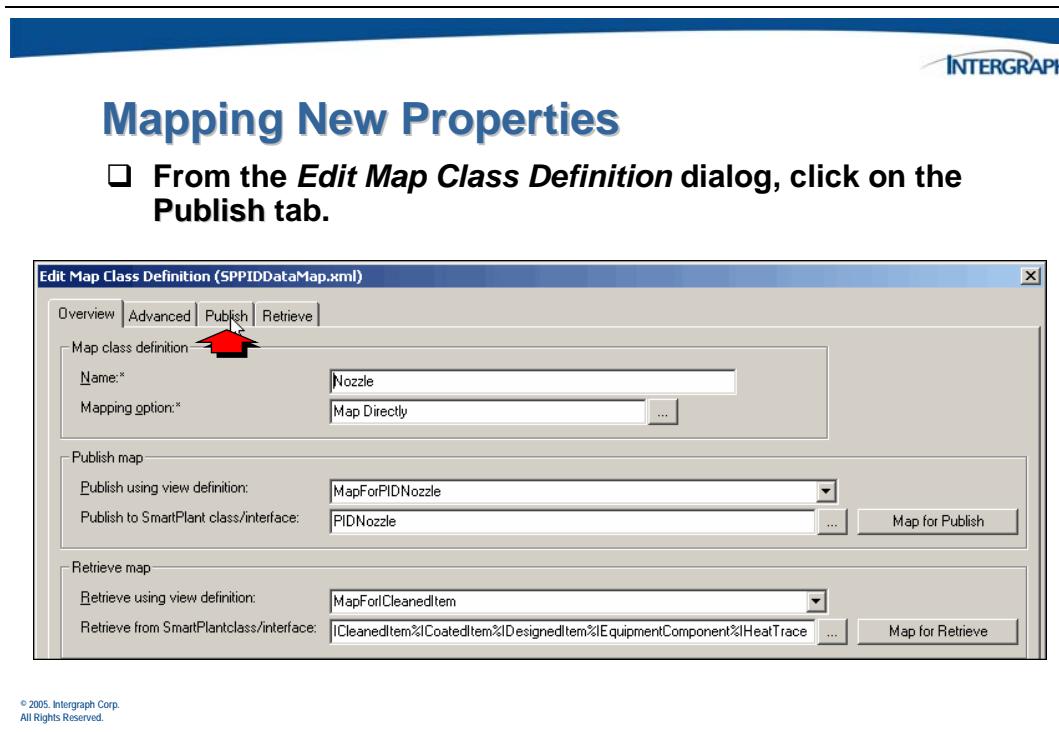
© 2005, Intergraph Corp.
All Rights Reserved.

Again, since multiple **SystemCode** properties are used with different classes, continue to map the SmartPlant schema with the SPPID tool map schema.

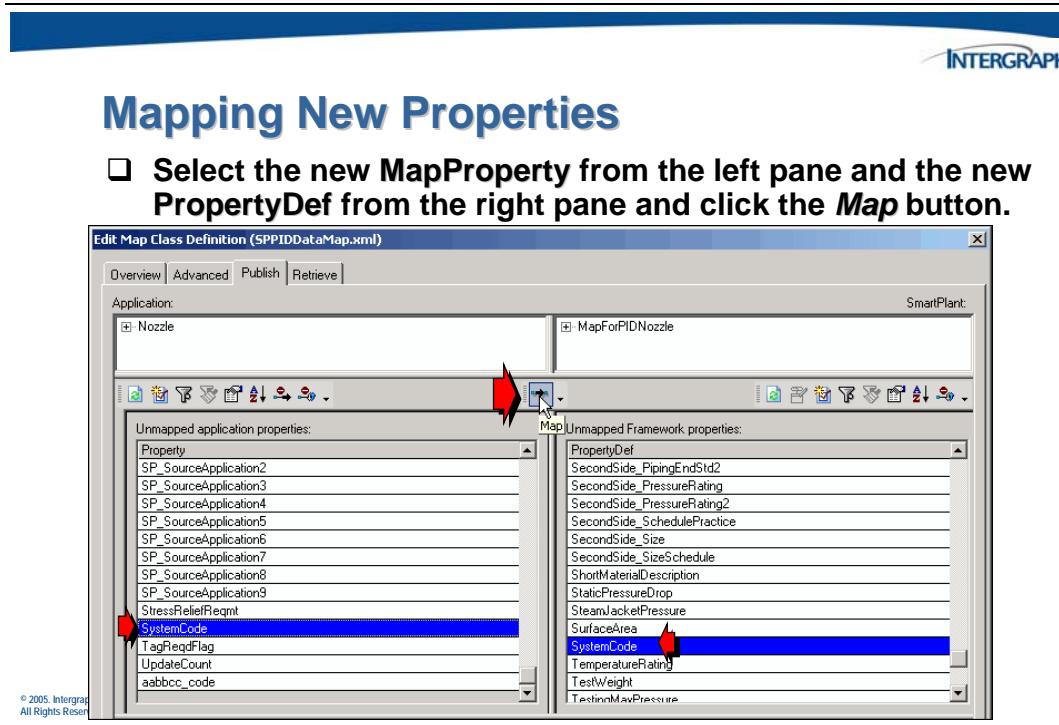


© 2005, Intergraph Corp.
All Rights Reserved.

The *Edit Map Class Definition* dialog box will display.



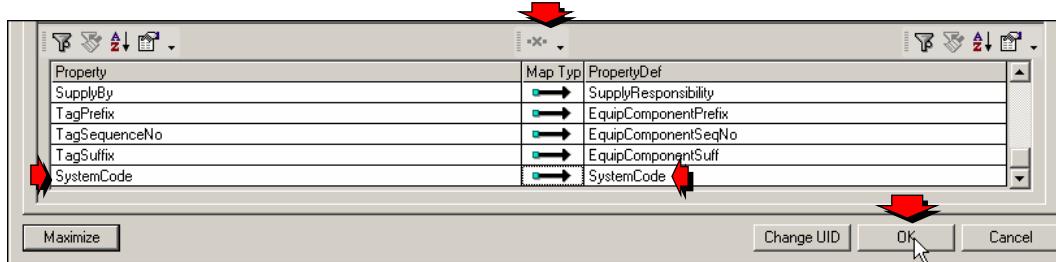
Perform the needed *Nozzle* to *PIDNozzle* mapping.





Mapping New Properties

- Verify that the property from the two schemas is mapped at the bottom of the dialog and click OK.



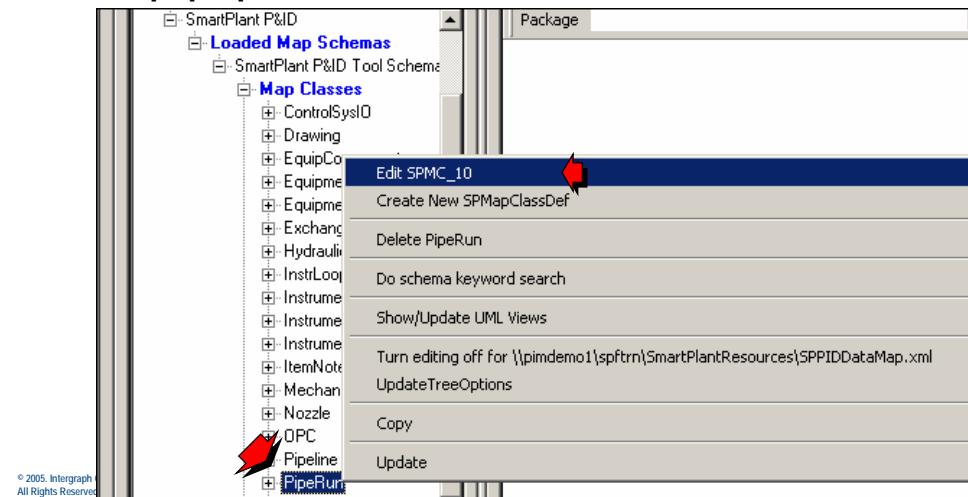
© 2005, Intergraph Corp.
All Rights Reserved.

Repeat the mapping process for any desired SmartPlant P&ID classes.

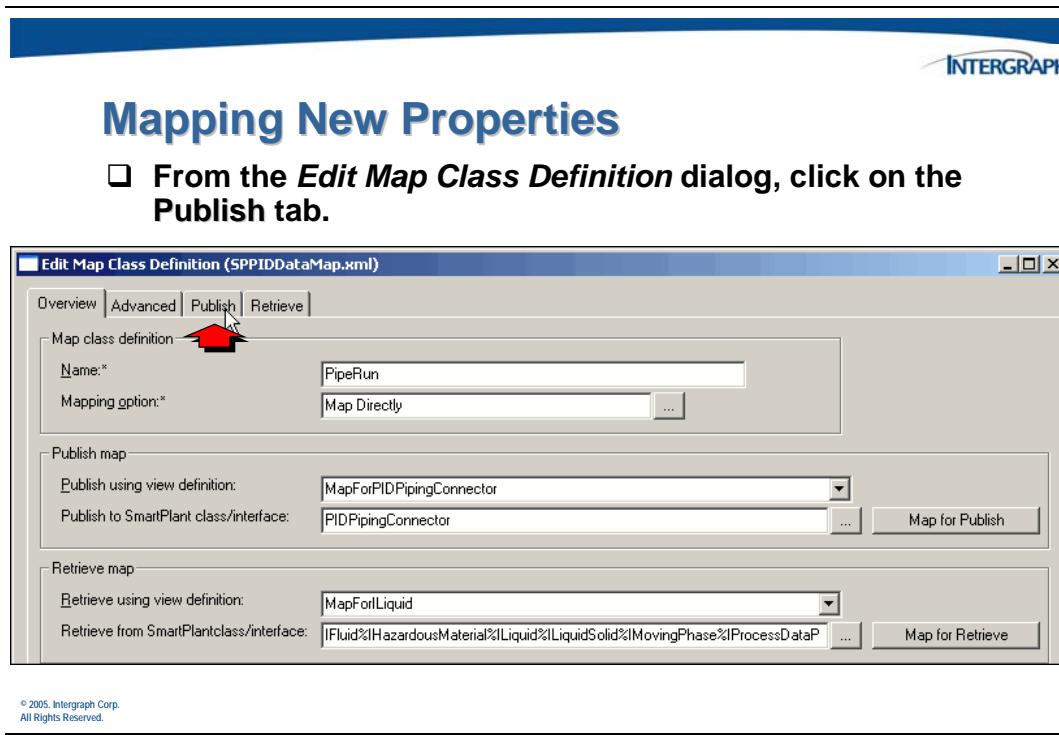


Mapping New Properties

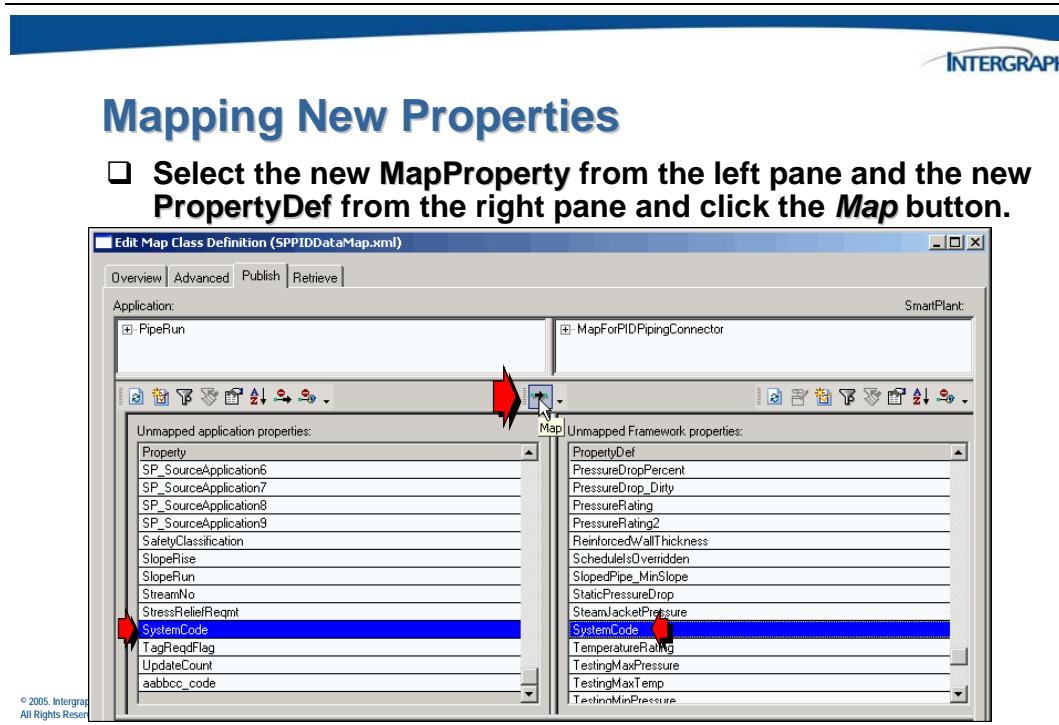
- Right click on **PipeRun** and select **Edit SPMC_10** from the pop-up menu.



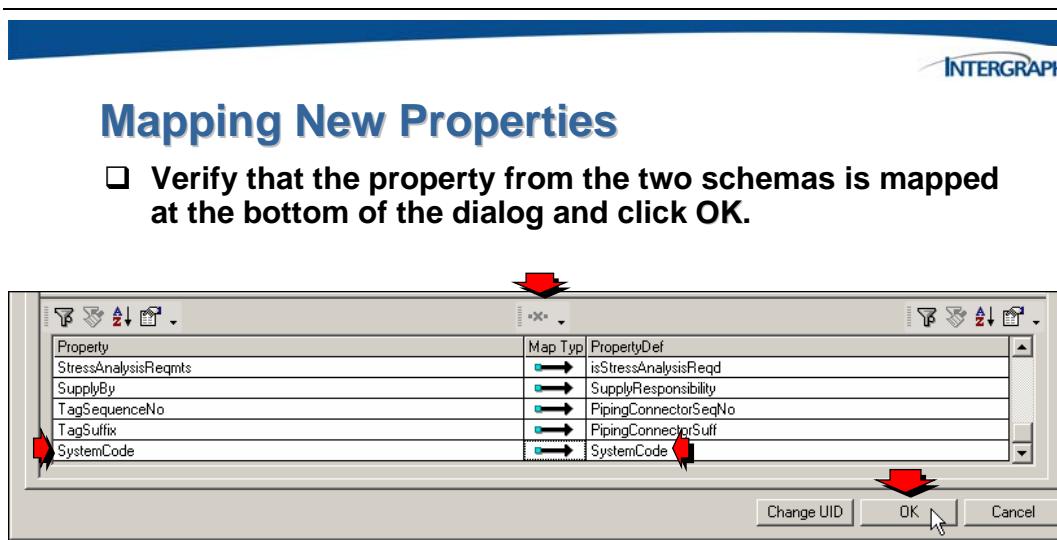
Once more, the *Edit Map Class Definition* dialog box will display.



Perform the needed *PipeRun* to *PIDPipingConnector* mapping.

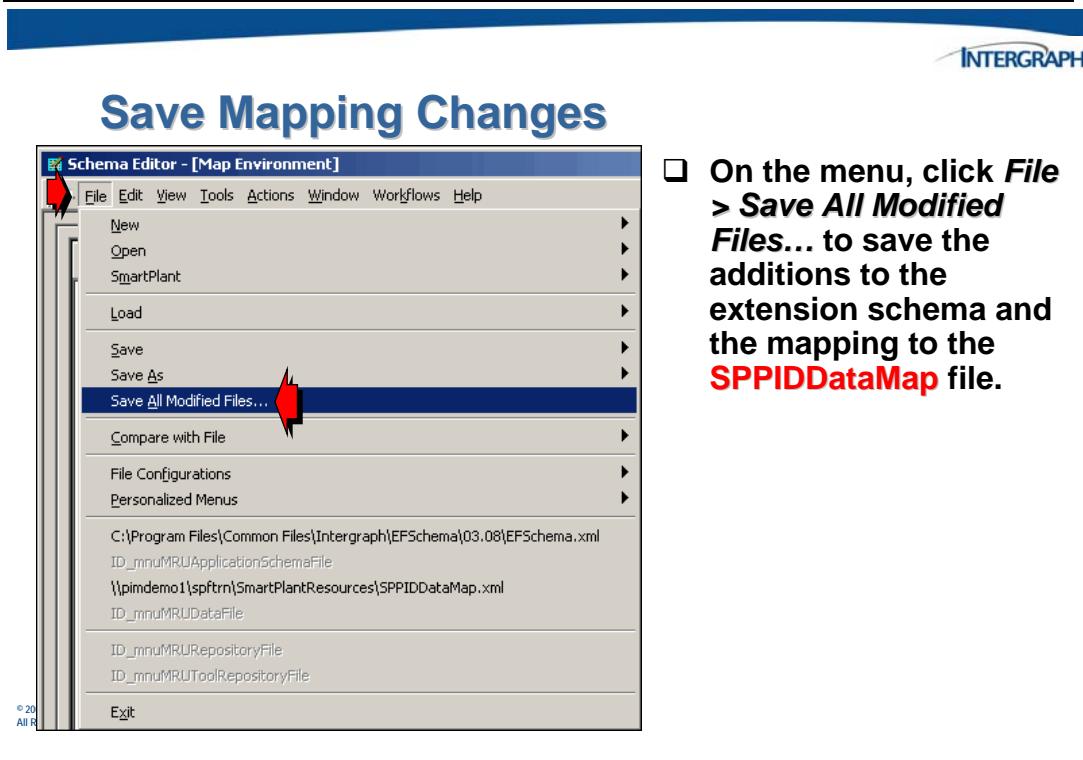


This creates the *PipeRun SystemCode* to *PIDPipingConnector SystemCode* property mapping.



8.3.3 Saving Mapping Changes

At this point, the additions to the SmartPlant schema and the mapping information is stored in memory will need to be saved to the respective xml files.



- On the menu, click *File* > *Save All Modified Files...* to save the additions to the extension schema and the mapping to the **SPPIDDataMap** file.**

You will be prompted to save the changes to the tool map schema.



Save Mapping Changes

- Verify the tool map schema name for the mapping changes and choose **Yes**.

© 2005, Intergraph Corp.
All Rights Reserved.

You will also be prompted to save the changes to the SmartPlant extension schema.

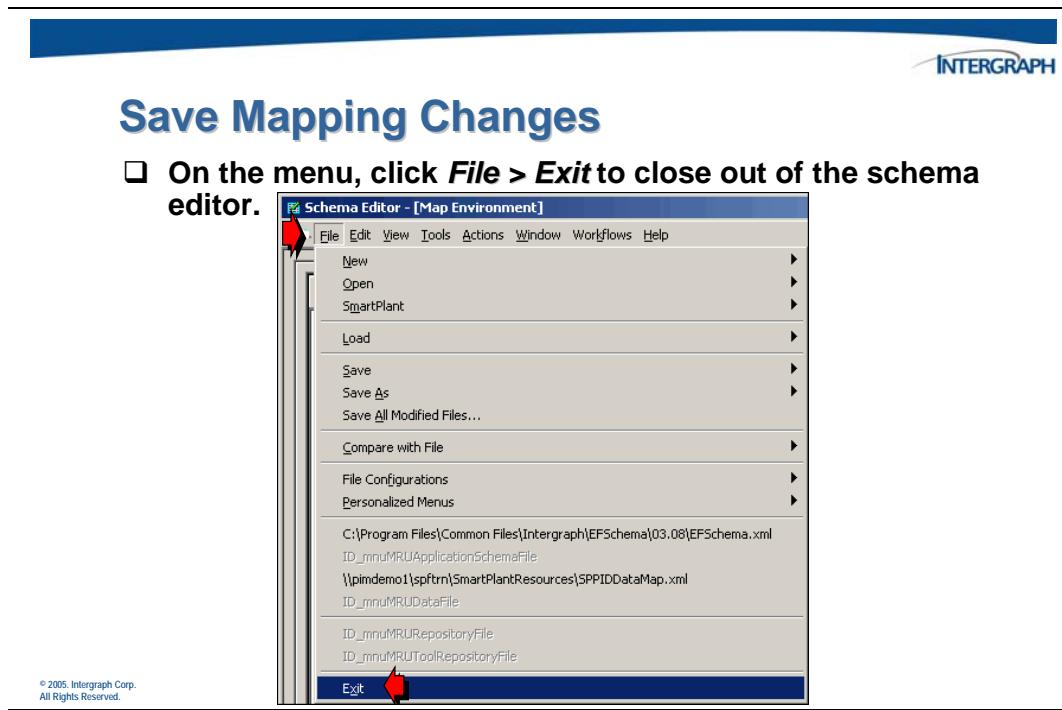


Save Mapping Changes

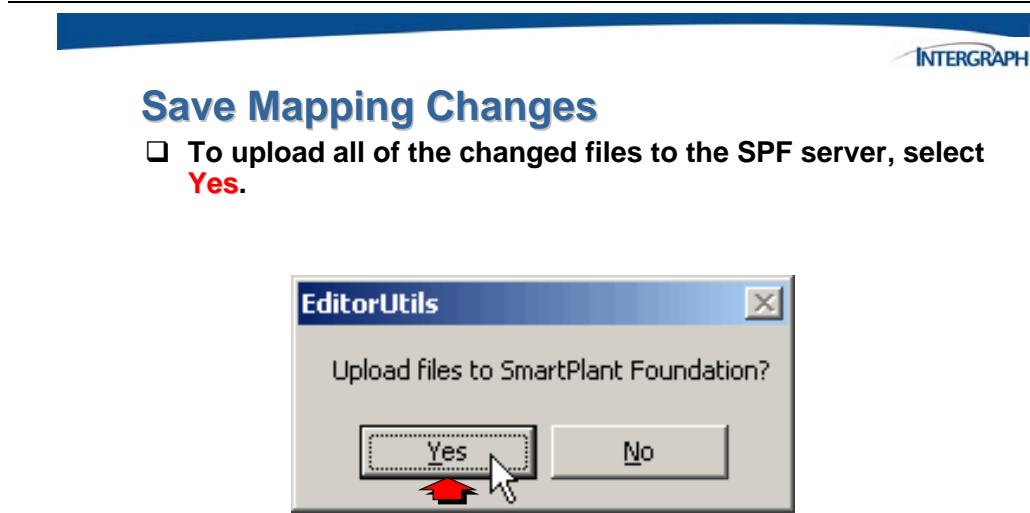
- Verify the extension schema name for the property addition and choose **Yes**.

© 2005, Intergraph Corp.
All Rights Reserved.

Once all of the schema files have been saved exit the Schema Editor.



The schema files have been copied to a temporary location in order to perform the additions and mapping. The last step is to upload these files to the schema location on the SmartPlant Foundation server.



© 2005, Intergraph Corp.
All Rights Reserved.

All schema files in the EFSchema configuration will be copied to the SmartPlant Foundation server.

8.4 Activity 1 – Adding and Mapping a Simple Property

The goal of this activity is to give you the opportunity to create a custom property in the SmartPlant P&ID application meta data, the SmartPlant P&ID tool map schema and the SmartPlant schema. You will then perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant engineering Manager > Data Dictionary Manager** to start the *SmartPlant Data Dictionary Manager*.
3. Open the Plant Item database table.
 - Click **Plant Item** in the Database Tables window.
 - Select **Edit> Add Property** from the menu.
 - In the *Add Property* dialog box, enter the following information:
 - Name – **SystemCode**
 - Display Name – **System Code**
 - Data Type – **String**
 - Maximum Length – **8**
 - Display to User – **Yes**
 - Use for Filtering – **Yes**
 - Category – **Miscellaneous**
 - Depends on – **None**
 - Click **OK** to close the *Add Property* dialog box.
4. Save the changes to the SmartPlant Data Dictionary Manager.
 - Click **File > Save**.
 - Click **File > Exit**.

5. Click **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
6. Open the schema XML files for analysis.
 - Click the **File Configurations** button and select **Open Configuration...** from the *Schema Editor Workflows* dialog
 - Select the **EFSchema.cfg** file and click **Open** (Path is C:\Program Files\Common Files\Intergraph\EF Schema\03.08, not the temporary web cache path).
7. View the open schema file using the **Tree/Drag-Drop UML** view.
 - On the *Workflows* dialog box, click the **View** button beside the **Another Schema File** button.
 - When the *View Schema* dialog box appears, select the **Tree/Drag-Drop UML** option and click **OK**.
8. In the base classes tree, expand the list of classes and then the **PIDInstrumentLoop** class. Expand and display the **Realized Interface Definitions**.
 - List the Realized Interface Definitions below:
 - _____
 - _____
 - _____
 - _____
9. Repeat this for the following classes and list their **Realized Interface Definitions**.
 - Class: **PIDNozzle**
 - Interfaces: _____
 - _____
 - _____
 - _____
 - Class: **PIDPipingComponent**
 - Interfaces: _____
 - _____

- _____
- _____
- Class: **PIDInstrument**
- Interfaces: _____
- _____
- _____
- _____
- Class: **PIDProcessEquipmentComponent**
- Interfaces: _____
- _____
- _____
- _____
- Class: **PIDPipingConnector**
- Interfaces: _____
- _____
- _____
- _____

10. List the common interfaces for all classes from steps 8 and 9 above.

- _____
- _____

Use one of these interfaces as the expose relationship for any new properties to be added to the schema.

11. Exit and re-start the *Schema Editor* by selecting *Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor*.

12. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant...**

13. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with.
 - Click **Next**.
 - Select the tool map schema to be loaded, *SMARTPLANTPID/SmartPlant P&ID Tool Schema*.
 - Enable the **Load map schema** toggle.
 - Enable the **Connect to application schema** toggle.
 - Click **Finish**.
14. When the *Synchronize* dialog displays, confirm that the new property values have been imported by the meta data adapter and click **OK**.
15. Add a new property to the SmartPlant schema to correspond to the new property that has been added to a tool meta schema and tool map schema.
 - ❑ Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant P&ID Tool Schema entries are displayed.
 - ❑ Expand the **Map Classes** and right-click on the *InstrLoop* entry in the tree and choose **Edit SPMC_6** from the dynamic menu.
 - ❑ Select the **Publish** tab in the *Edit Map Map Class Definition* dialog. Expand the tree for both the application/tool map schema and the SmartPlant schema in the upper control. Highlight **Identification** in the SmartPlant tree.
 - ❑ Select the **New Property Definition** button **below** the SmartPlant tree in the upper control.
 - ❑ In the *New Property Definition* dialog box, enter the following information:
 - Name – **SystemCode**
 - Description – **System code property**
 - Display Name – **System Code**
 - Click  next to the *Exposed by interface definitions* box, UNSELECT the *IInstrumentLoop primary interface* and then select the **IPBSItem InterfaceDef** (if that was on your list from step 10) from the *Possible ExposedByInterfaceDefs for SystemCode* dialog box. The *Exposed by interface definitions* field should contain only ONE entry.

- Click  next to the *Scoped by property type* box and select the **string8** property type from the *Possible ScopedByPropertyType for SystemCode* dialog box.
 - Click **OK** to close the *New Property Definition* dialog box.
16. Highlight the *SystemCode* MapProperty in the **Unmapped applications properties** control and the *SystemCode* MapProperty in the **Unmapped SmartPlant properties** control in order to perform the schema mapping.
- Make sure that **SystemCode** is highlighted in the middle control in the application section and that **SystemCode** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPPID tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *SystemCode* maps to *SystemCode* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Class Definition* dialog box.
17. Repeat the mapping for the following classes:
- **EquipComponent**
 - **Instrument**
 - **Nozzle**
 - **Pipeline**
 - **PipeRun**
 - **PipingComp**
18. Save the changes to the all schema files.
- From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **Yes**.
19. Use Windows Explorer to verify that all the files were uploaded to **C:\Program Files\Common\Intergraph\EFSchema\03.08** with the latest time and date.

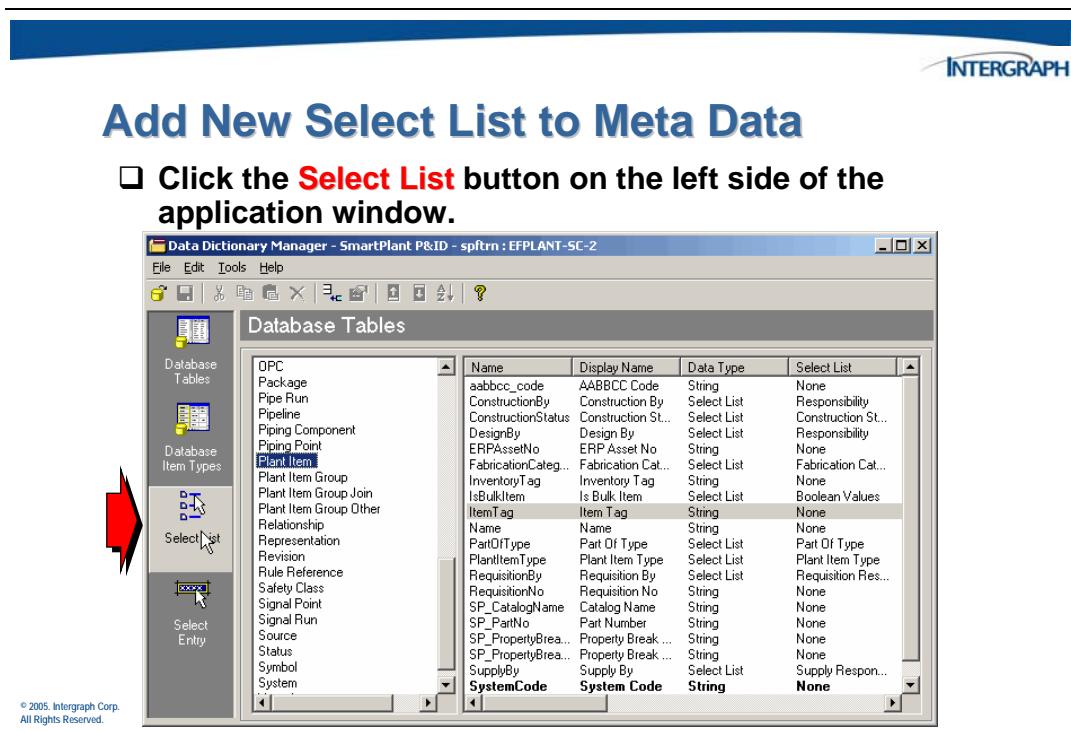
20. When you are finished with this activity, you may take a short break until the other students have finished.

8.5 Adding a New Select List/Enum List

In the last section, the task of extending the SmartPlant P&ID meta data/SmartPlant Schema and adding a new simple property was demonstrated. A simple property is a property definition that uses a simple property type such as a string property type.

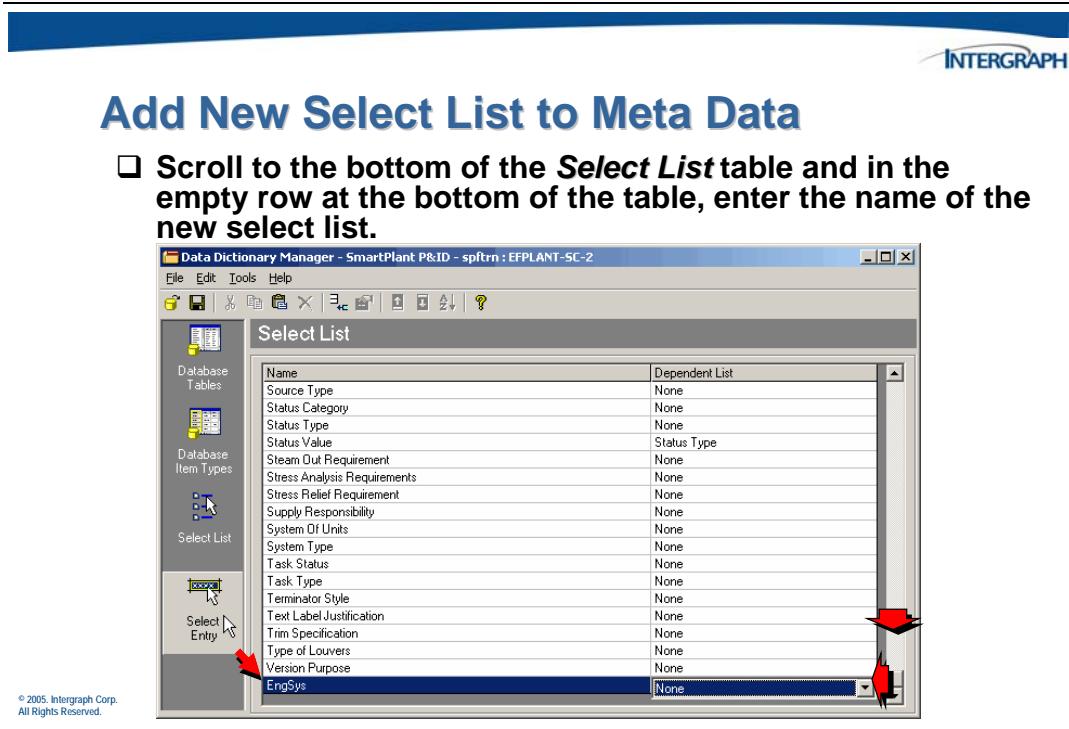
Another property type that is useful in SmartPlant is the **Select List**, which is a picklist. This is a more complex type of property. The select list will equate to an **Enumerated List** in the tool map schema and the SmartPlant schema.

To add a new property to SPPID that uses a select list, begin by starting the **Data Dictionary Manager**.

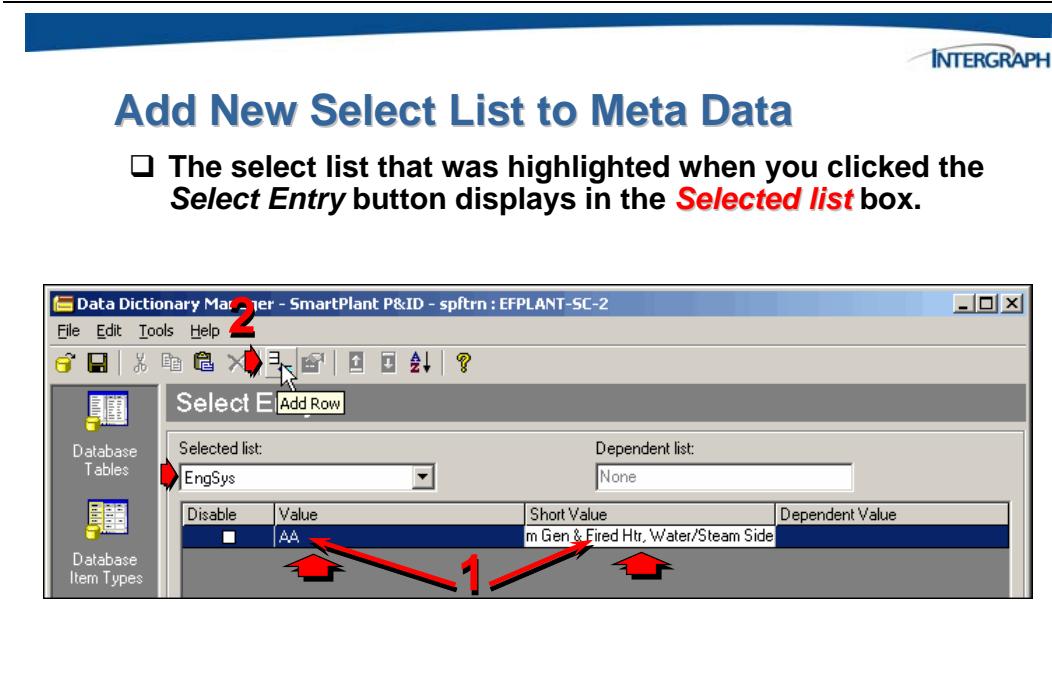


Next, define a property that has a select list associated with it. Before you create the property, create the select list that will be associated with the property.

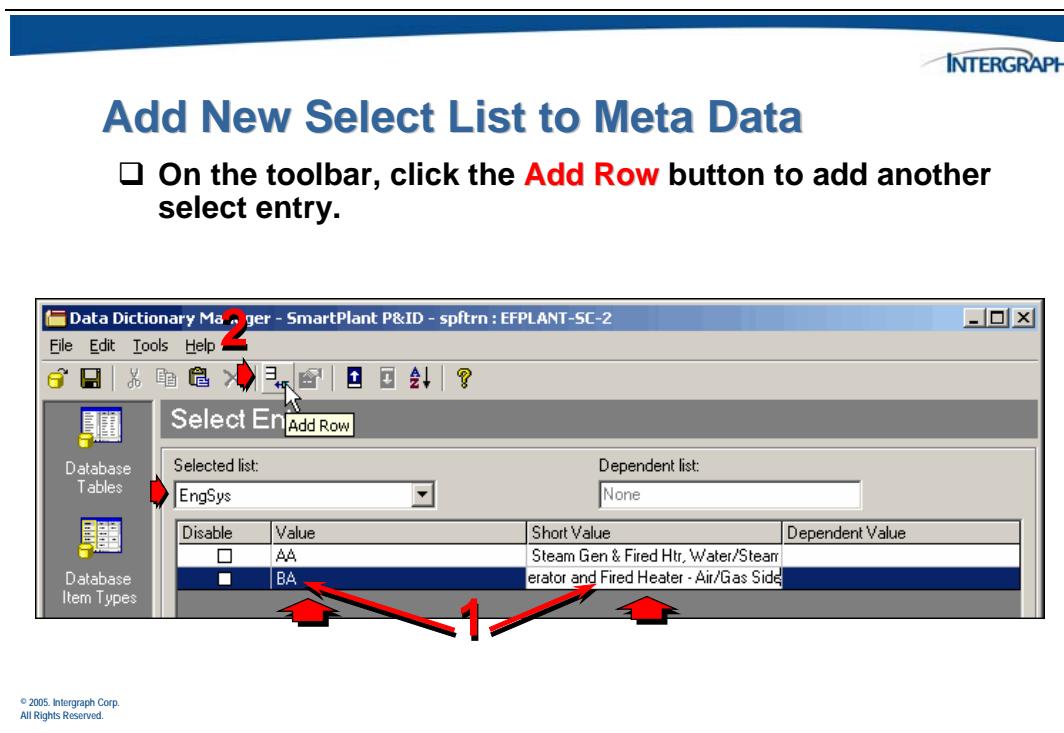
You can add a new select list by typing the name of the list in the blank row at the bottom of the *Select List* table.



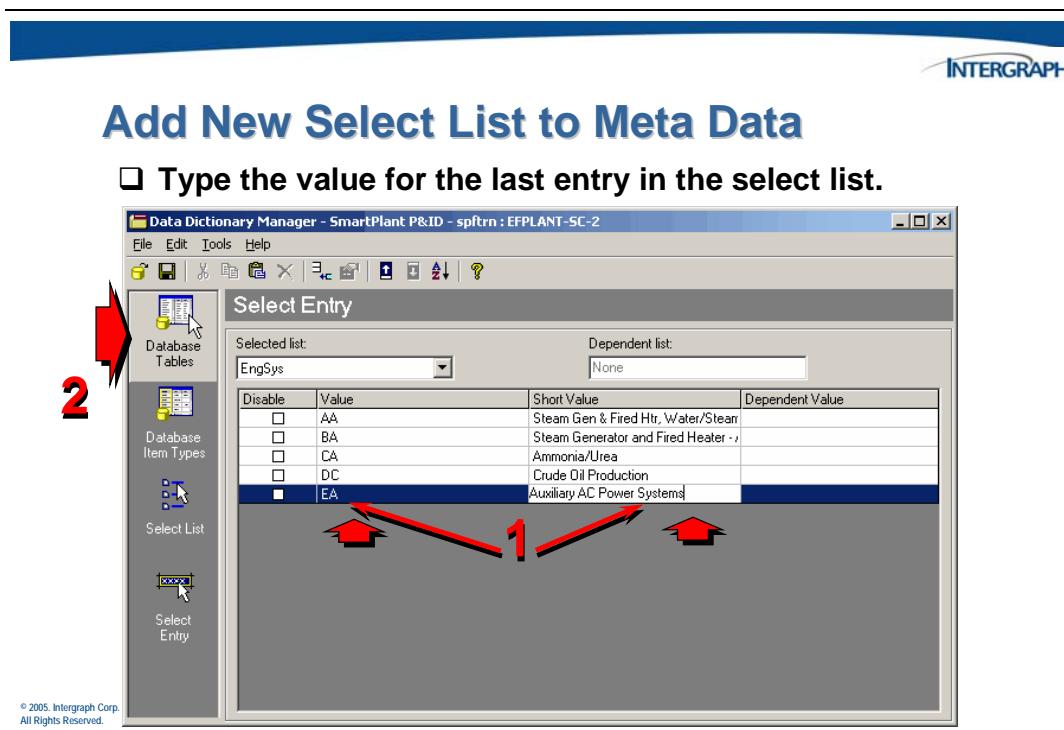
All that is required to define a new select list is the name of the select list. To define entries for the new select list, click the *Select Entry* button. Add the first select entry to the list by typing it in the first row of the table.



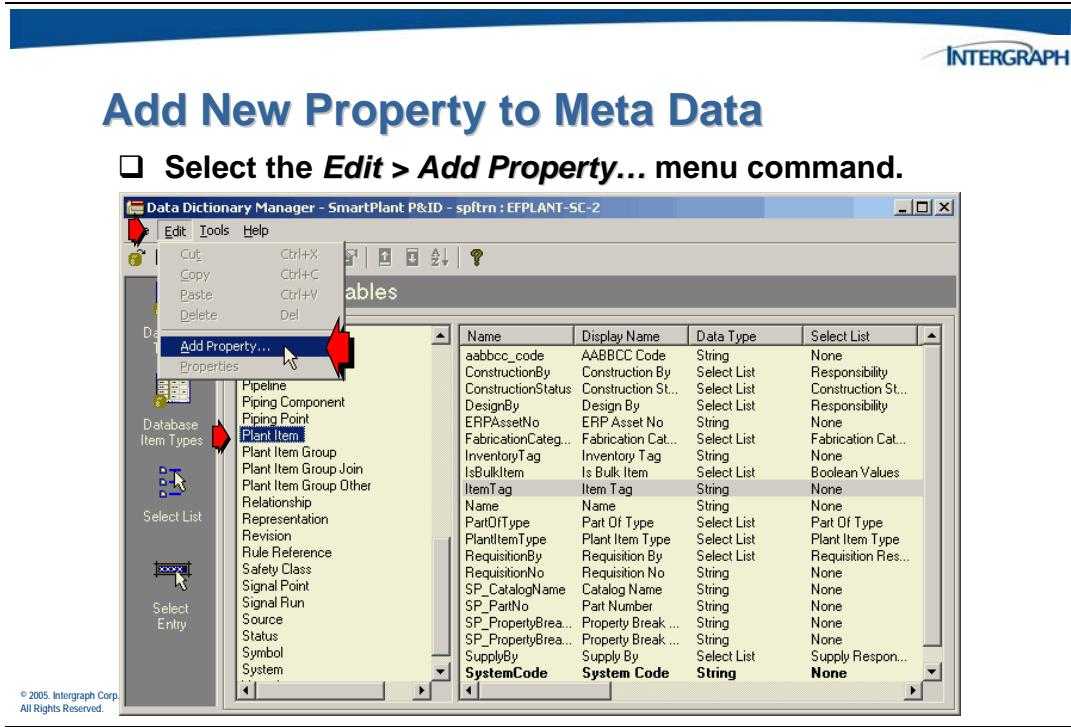
Define a second select entry for the select list.



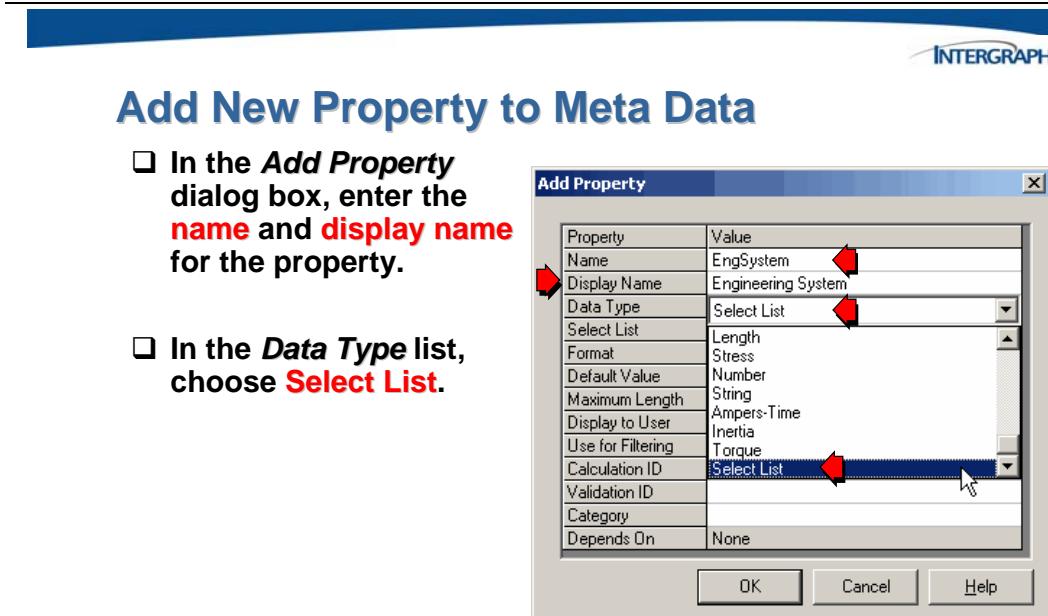
Define the remaining select list entries.



After you add the last entry to the select list, create the property with which you want to associate the select list. To create the property, click the *Database Tables* button then select *Plant Item* in the list of database tables.



This will add the new **EngSystem** property to the *Plant Item* table. Choosing **Select List** as the data type allows you to select the appropriate list in the *Select List* box.



The new property will use the custom select list, EngSys, defined earlier.

Add New Property to Meta Data

In the **Select List** list, choose the name of the select list you want to associate with the property.

Property	Value
Name	EngSystem
Display Name	Engineering System
Data Type	Select List
Select List	EngSys
Format	Task Status
Default Value	Task Type
Maximum Length	Terminator Style
Display to User	Text Label Justification
Use for Filtering	Trim Specification
Calculation ID	Type of Louvers
Validation ID	Version Purpose
Category	EngSys
Depends On	None

OK Cancel Help

© 2005, Intergraph Corp.
All Rights Reserved.

Finish specifying the new property characteristics.

Add New Property to Meta Data

In the **Category** list, choose **Process**.

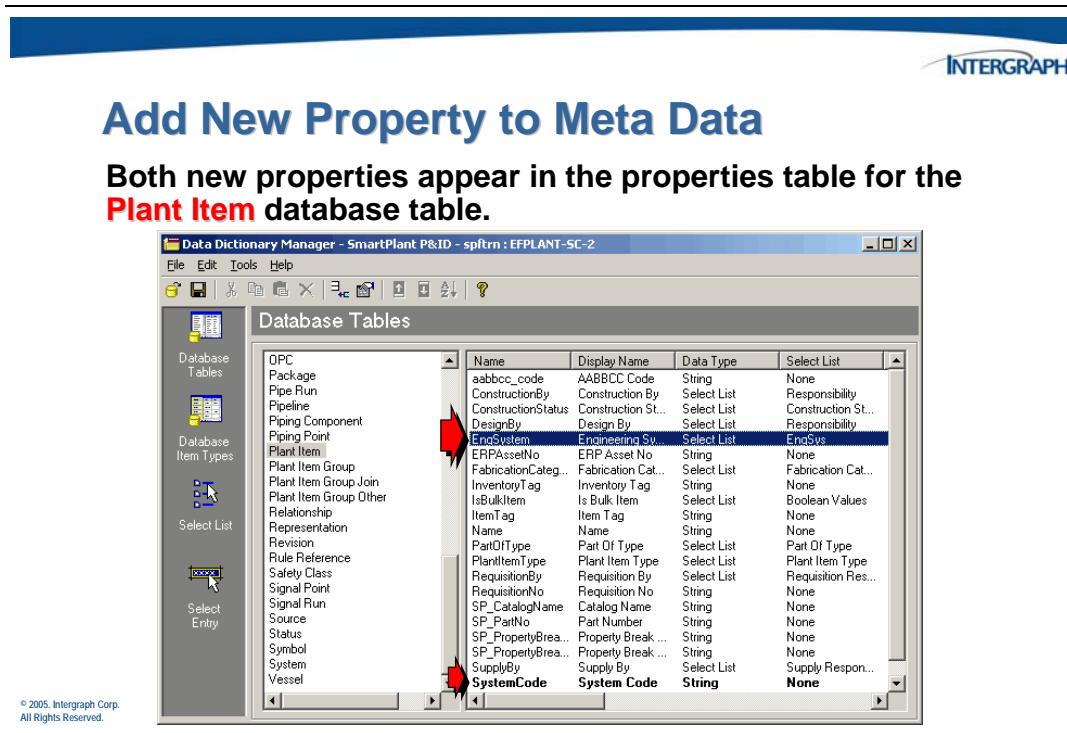
Click **OK** to save the new property.

Property	Value
Name	EngSystem
Display Name	Engineering System
Data Type	Select List
Select List	EngSys
Format	Variable length
Default Value	None
Maximum Length	Yes
Display to User	Yes
Use for Filtering	
Calculation ID	
Validation ID	
Category	Process
Depends On	None

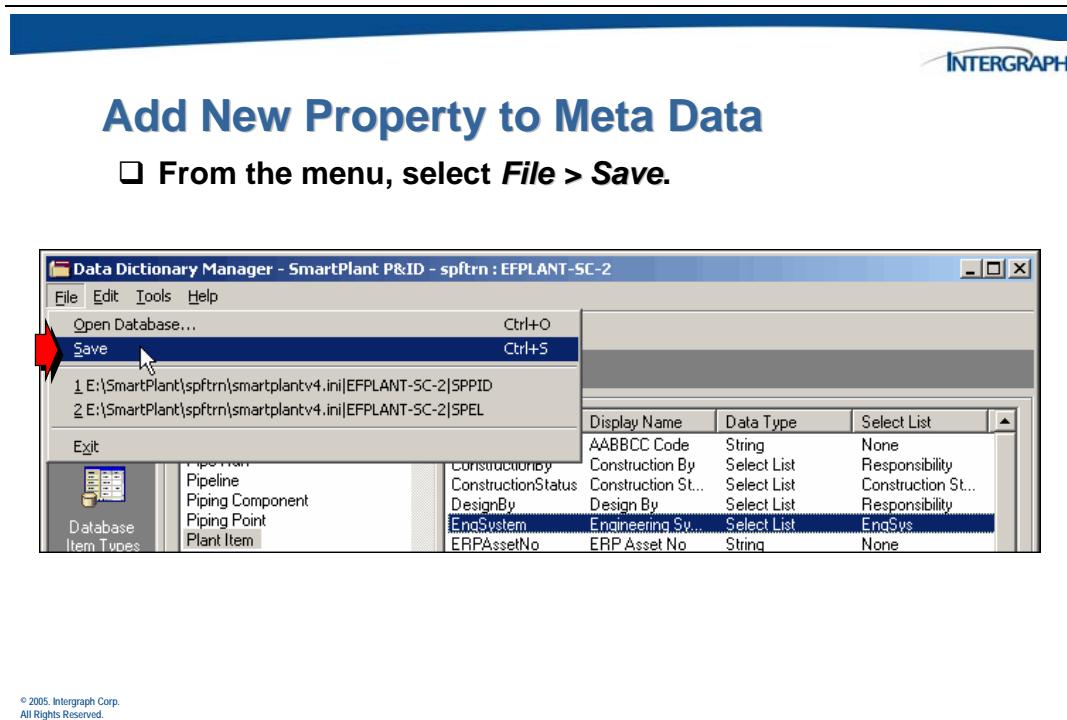
OK Cancel Help

© 2005, Intergraph Corp.
All Rights Reserved.

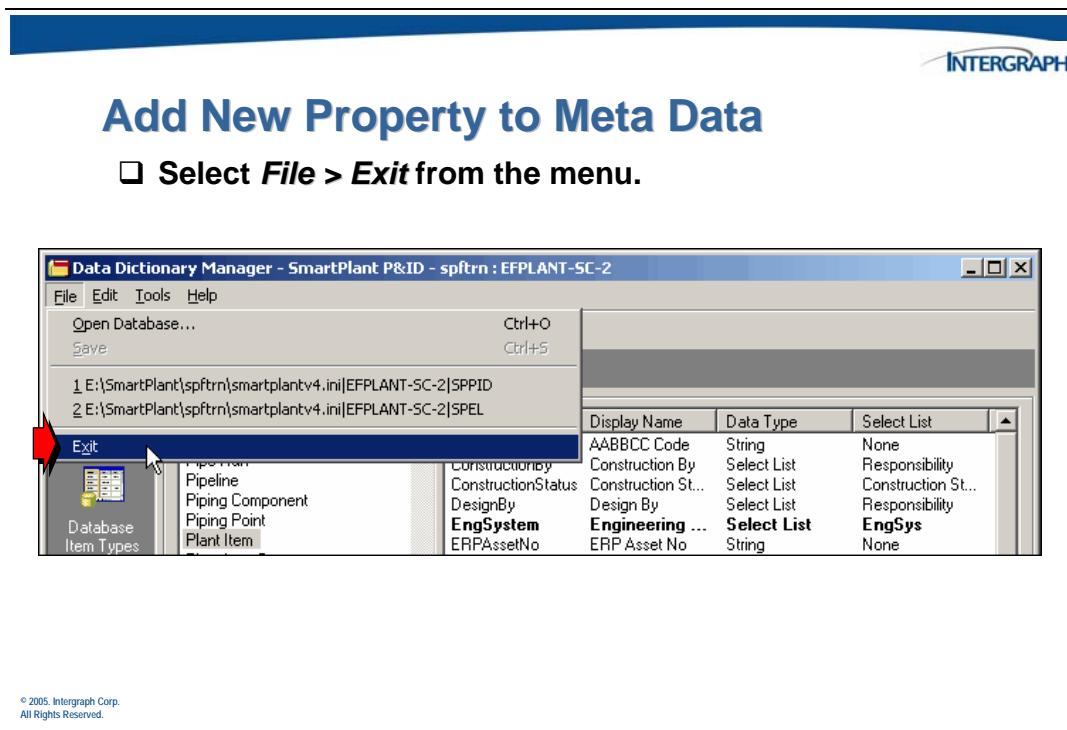
When you click **OK**, the list of properties for the *Plant Item* table updates with the new property.



After adding the property, save the changes to the SmartPlant P&ID meta data.



When you click **File > Exit**, this will close Data Dictionary Manager.



8.5.1 Adding Enum Extensions to the Schemas

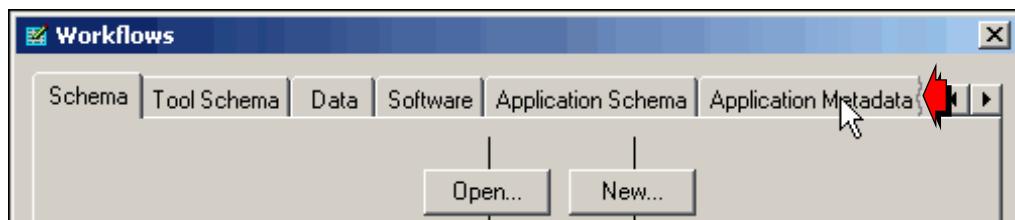
Once the new property/select list has been added to the SPPID meta data, use the Schema Editor to add the new property to the SmartPlant schema.

Start the Schema Editor and connect to SmartPlant.

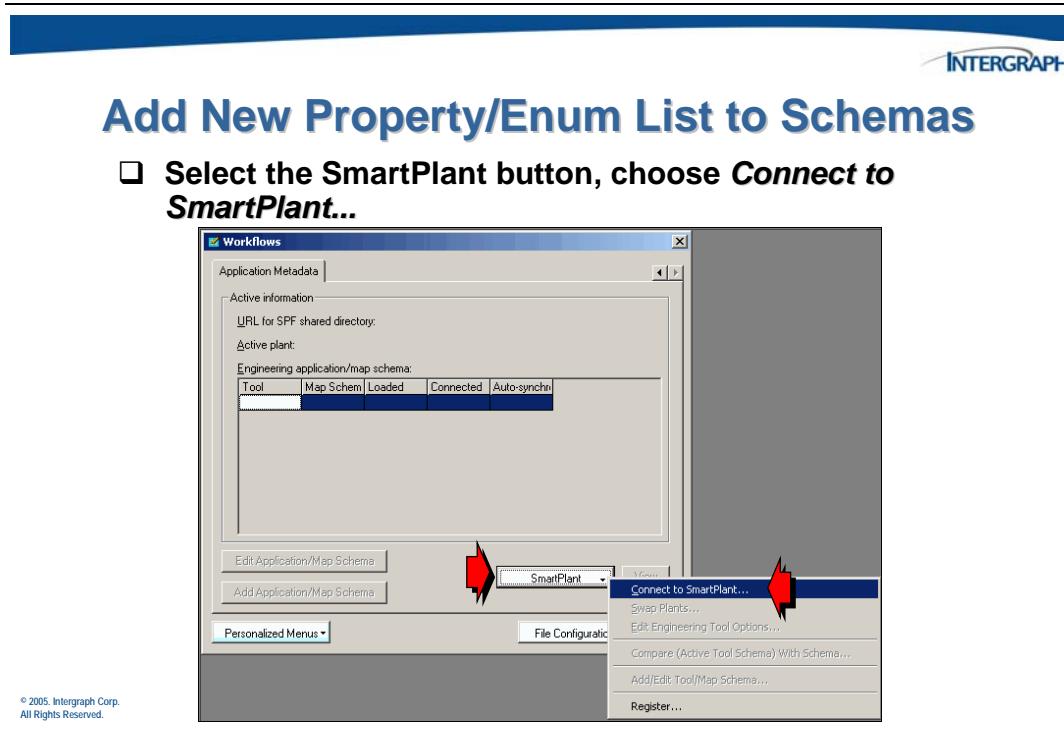


Add New Property/Enum List to Schemas

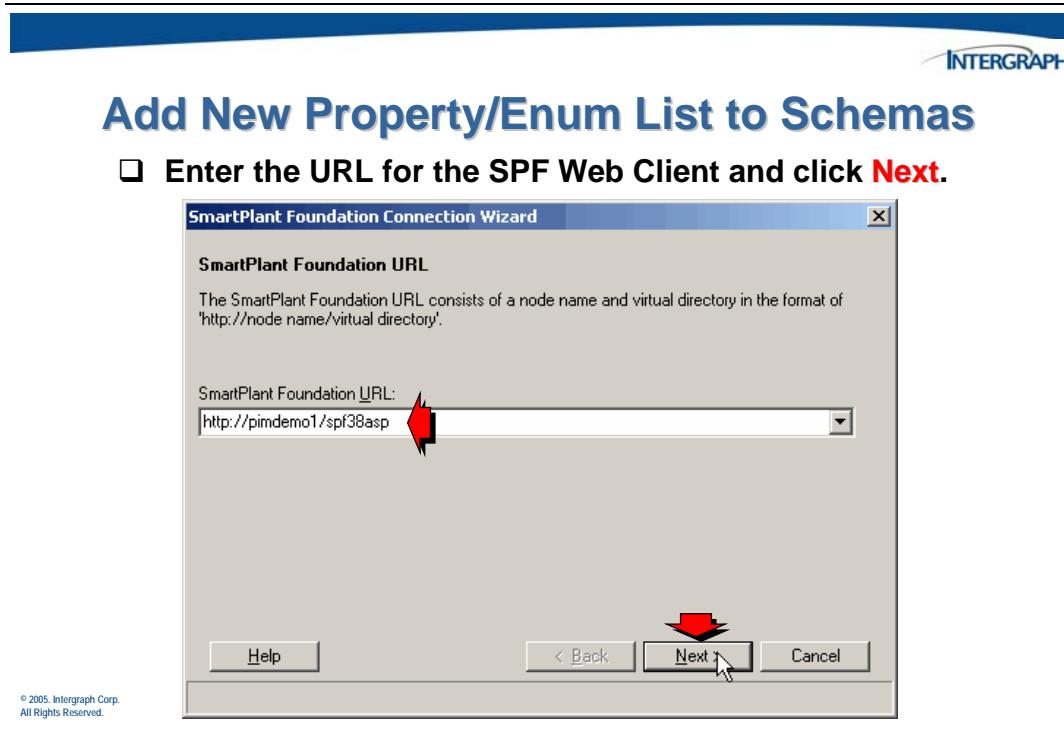
- Click on the **Application Metadata** tab in the Workflows dialog to connect to SmartPlant.



The *Application Metadata* dialog will display.



The *SmartPlant Foundation Connection Wizard* will be displayed.

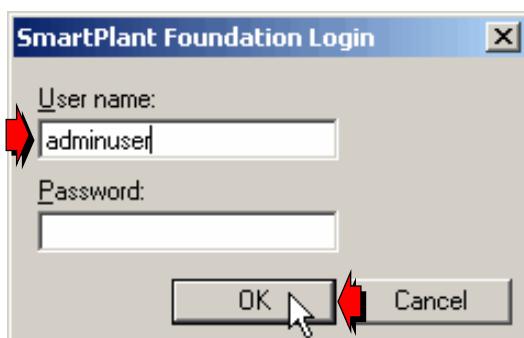


In the *SmartPlant Foundation URL* field, select the SmartPlant Foundation database with which you want to connect and click **Next**.



Add New Property/Enum List to Schemas

- Enter a valid SPF *User name* and *Password* to log on to the SPF server from the schema editor.



© 2005, Intergraph Corp.
All Rights Reserved.

Select the SmartPlant Foundation plant configuration that you will use for schema extension and mapping.



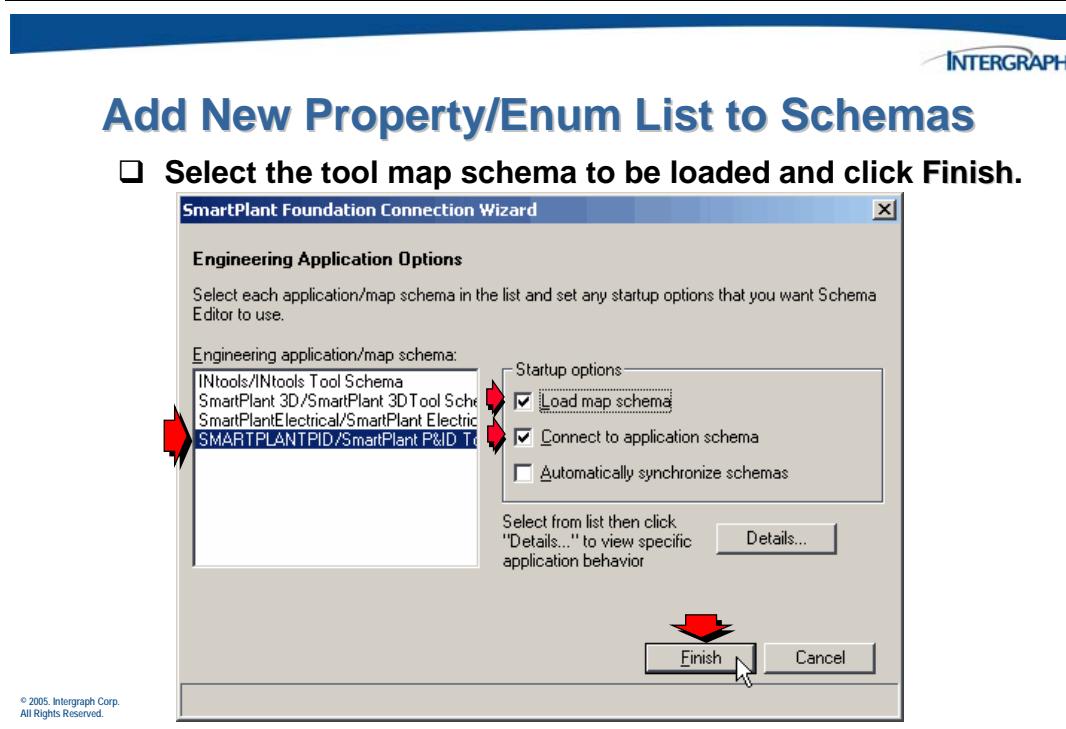
Add New Property/Enum List to Schemas

- Select the plant configuration to be used for the schema mapping and click **Next**.



© 2005, Intergraph Corp.
All Rights Reserved.

The Schema Editor will find the SmartPlant schema and all **registered** tool map schemas.



After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema.

The differences between the two schemas are displayed in the **Synchronize** dialog.



Add New Property/Enum List to Schemas

Click OK from the Synchronize dialog to have the schema editor read the tool meta-schema and automatically add the new property/enum extensions to the tool map schema.

Synchronize

For each row below select option that should be applied

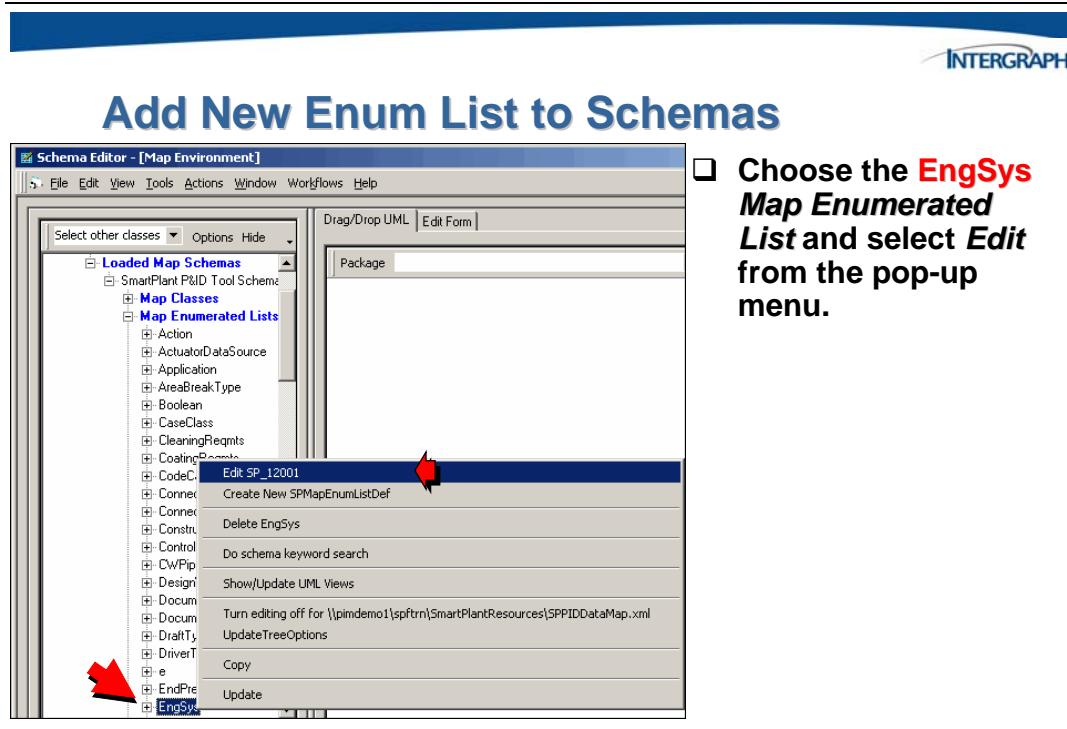
Class	Parent Name	Name	Property	SMARTPLANTPID	Value
SPMapPropertyDef	EquipComponent	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	EquipmentOther	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Exchanger	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	HydraulicCircuit	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	InstLoop	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Instrument	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Mechanical	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Nozzle	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	PipeRun	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Pipeline	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	PipingComp	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	PipingPoint	PlanItem EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	SignalPoint	PlanItem EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	SignalRun	EngSystem	Exists - Delete it	Missing - Add it	
SPMapPropertyDef	Vessel	EngSystem	Exists - Delete it	Missing - Add it	
SPMapEnumListDef		EngSys	Exists - Delete it	Missing - Add it	
SPMapEnumDef	EngSys	AA	Exists - Delete it	Missing - Add it	
SPMapEnumDef	EngSys	BA	Exists - Delete it	Missing - Add it	
SPMapEnumDef	EngSys	CA	Exists - Delete it	Missing - Add it	
SPMapEnumDef	EngSys	DC	Exists - Delete it	Missing - Add it	
SPMapEnumDef	EngSys	EA	Exists - Delete it	Missing - Add it	

OK

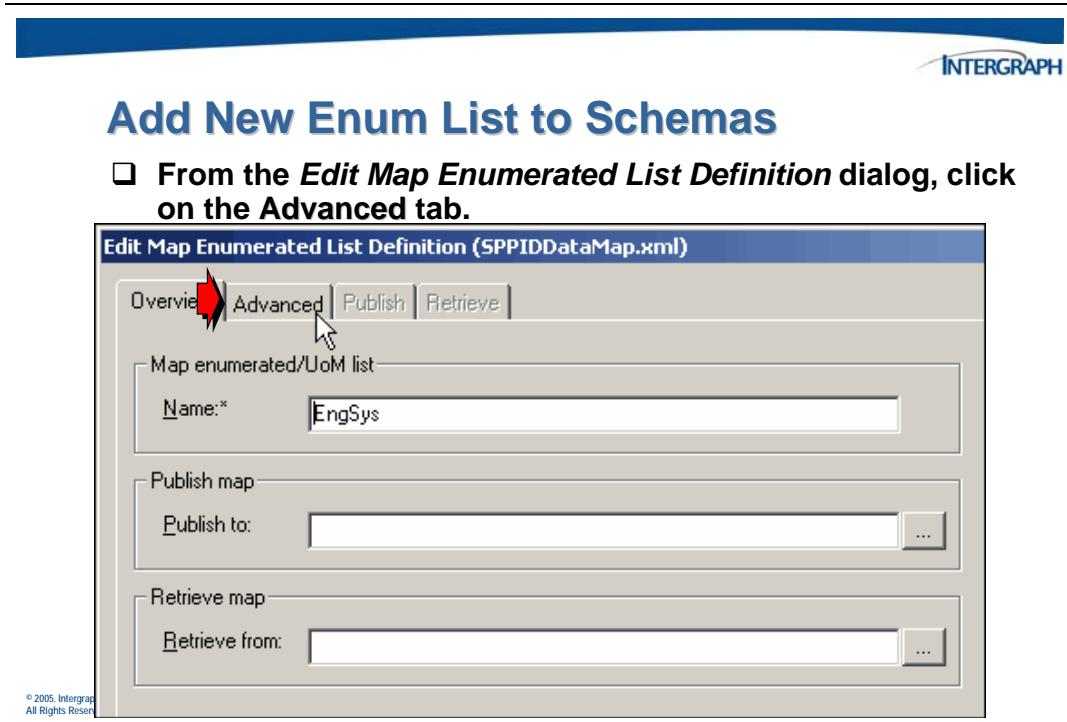
© 2005, Intergraph Corp.
All Rights Reserved.

In the above example, you will note all the instances of the **EngSystem** SPMapProperty and the **EngSys** SPMapEnum have been parsed for many of the SmartPlant P&ID classes.

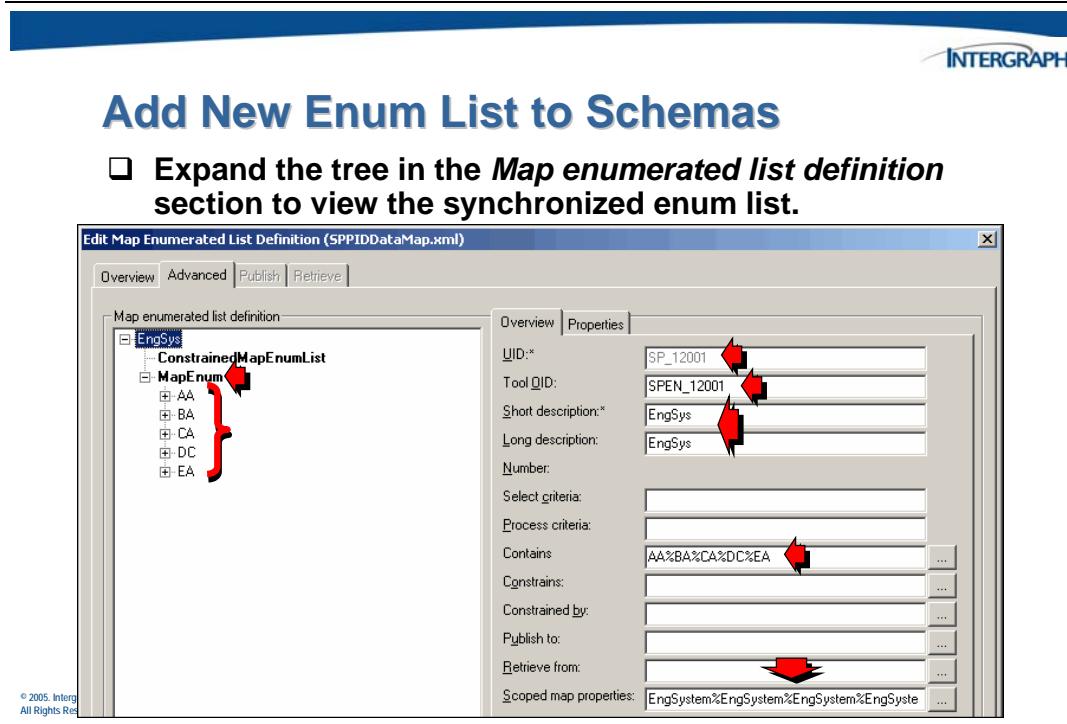
Next, edit the new SPMMapEnum list that has been added to the SPPID tool map schema as a result of the synchronization.



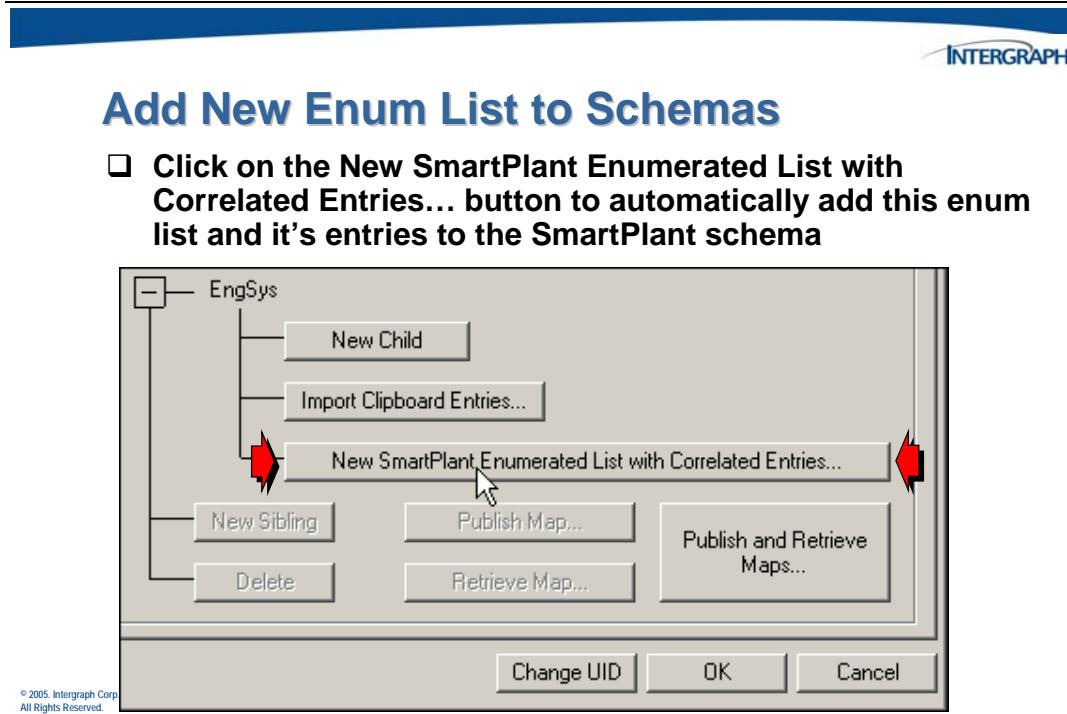
The automated correlation between the tool map schema and the SmartPlant schema can be accessed by selecting the **Advanced** tab.



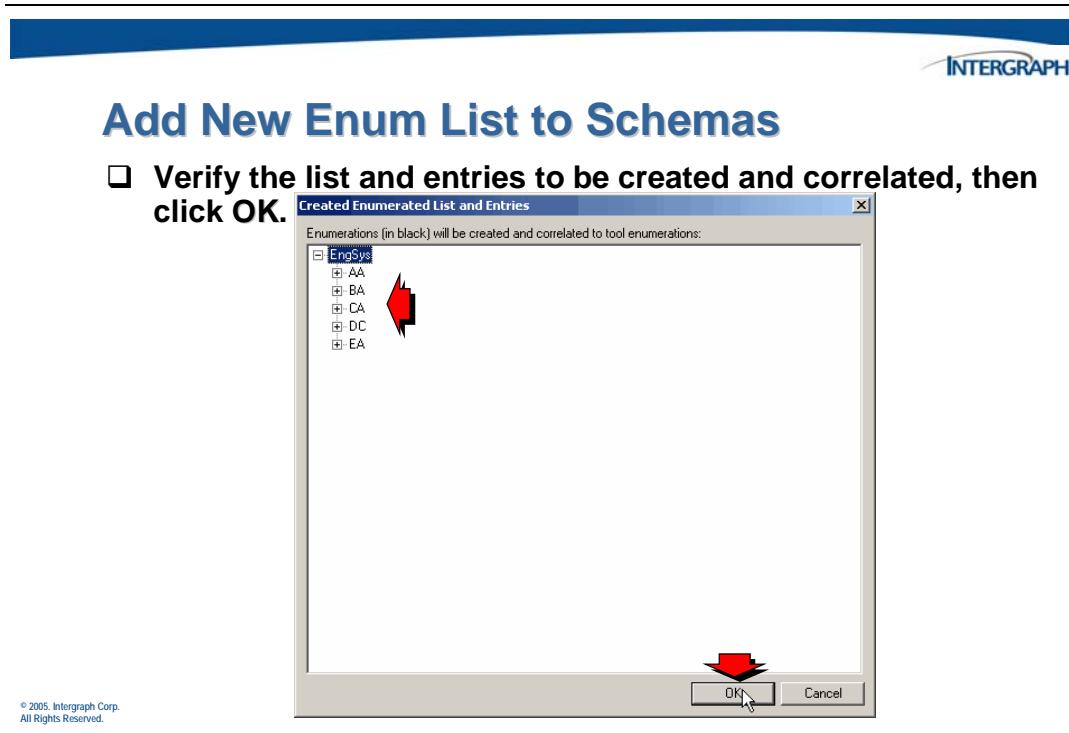
The results of the synchronization can be seen when the *Edit Map Enumerated List Definition* dialog displays.



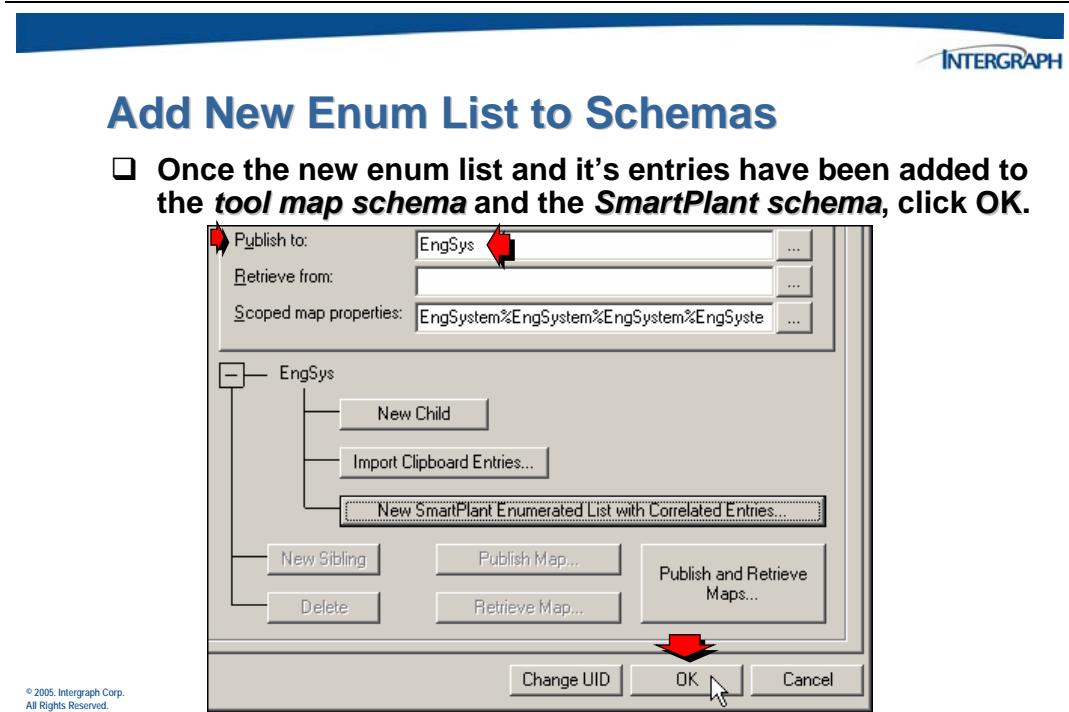
The additions that have been made to the tool map schema can be automatically added to the SmartPlant schema and mapped.



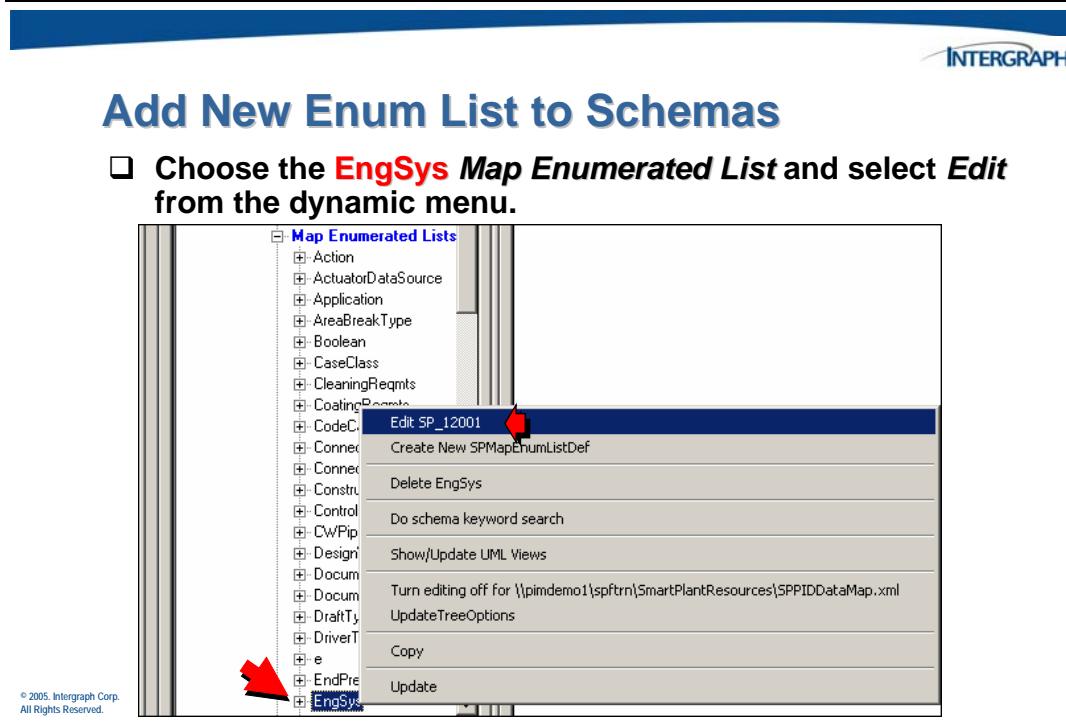
The *Created Enumerated List and Entries* dialog will display the changes that will be created in the SmartPlant schema.



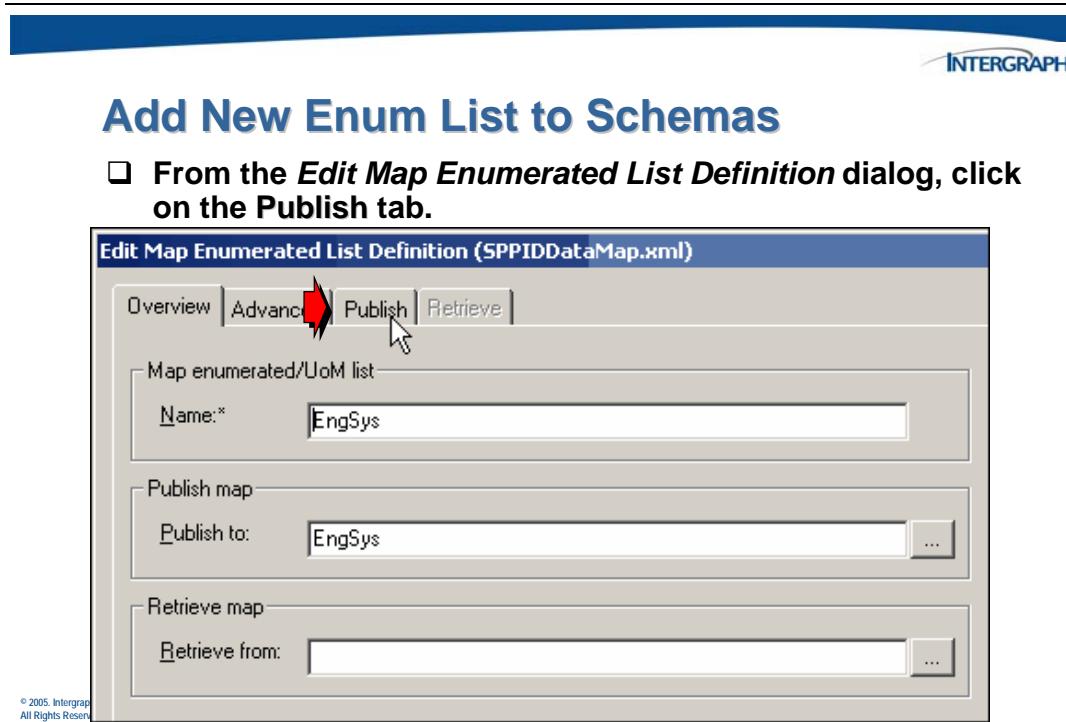
The new enum list/entries will also be correlated (mapped) between the tool map schema and the SmartPlant schema.



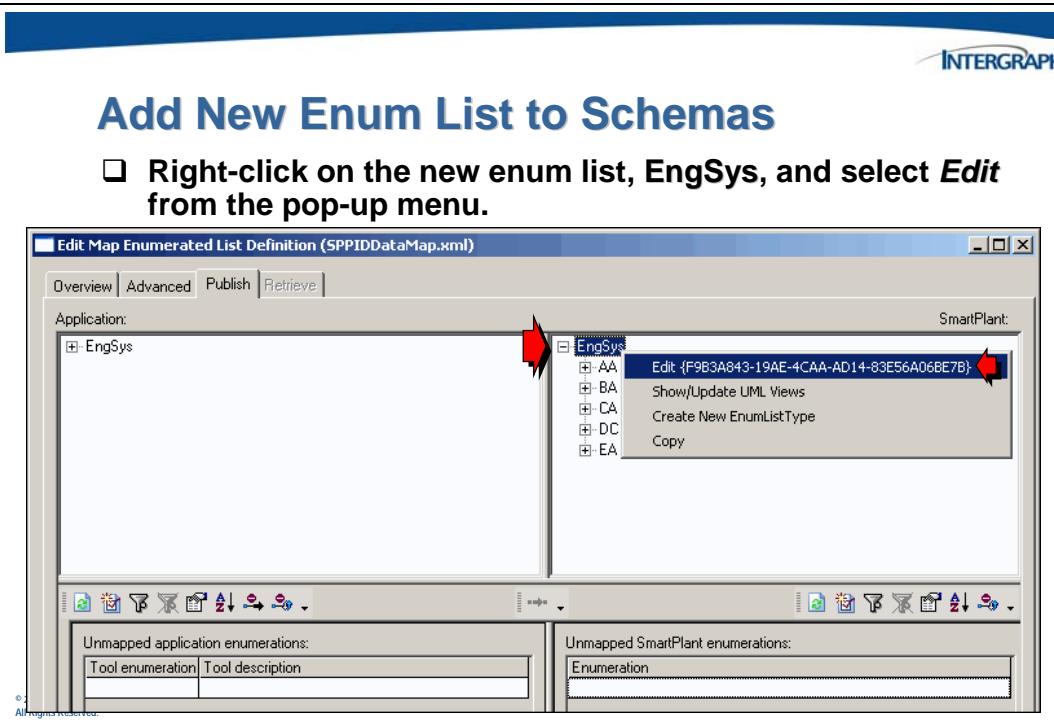
Because this enumerated list will be used later on in the integration process with SmartPlant 3D, some modifications must first be made.



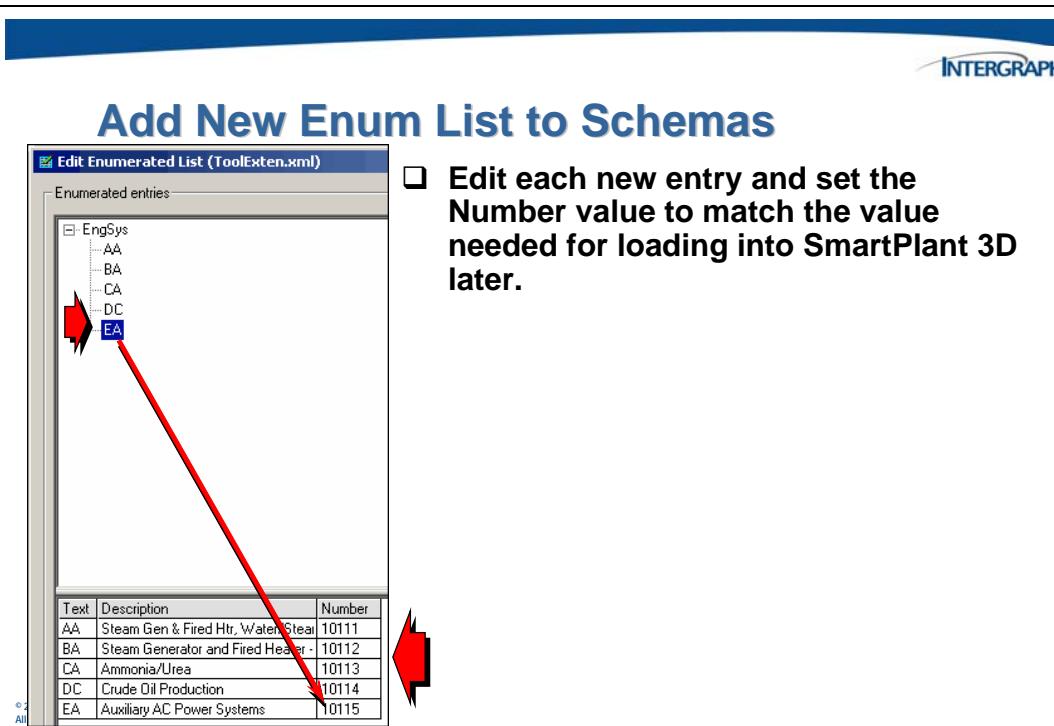
When the *Edit Map Enumerated List Definition* dialog displays, note the **Publish map** enum list has been filled in.



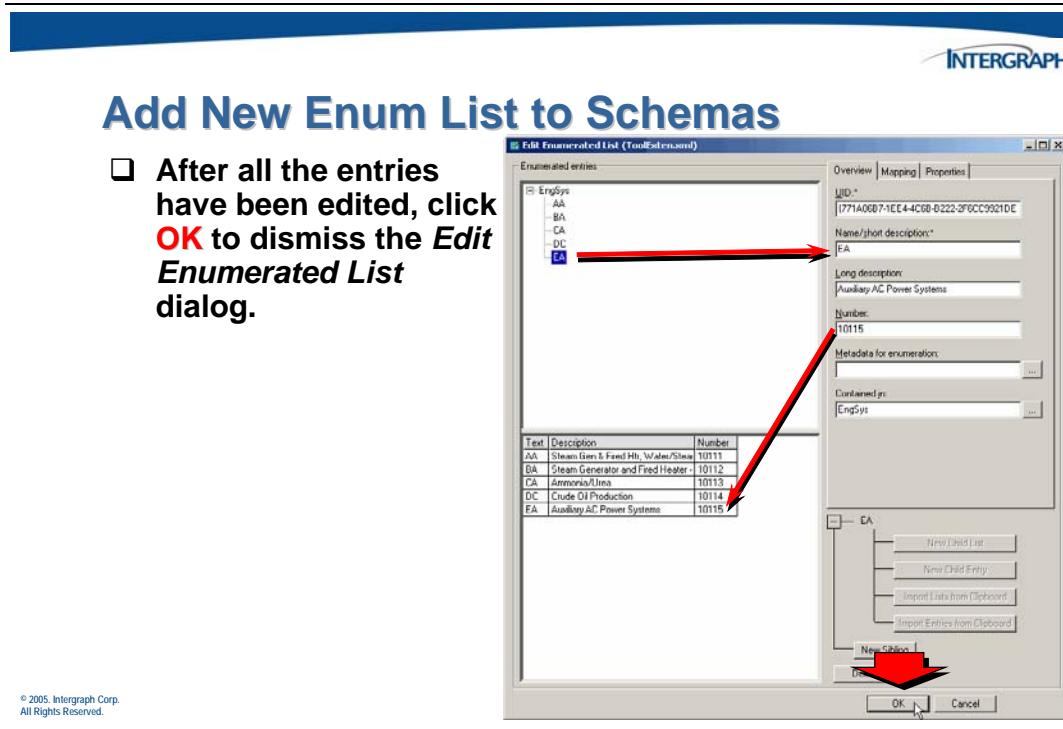
Edit the newly added enum list to modify the defaulted index **Numbers** to match what is needed for SmartPlant 3D.



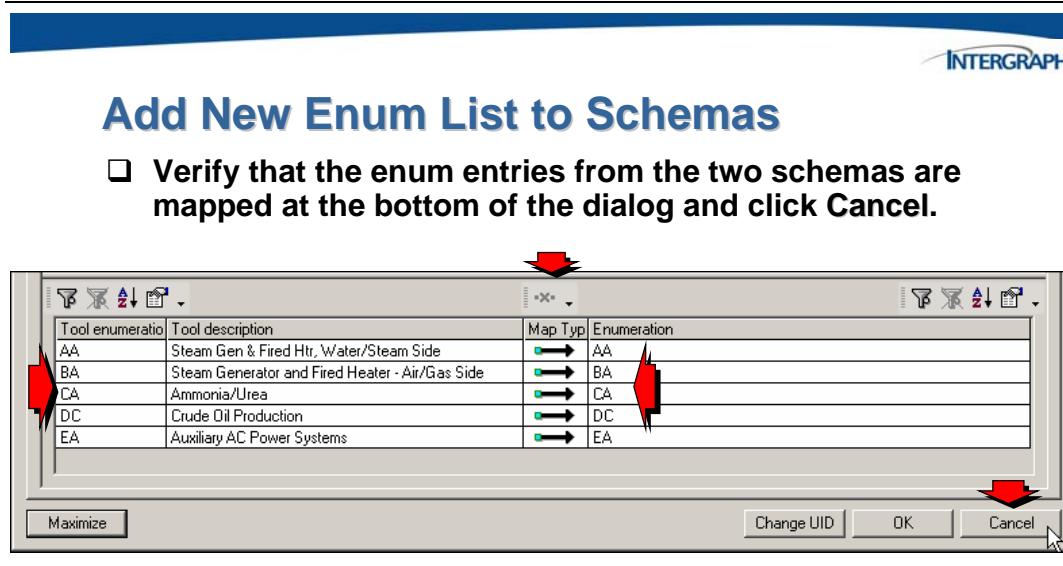
SmartPlant 3D requires that user defined enumerated values use numbers that are in the 10000 range.



Repeat the edit for each of the entries in this enumerated list.



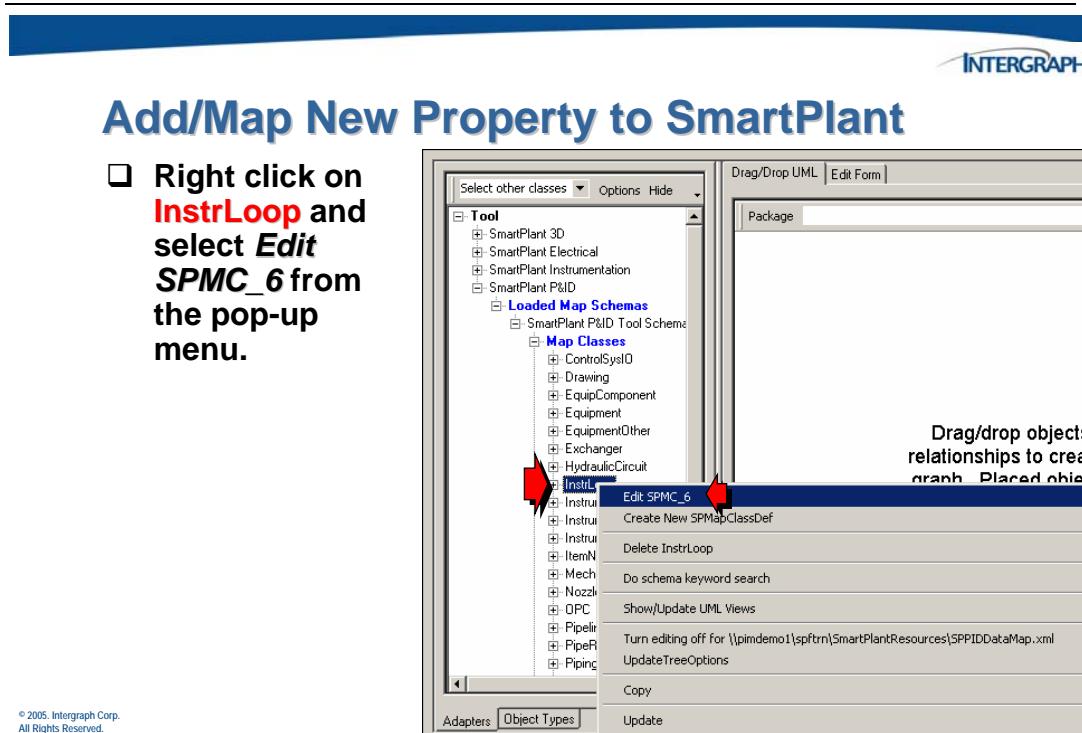
At the bottom of the *Edit Map Enumerated List Definition* dialog, the mapping that was performed automatically is displayed.



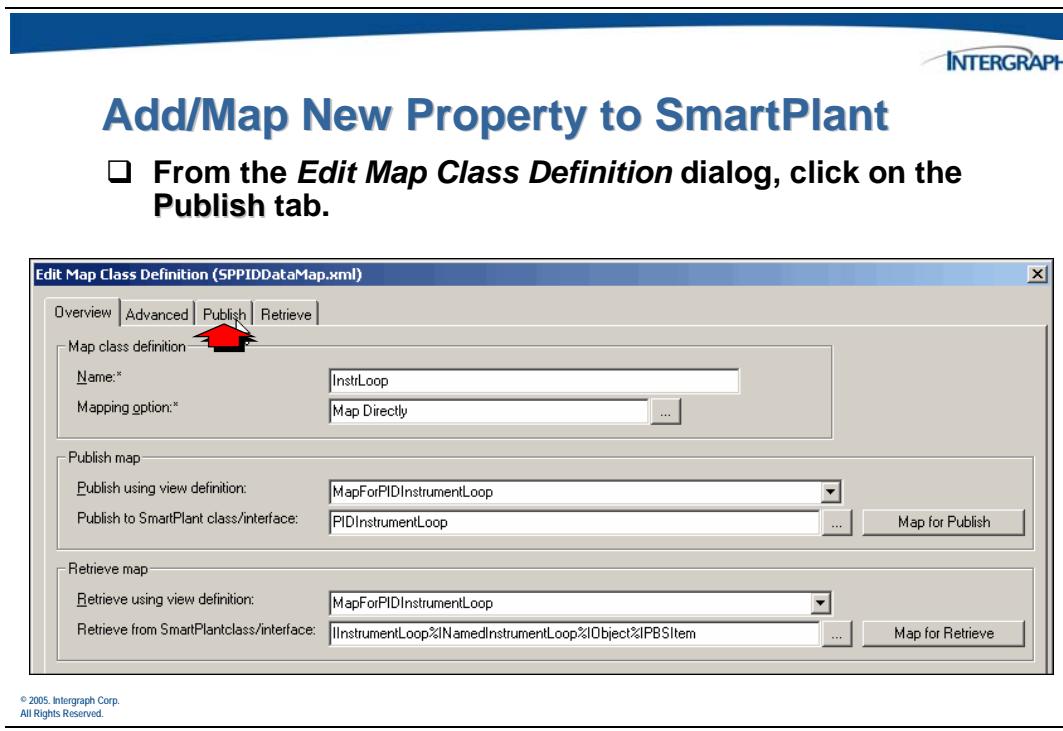
8.5.2 Adding a Complex Property to the Schemas

Use the Map Environment window to add the new complex property in the SmartPlant schema and perform the mapping to the SmartPlant P&ID tool map schema.

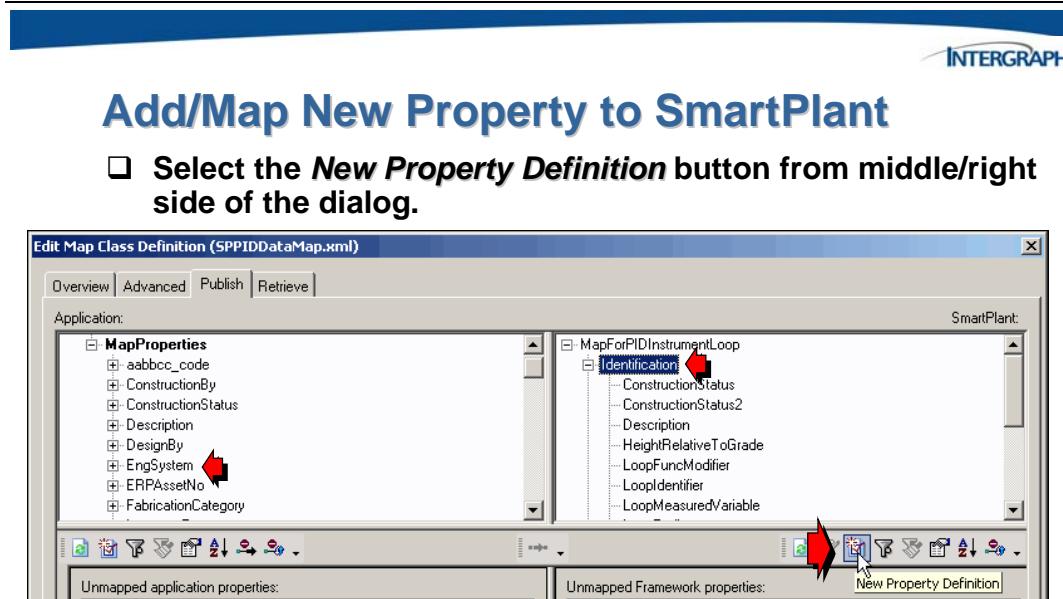
Locate the existing *InstrLoop* Map Class in the tree.



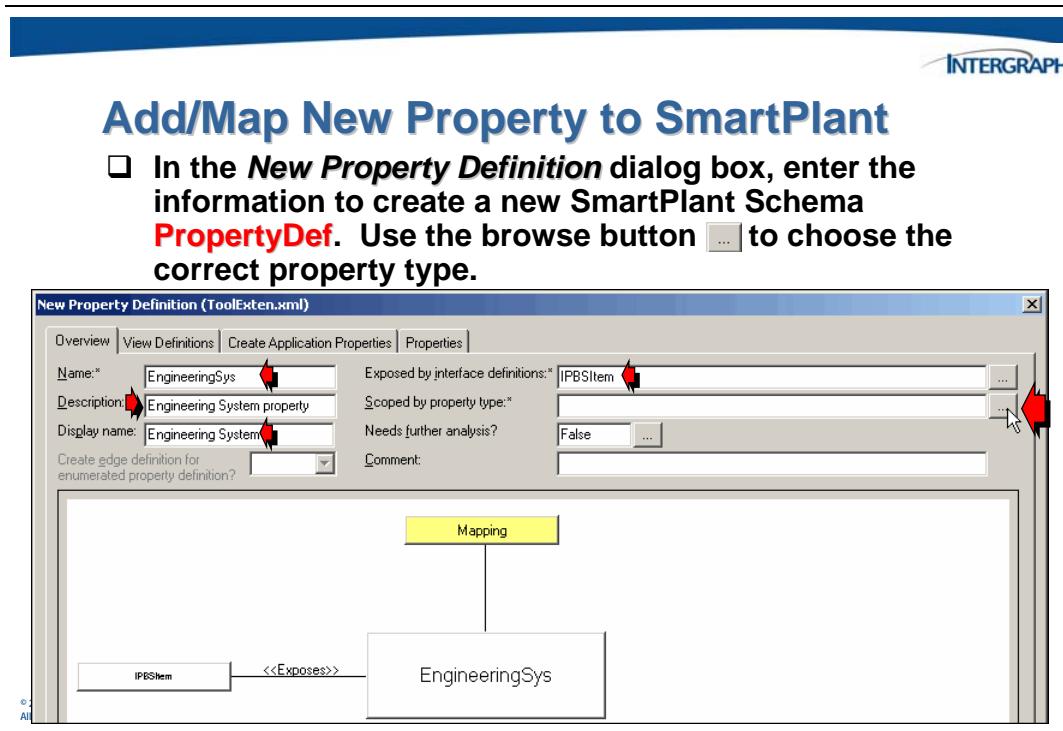
The *Edit Map Class Definition* dialog will be displayed.



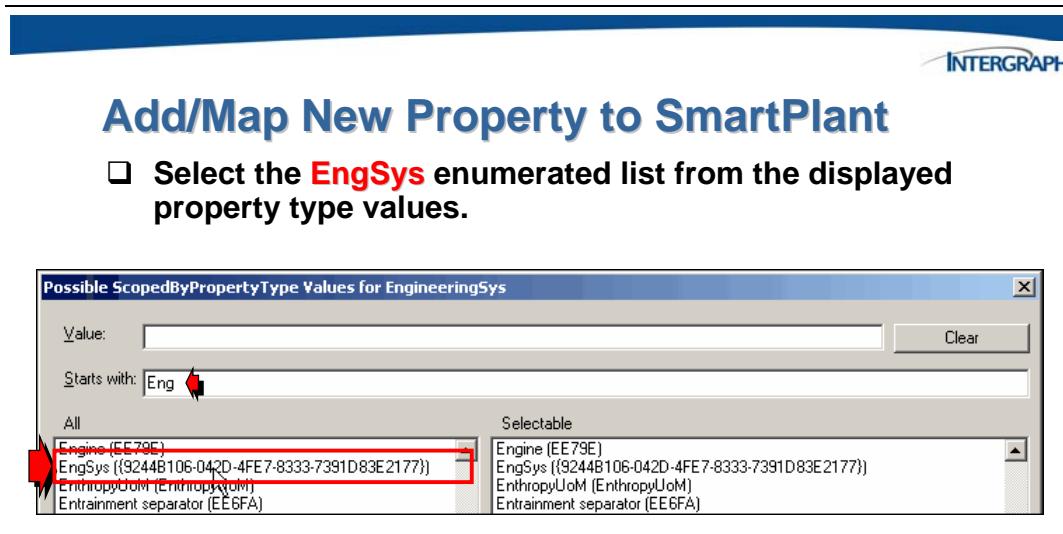
Expand the trees for both the tool map schema (left side) and the SmartPlant schema (right side). The new property will be associated with the *Identification* category in the dynamic view definition.



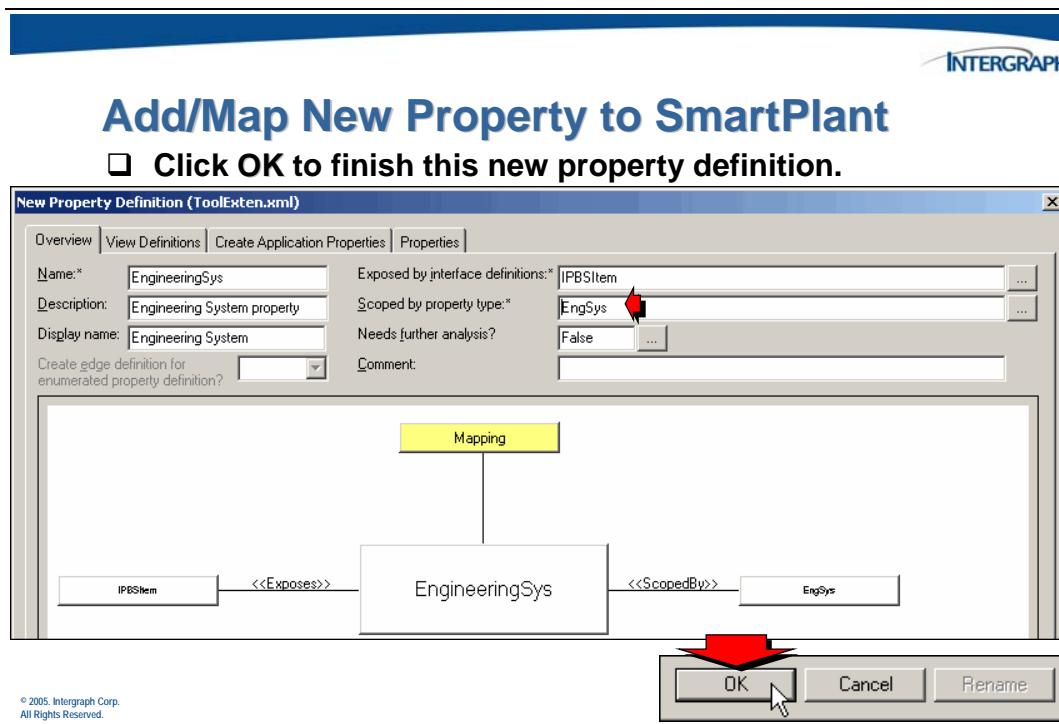
The *New Property Definition* dialog will display.



The new property will be exposed by the **IPBSItem** interface based on the earlier analysis performed. This property will be scoped by the new enum list that has already been created and mapped.

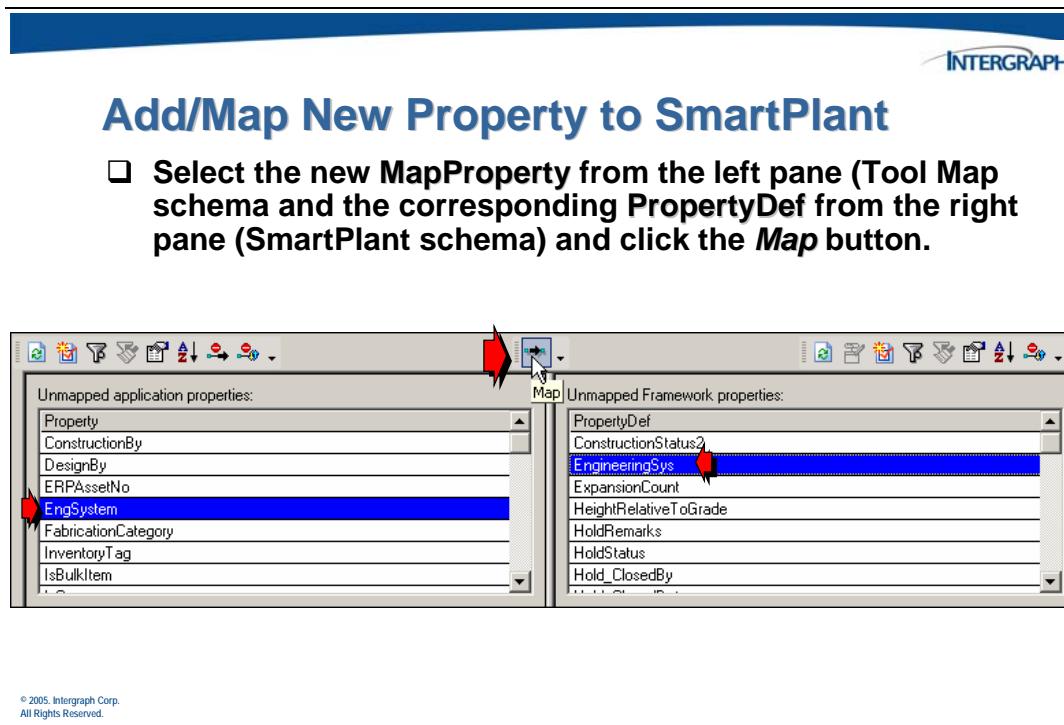


This will complete the creation of the new complex property.



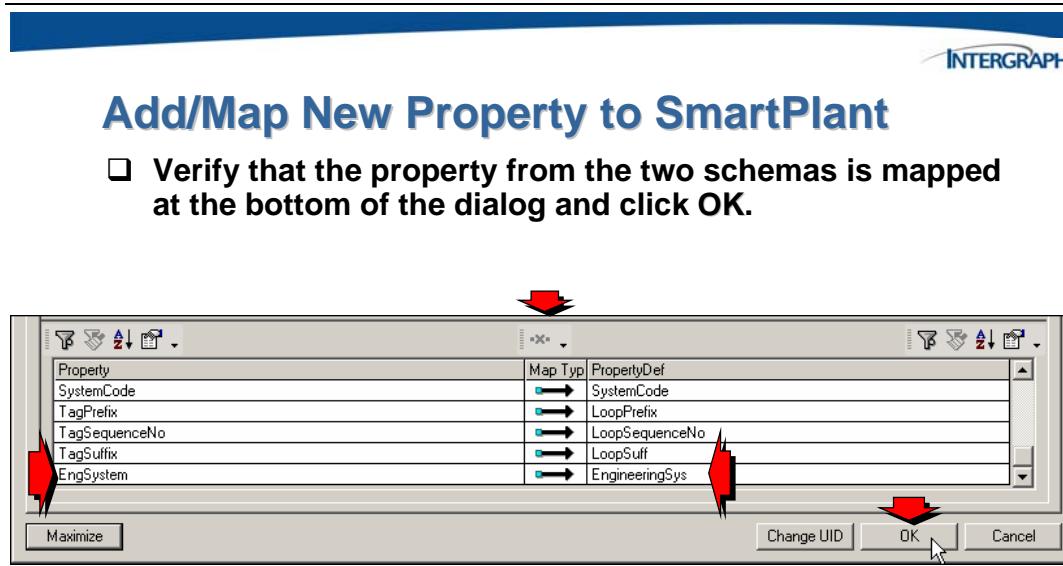
8.5.3 Mapping a Complex Property

In the middle control of the *Edit Map Class Definition* dialog the property mapping for the two schemas can be performed. Clicking a map property in the middle tree control selects it from the unmapped control.



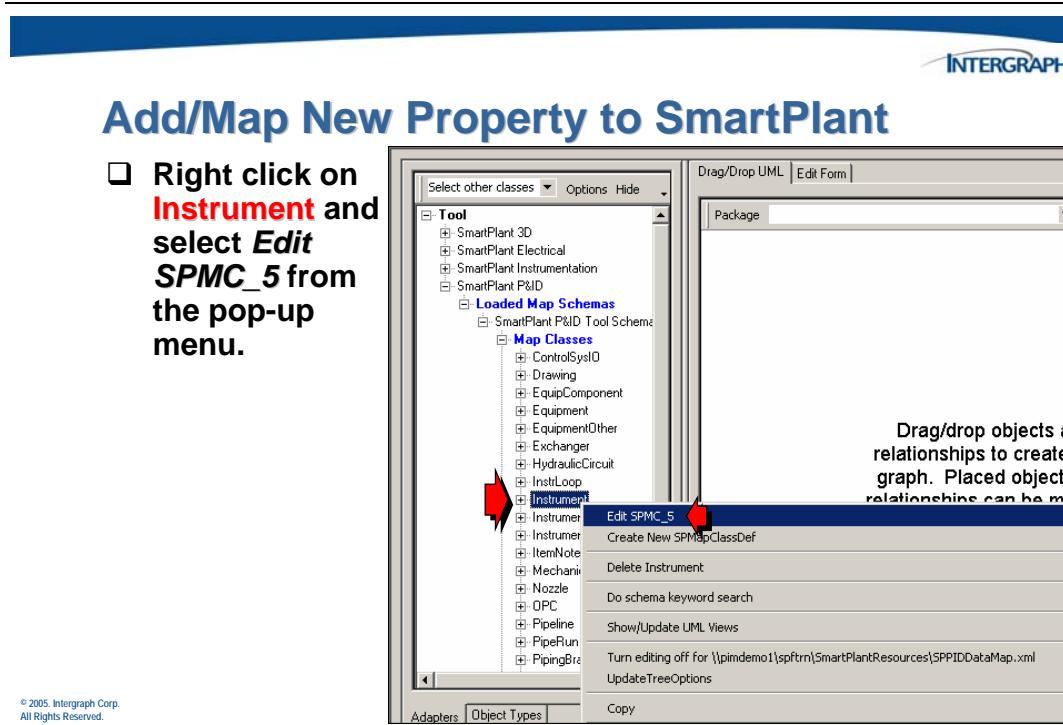
The **Map** button is used to map an unmapped application property to an unmapped SmartPlant property definition. Note that the two properties to be mapped **DO NOT** have to have the same name.

The results of the mapping will be shown in the bottom control.



© 2005, Intergraph Corp.
All Rights Reserved.

Again, since multiple P&ID classes inherit the **EngSystem** property, additional mapping should be performed.

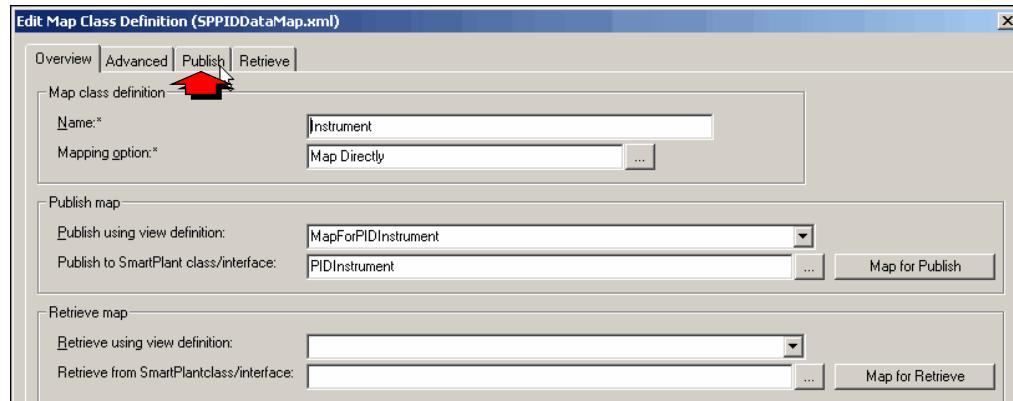


© 2005, Intergraph Corp.
All Rights Reserved.



Add/Map New Property to SmartPlant

- From the *Edit Map Class Definition* dialog, click on the Publish tab.



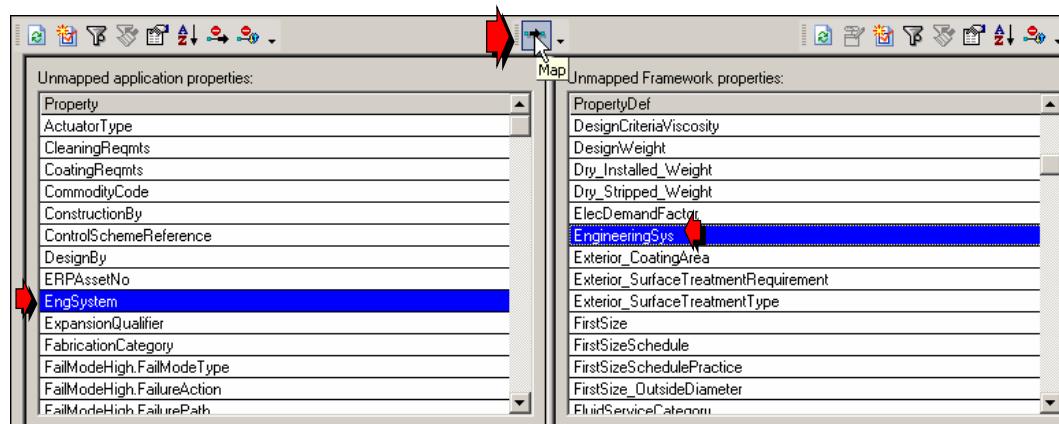
© 2005, Intergraph Corp.
All Rights Reserved.

In the middle control of the *Edit Map Class Definition* dialog, the unmapped EngSystem/EngineeringSys properties will be displayed.



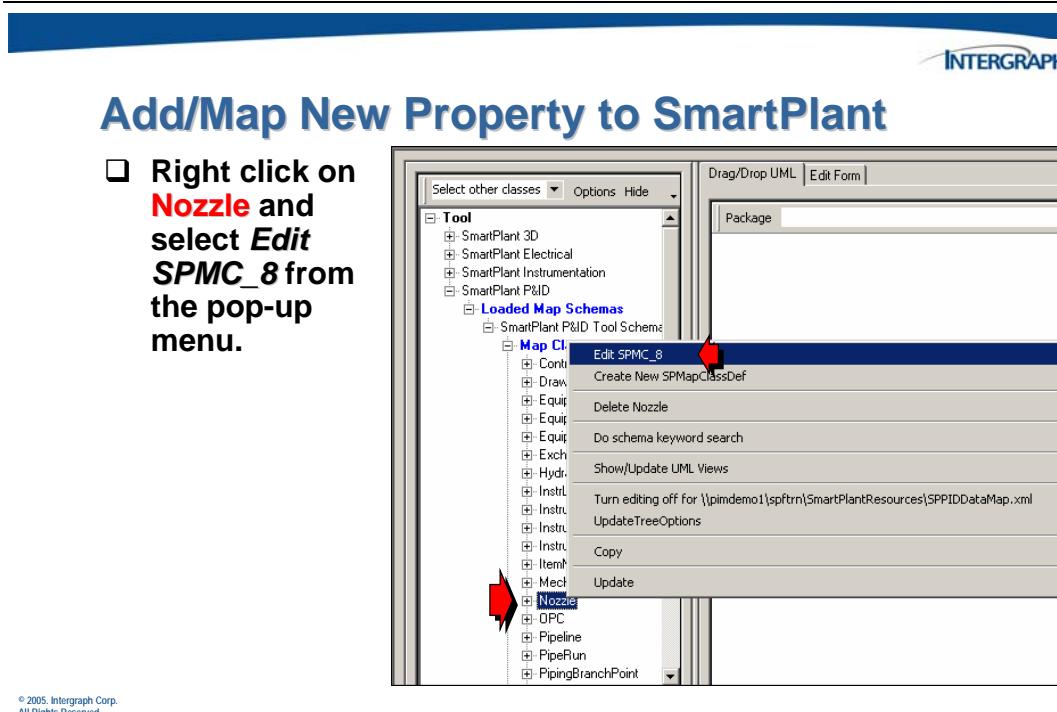
Add/Map New Property to SmartPlant

- Select the new MapProperty from the left pane and the new PropertyDef from the right pane and click the *Map* button.

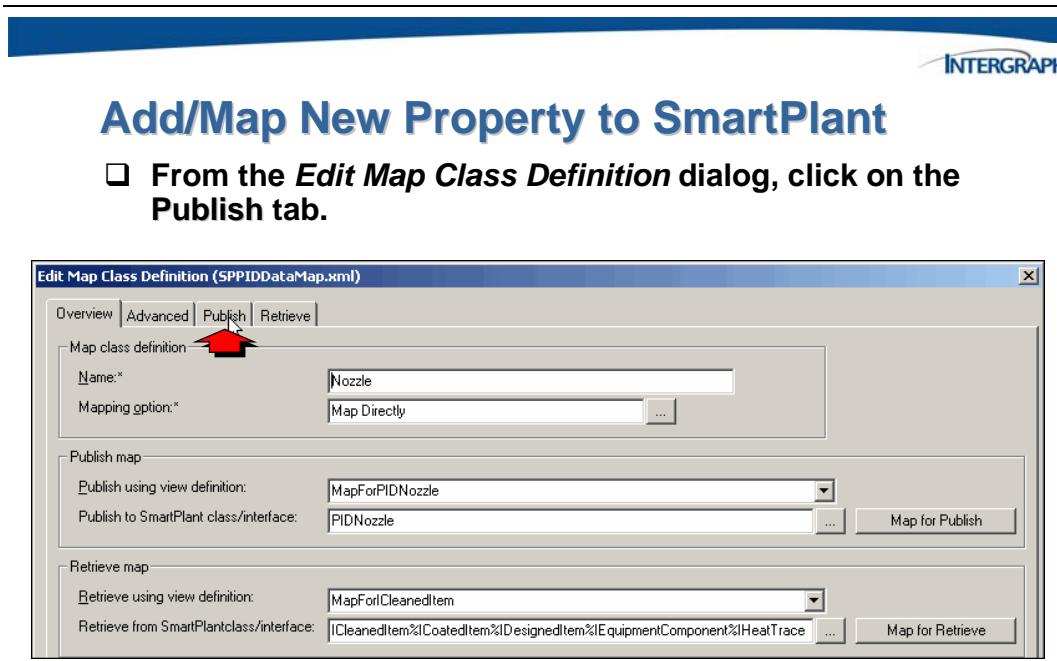


© 2005, Intergraph Corp.
All Rights Reserved.

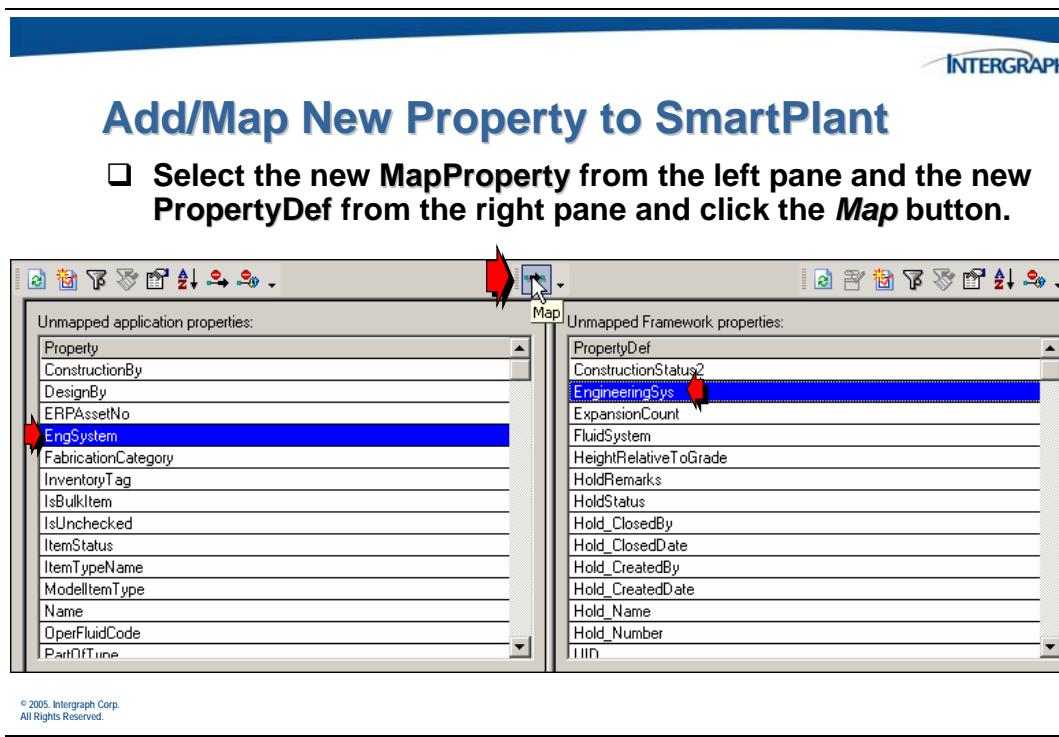
Again, since multiple **EngSystem/EngineeringSys** properties are used with different classes, continue to map the SmartPlant schema with the SPPID tool map schema.



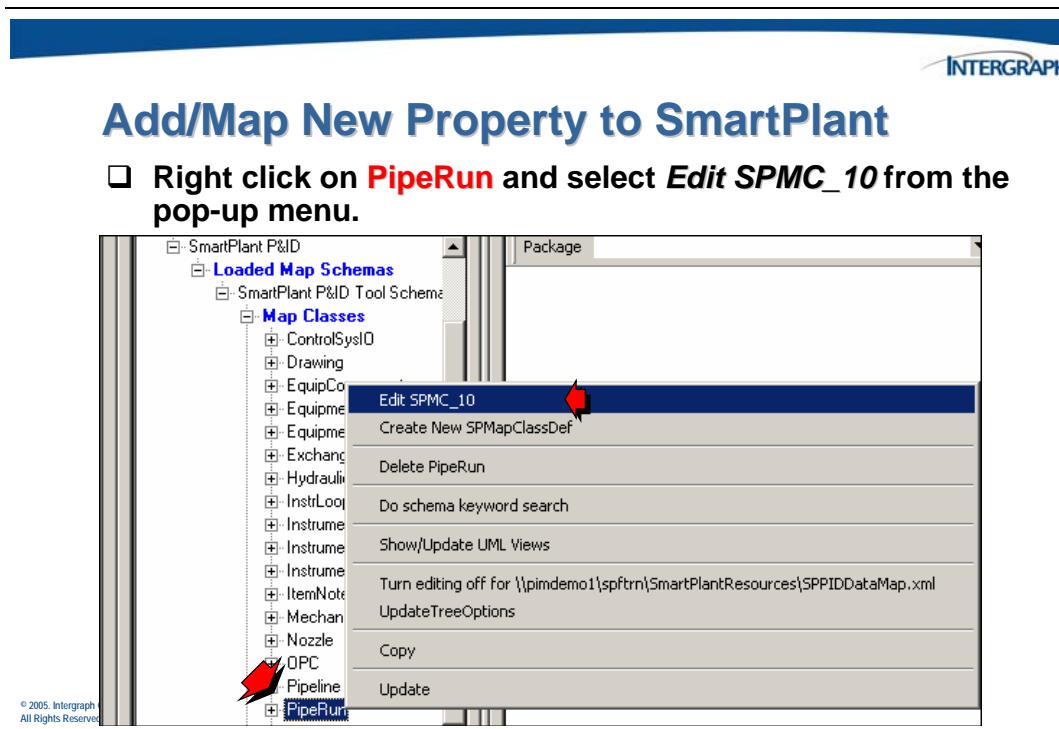
The *Edit Map Class Definition* dialog box will display.



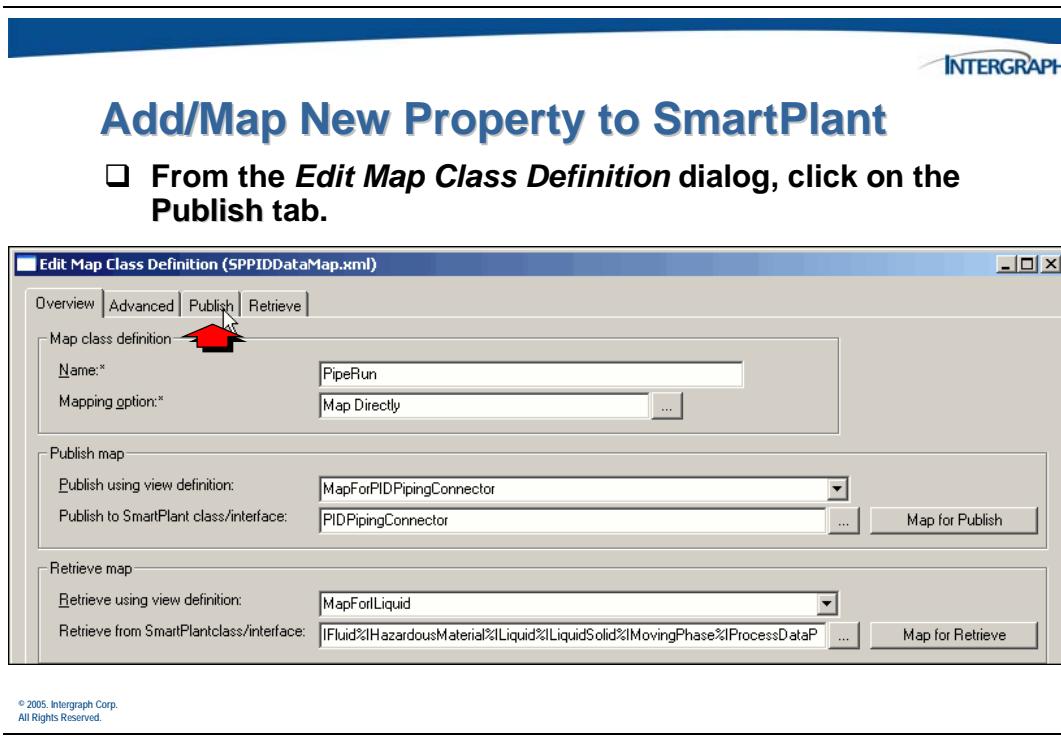
Perform the needed *Nozzle* to *PIDNozzle* mapping.



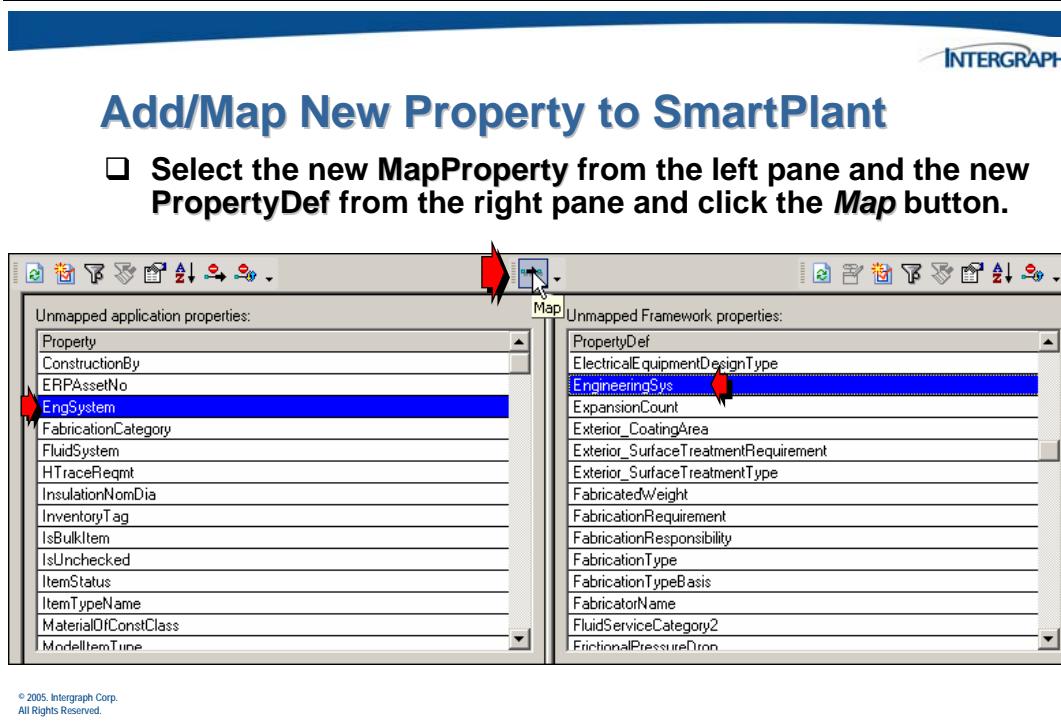
Repeat the mapping process for any desired SmartPlant P&ID classes.



Once more, the *Edit Map Class Definition* dialog box will display.



Perform the needed *PipeRun* to *PIDPipingConnector* mapping.

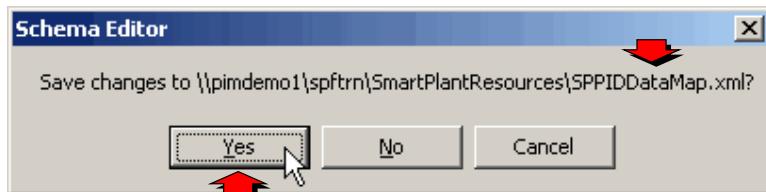


8.5.4 Saving Enum/Property Additions

Again, the additions to the SmartPlant schema and the mapping information will need to be saved to the respective xml files. You will be prompted to save the changes to the tool map schema.

Save Mapping Changes

- Verify the tool map schema name for the mapping changes and choose Yes.

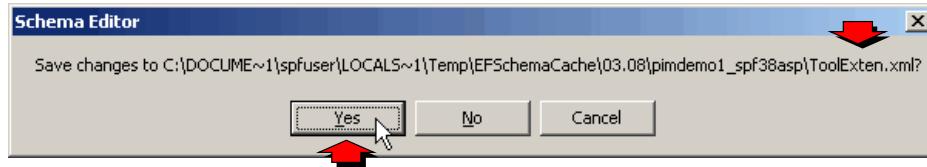


© 2005. Intergraph Corp.
All Rights Reserved.

You will also be prompted to save the changes to the SmartPlant extension schema.

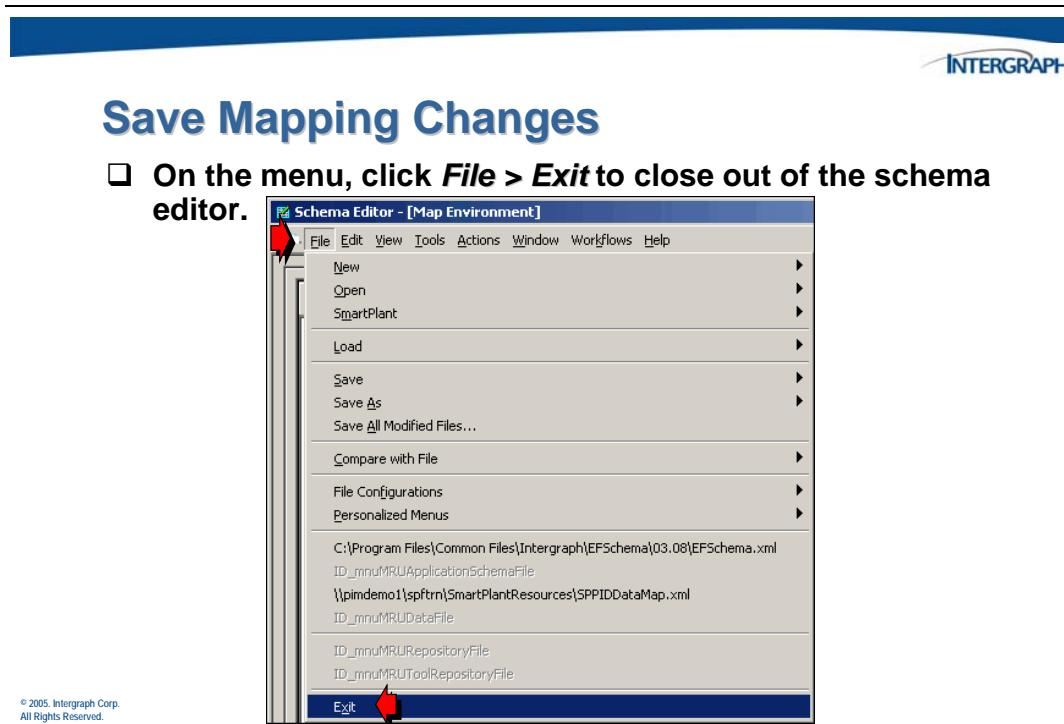
Save Mapping Changes

- Verify the extension schema name for the property addition and choose Yes.

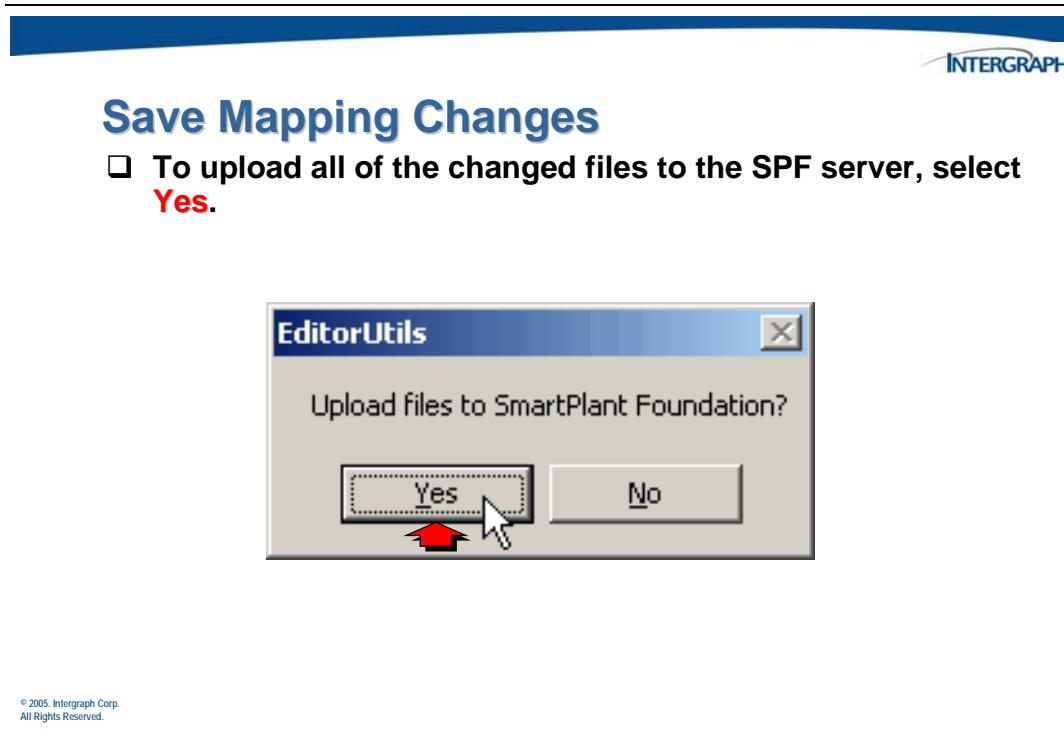


© 2005. Intergraph Corp.
All Rights Reserved.

Once all of the schema files have been saved exit the Schema Editor.



The final step in this scenario is to upload these files to the schema location on the SmartPlant Foundation server.



8.6 Activity 2 – Adding and Mapping a Complex Property

In this activity you will be creating custom properties in the SmartPlant P&ID authoring tool. You will also create a select list that will be used with a new property. Finally, you will perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in). Use Microsoft Excel to open the following file: D:\SPF_Training\Map_SpreadSheets\engsystems.xls before starting Data Dictionary Manager.
2. Click **Start > All Programs > Intergraph SmartPlant engineering Manager > Data Dictionary Manager** to start the *SmartPlant Data Dictionary Manager*.
3. Add a new select list (picklist).
 - Click **Select List** on the left side of the application window.
 - Scroll to the bottom of the Select List table and enter a new name in the empty row at the bottom of this table.
 - Name – **EngSys**
 - Dependent List – **None**
 - Click **Select Entry** on the left side of the application window.
 - In the **Select Entry** window, enter the following information in the blank row shown (you can copy and paste the Short Value from the excel spreadsheet):
 - Value – **AA**
 - Short Value – **Steam Gen & Fired Htr, Water/Steam Side**
 - Dependent Value – **<blank>**
 - Select the **Add Row** command from the tool bar to add the next entry to the new Select List:
 - Value – **BA**
 - Short Value – **Steam Generator and Fired Heater - Air/Gas Side**
 - Dependent Value – **<blank>**
 - Select the **Add Row** command from the tool bar to add the next entry to the new Select List:

- Value – **CA**
 - Short Value – **Ammonia/Urea**
 - Dependent Value – <blank>
- Select the **Add Row** command from the tool bar to add the next entry to the new Select List:
- Value – **DC**
 - Short Value – **Crude Oil Production**
 - Dependent Value – <blank>
- Select the **Add Row** command from the tool bar to add the last entry to the new Select List:
- Value – **EA**
 - Short Value – **Auxiliary AC Power Systems**
 - Dependent Value – <blank>
4. Add a new property to the Instrument database table that utilizes the new Select List just created.
- Click the *Database Tables* button then **Plant Item** in the Database Tables window.
- Select **Edit > Add Property** from the menu.
- In the *Add Property* dialog box, enter the following information:
- Name – **EngSystem**
 - Display Name – **Engineering System**
 - Data Type – **Select List**
 - Select List – **EngSys**
 - Format – **Variable length**
 - Default Value – **None**
 - Display to User – **Yes**
 - Use for Filtering – **Yes**
 - Category – **Process**
 - Depends on – **None**
- Click **OK** to close the *Add Property* dialog box.

5. Save the changes to the SmartPlant Data Dictionary Manager.
 - Click **File > Save**.
 - Click **File > Exit**.

Adding a Schema Property and Enumerated List

6. Click **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
7. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant...**
8. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with.
 - Click **Next**.
 - Select the tool map schema to be loaded, *SMARTPLANTPID/SmartPlant P&ID Tool Schema*.
 - Enable the **Load map schema** toggle.
 - Enable the **Connect to application schema** toggle.
 - Click **Finish**.
9. When the *Synchronize* dialog displays, confirm that the new property values have been imported by the meta data adapter and click **OK**.
10. Add new entries to a new Enumerated List in the SmartPlant schema to correspond to new entries that have been added to a tool map schema during the synchronization.
 - Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant P&ID Tool Schema entries are displayed.

- Expand the **Map Enumerated Lists** objects and verify that the entry for **EngSys** was added to the SmartPlant P&ID Tool Map Schema as a result of the synchronization with the SPPID application meta schema.

- Right-click on the EngSys entry in the tree and choose **Edit SP_12001** from the dynamic menu.

- What does *SP_12001* represent?

- Where does the number originate from?_____

- Select the **Advanced** tab in the *Edit Map Enumerated List Definition* dialog. Verify that the new list is displayed and has all of the correct entries in it.

- At the bottom of the dialog, locate and select the **New SmartPlant Enumerated List with Correlated Entries** button.

- What does this button do?

- Click the **OK** button to close the *Edit Map Enumerated List Definition* dialog.

- Right-click on the EngSys entry in the tree and choose **Edit SP_12001** from the dynamic menu once more to change the child entry numbers in the EngSys list.

- Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog. Then right-click on the EngSys enum list in the SmartPlant control and choose **Edit** from the dynamic menu.

- In the *Edit Enumerated List* dialog box, change the **Number** information:

Text	Description	Number
AA	Steam Gen & Fired Htr, Water/Steam Side	10111
BA	Steam Generator and Fired Heater - Air/Gas Side	10112
CA	Ammonia/Urea	10113
DC	Crude Oil Production	10114
EA	Auxiliary AC Power Systems	10115

- Click the **OK** button to close the *Edit Enumerated List* dialog and the *Edit Map Enumerated List Definition* dialog.

11. Add a new property to the SmartPlant schema to correspond to the new property that has been added to a tool meta schema and tool map schema.

- Expand the **Map Classes** and right-click on the *InstrLoop* entry in the tree and choose **Edit SPMC_6** from the dynamic menu.
- Select the **Publish** tab in the *Edit Map Map Class Definition* dialog. Expand the tree for both the application/tool map schema and the SmartPlant schema in the upper control.
- Highlight **Identification** in the tree and select the **New Property Definition** button **below** the SmartPlant tree in the upper control.
- In the *New Property Definition* dialog box, enter the following information:
 - Name – **EngineeringSys**
 - Description – **Engineering System property**
 - Display Name – **Engineering System**
 - Click  next to the *Exposed by interface definitions* box, UNSELECT the *IInstrumentLoop primary interface* and then select the **IPBSItem InterfaceDef** (if that was on your list from step 10 in activity 1) from the *Possible ExposedByInterfaceDefs for EngineeringSysEngineering* dialog box. The *Exposed by interface definitions* field should contain only ONE entry.
 - Click  next to the *Scoped by property type* box and select the **EngSys** enumerated list from the *Possible ScopedByPropertyType Values for EngineeringSys* dialog box.
- Click **OK** to close the *New Property Definition* dialog box.

Defining Schema Mapping

12. Highlight the *EngineeringSys MapProperty* in the **Unmapped applications properties** control and the *EngineeringSys MapProperty* in the **Unmapped SmartPlant properties** control in order to perform the schema mapping.
- Make sure that **EngSystem** is highlighted in the middle control in the application section and that **EngineeringSys** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPPID tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *EngSystem* maps to *EngineeringSys* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Class Definition* dialog box.

13. Repeat the mapping for the following classes:

- EquipComponent**
- Instrument**
- Nozzle**
- Pipeline**
- PipeRun**
- PipingComp**

14. Save the changes to the all schema files.

- From the menu, click **File > Save All Modified Files...**
- Answer **Yes** to all of the individual xml file save prompts.
- Click **File > Exit** to close and exit the Schema Editor.
- When prompted to Upload files to SmartPlant Foundation, select **Yes**.

15. Use Windows Explorer to verify that all the files were uploaded to **C:\Program Files\Common\Intergraph\EFSchema\03.08** with the latest time and date.

16. When you are finished with this activity, you may take a short break until the other students have finished.

9

CHAPTER

Loading and Testing the Schema Changes

9. The Schema Loader Interface

The SmartPlant Schema Loader is used by system administrators to load/update the SmartPlant schema into SPF when the SmartPlant schema changes. Importing the schema merges the SmartPlant schema data with data that already exists in the system administration database and extends the SmartPlant schema by adding unique keys, special interfaces only used by SPF, and additional attributes on relationship definitions.

9.1 Schema Load Process

The SmartPlant Schema Loader is installed with the SPF server component. By default, the installation location is `\SmartPlant\Foundation\2007`.

Note: If you load the SmartPlant database dump files during installation, you are not required to load the schema separately using this utility.



Schema Load Process

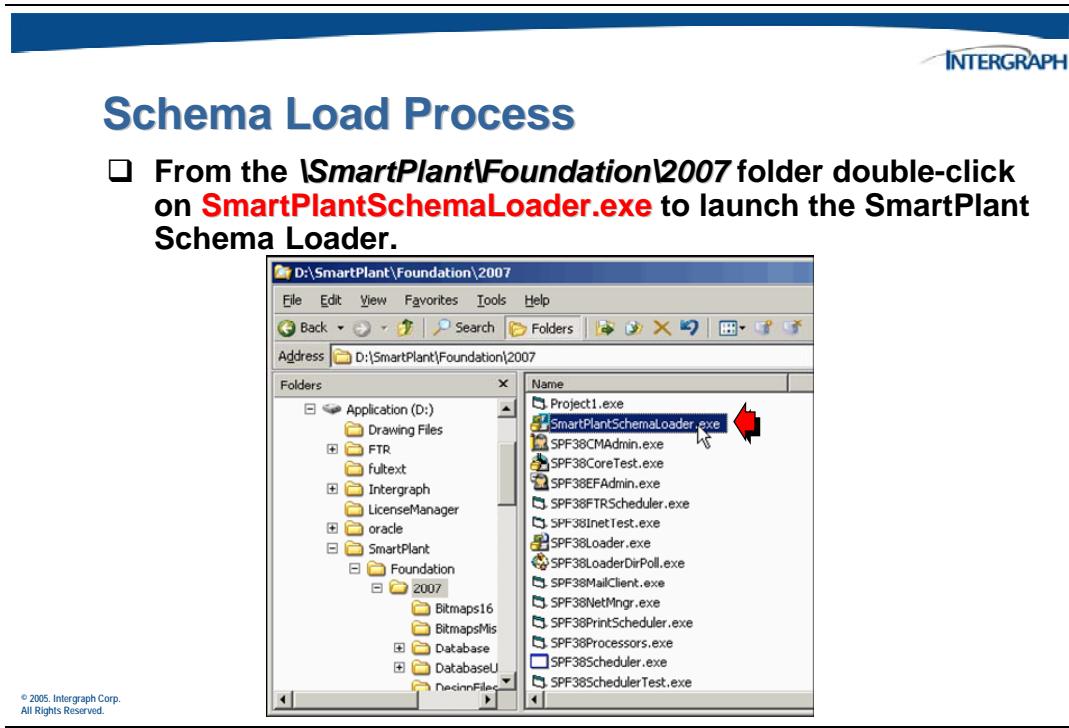
The SmartPlant Schema Loader is used by system administrators to perform the following tasks:

- Import the SmartPlant schema/extension schema(s) into the SmartPlant® Foundation system administration database when the SmartPlant schema changes**
- Export the data from the SmartPlant® Foundation system administration database to an .XML file**
- Delete everything specific to the SmartPlant schema from the system administration database tables in preparation for reloading the schema**

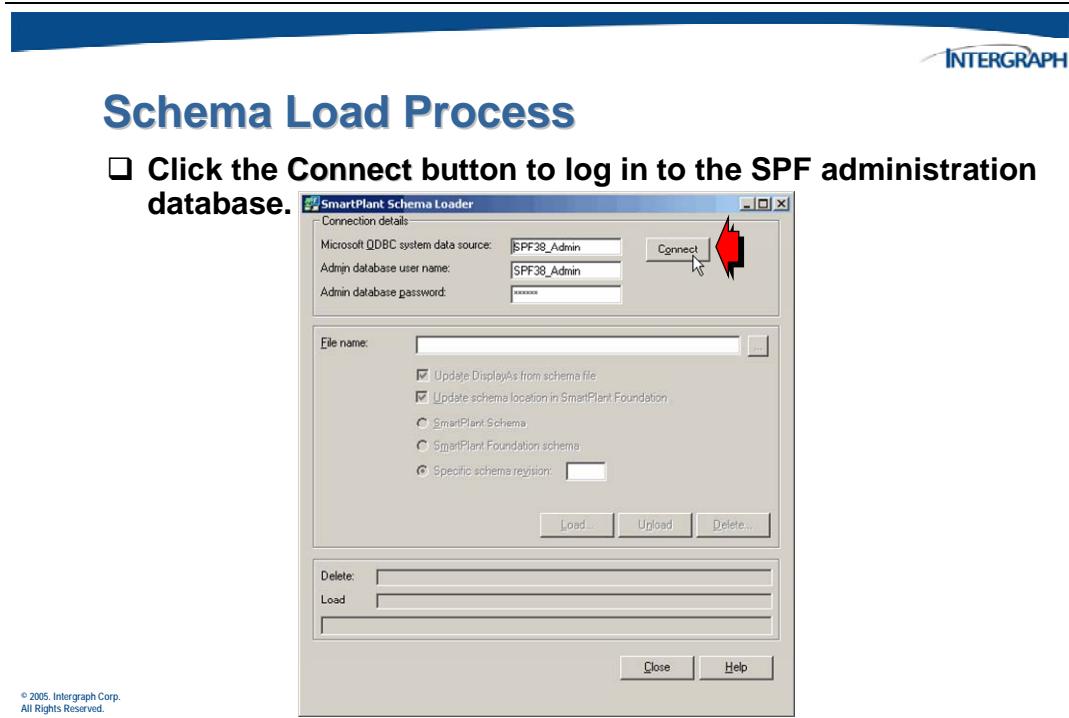
Deleting the SmartPlant schema does not delete users, methods, menus, and so on or any objects that do not appear in the SmartPlant schema.

All SmartPlant administration objects that are created in SPF have a property called *Schema Revision*, which is stamped with a value of either **EF**, **SPF**, a project/module specific value such as 'LL' (for Line List), or (if created interactively) **EFS**.

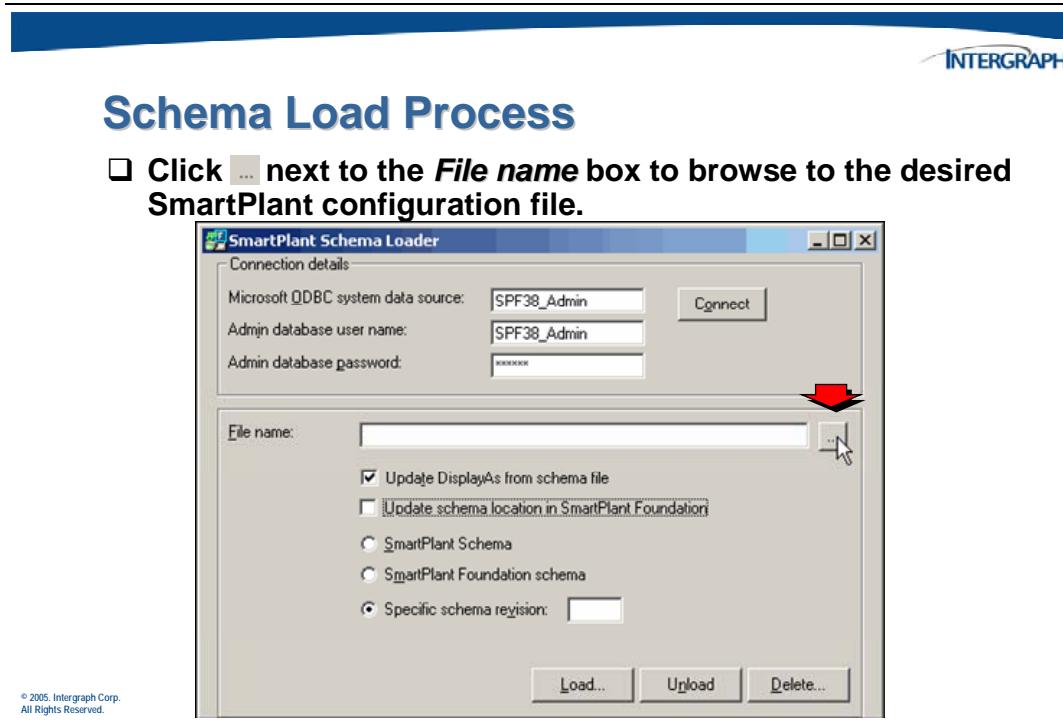
To launch the SmartPlant Schema Loader you can double-click **SmartPlantSchemaLoader.exe** or you can open a *Command Prompt* window and **cd** to the location of the executable.



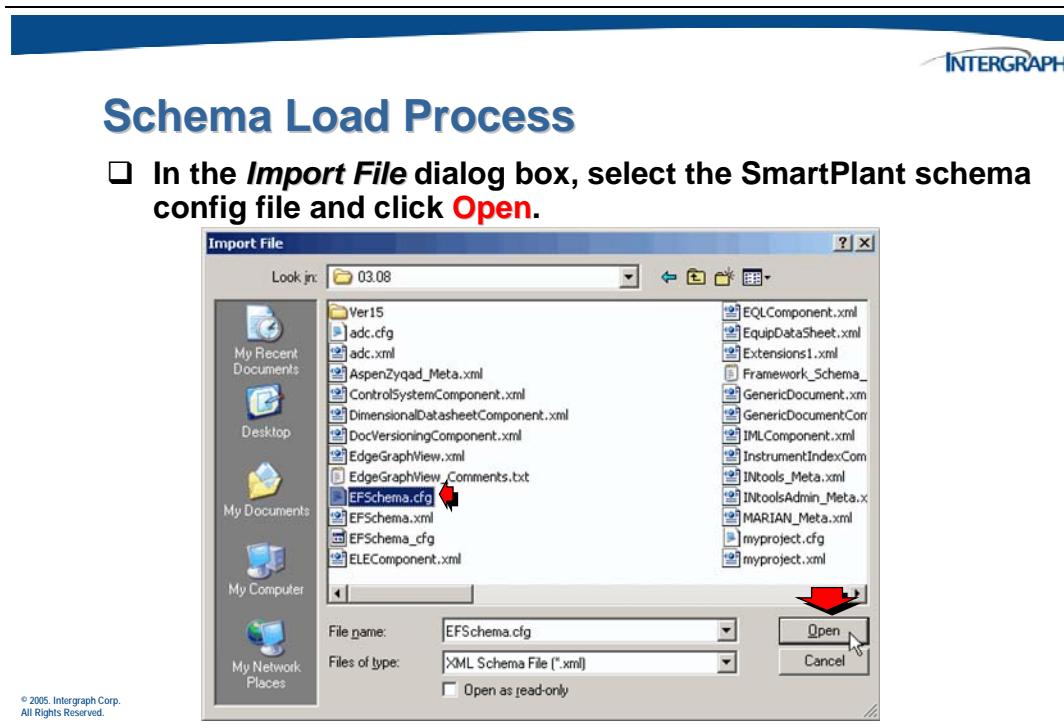
The *SmartPlant Schema Loader* window appears.



The software automatically uses the data source name (DSN), database user ID, and database password that you entered for the system administration database during SmartPlant Foundation installation/configuration.



The default installation location for the SmartPlant schema configuration is *C:\Program Files\Common Files\Intergraph\EFSchema*, and the file name is **EFSchema.cfg**.

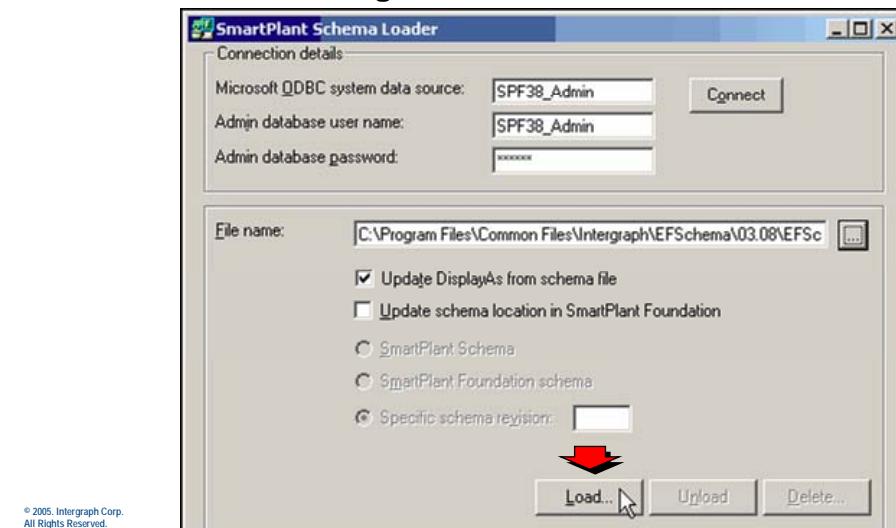


Note: Never load a component schema as the SmartPlant schema. Doing so will cause everything else to be deleted.



Schema Load Process

- ❑ Once the config file has been selected, click **Load**.

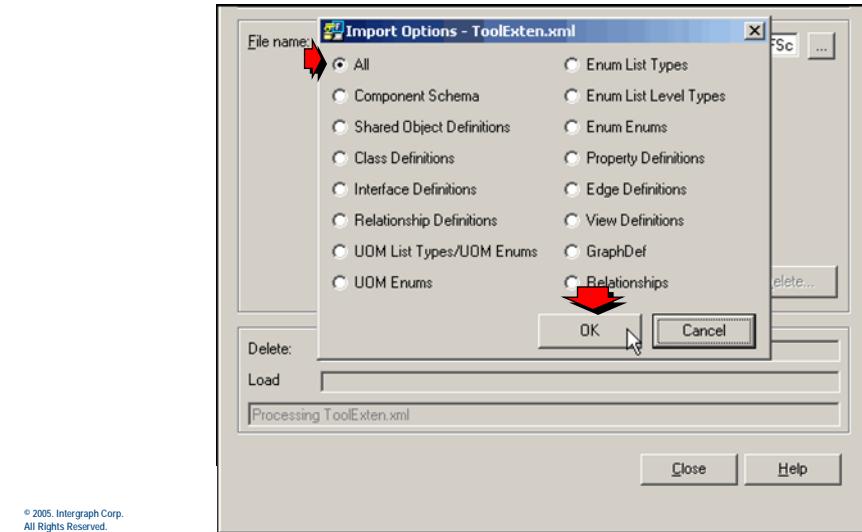


In the **Import Options** dialog box, select the type of data that you want to load from the schema into the system administration database.

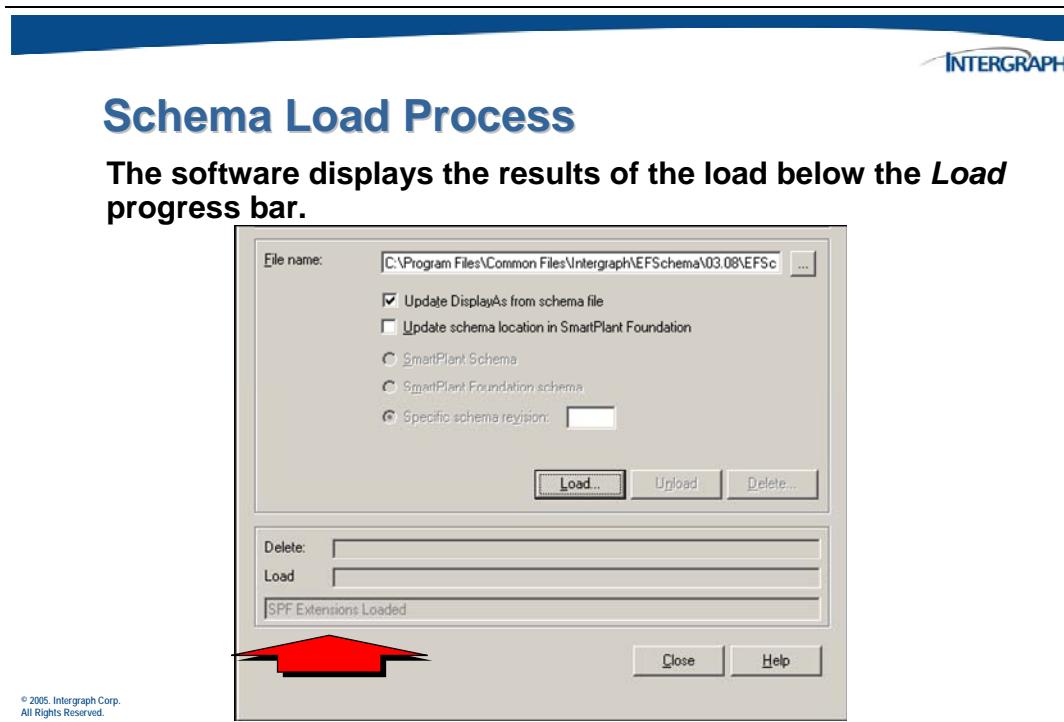


Schema Load Process

- ❑ Click **OK** to load the schema.



By default, the software loads the entire SmartPlant schema. However, you can limit the data that you load by selecting another option in the **Import Options** dialog box.



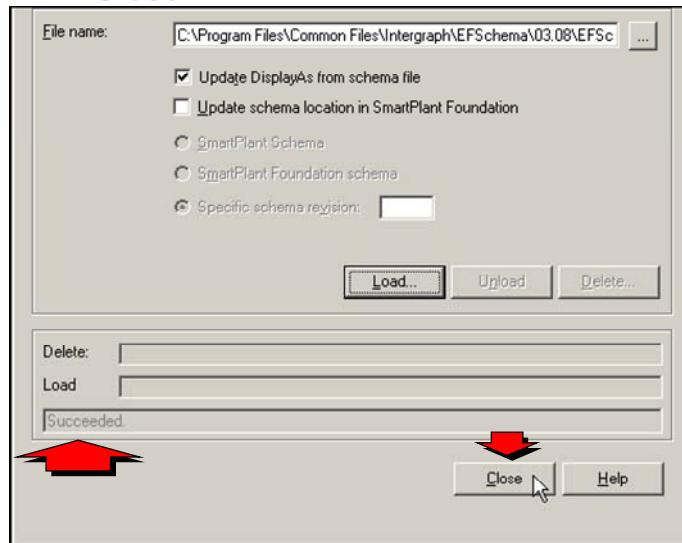
The following defines the options that can be set for the SmartPlant Schema Loader:

- SmartPlant Schema** – Updates the SPF SysAdmin database with any changes from the *EFSchema.xml* file. It will set an attribute value (flag), **EF** that will be used for a comparison.
- SPF** – Uses a file called *spfextensions.xml* to load SPF extensions into the SPF SysAdmin database. It will set an attribute value (flag) to **SPF**.
- Specific schema revision** – Is used for a project team to load custom extensions into the SPF SysAdmin database. It uses a custom xml file, <custom_component.xml> to update the SmartPlant Schema. It will set an attribute value (flag) to **ToolEx** for example.



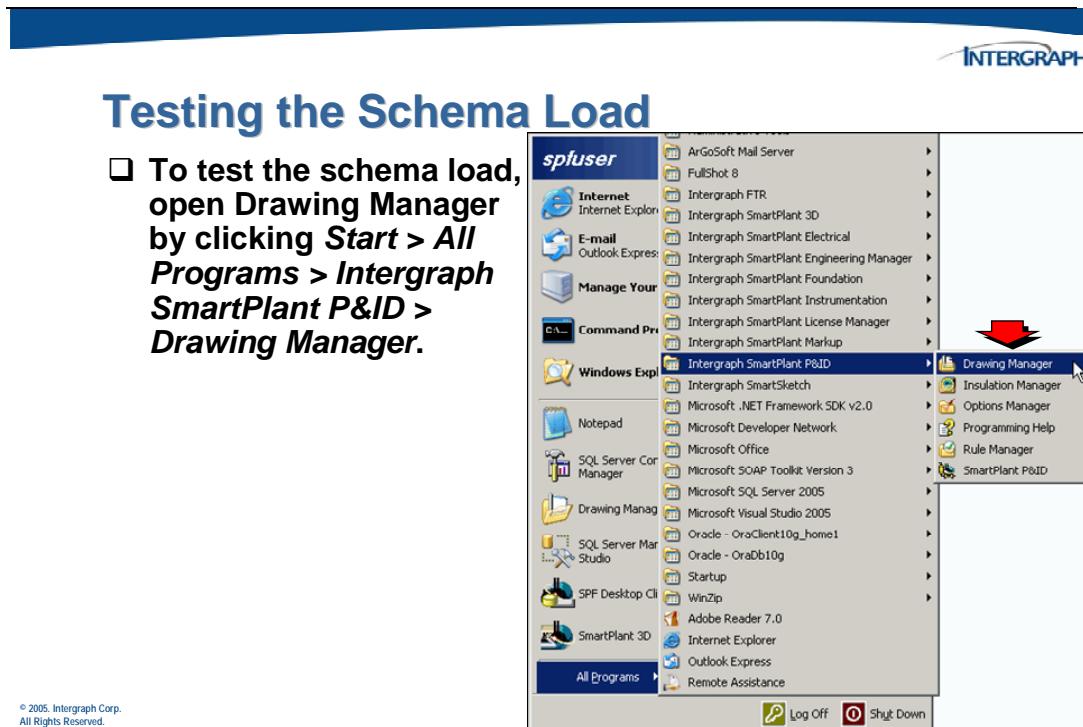
Schema Load Process

- Click the **Close** button once the schema has been loaded.

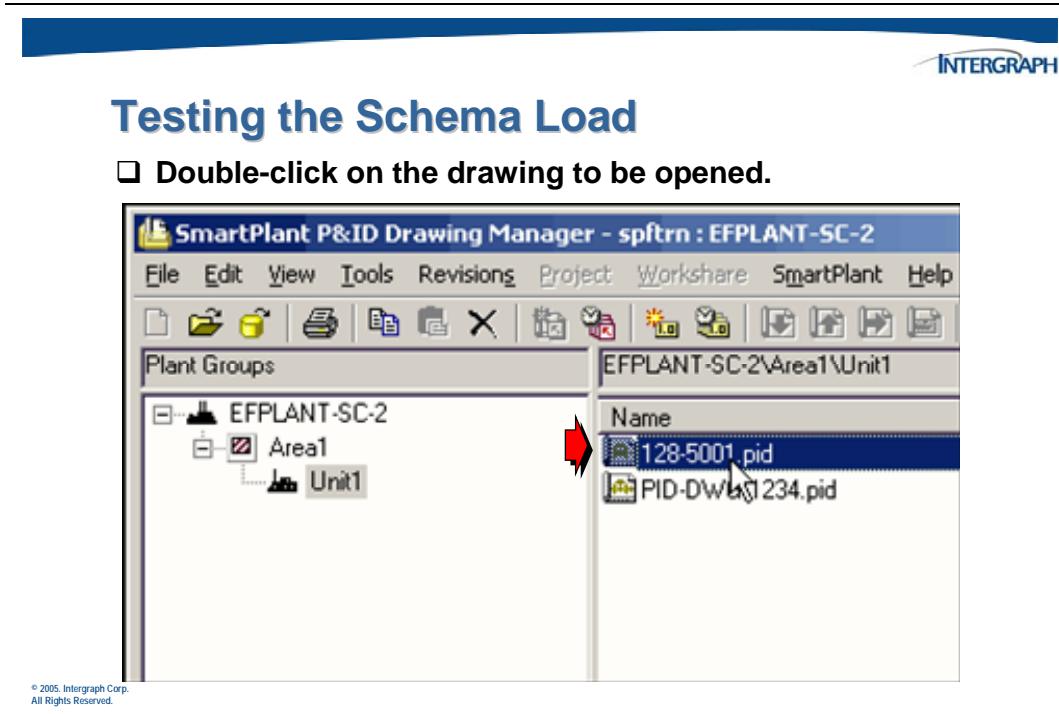


9.2 Testing the Schema Load

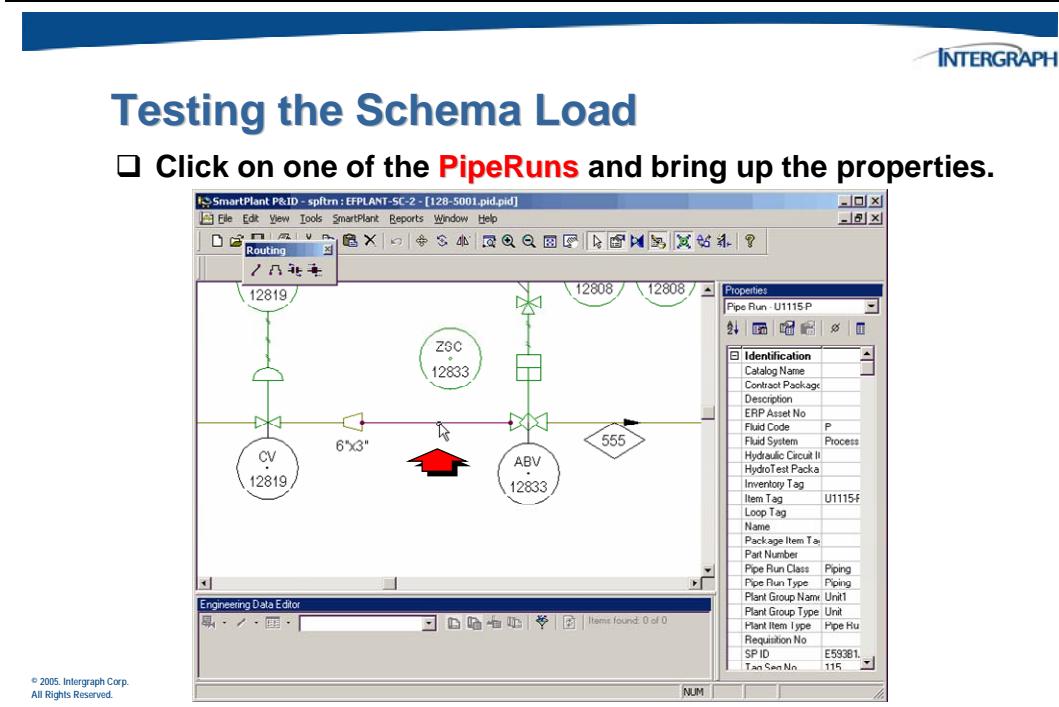
Each registered authoring tool can publish documents and associated data to the SmartPlant Foundation database. During a publish, an object is created in SmartPlant Foundation with a view file and associated data. In this chapter, the modified schema model has been loaded in the SmartPlant Foundation administration database. These modifications can be tested by publishing an update from SmartPlant P&ID.



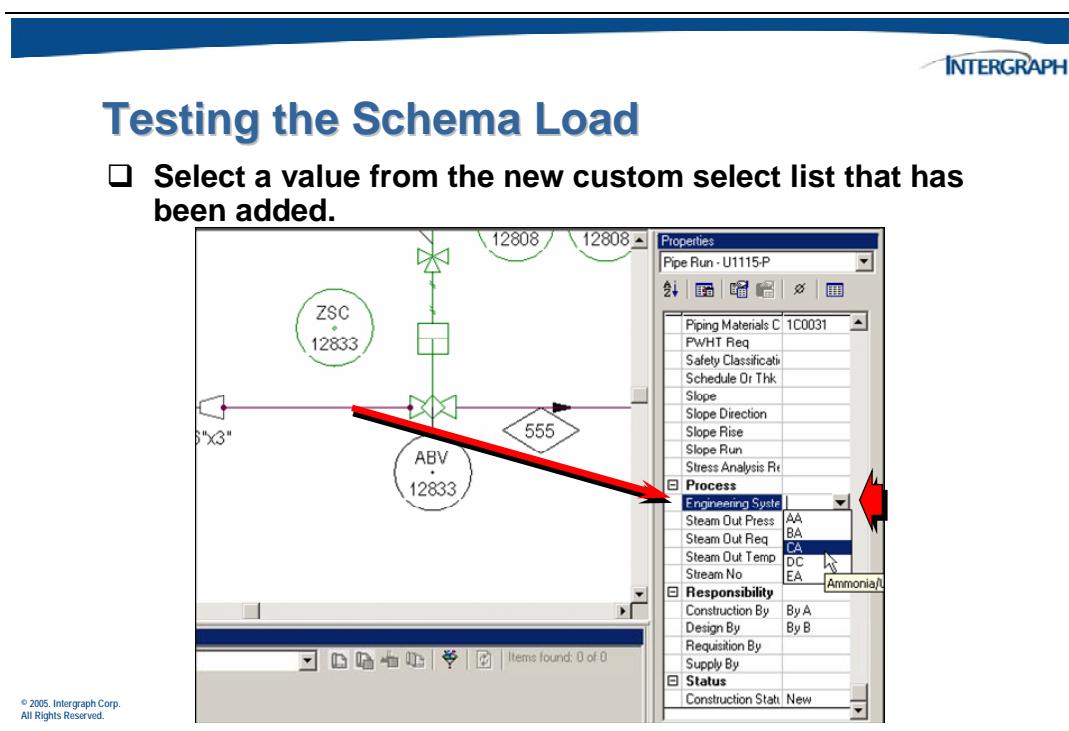
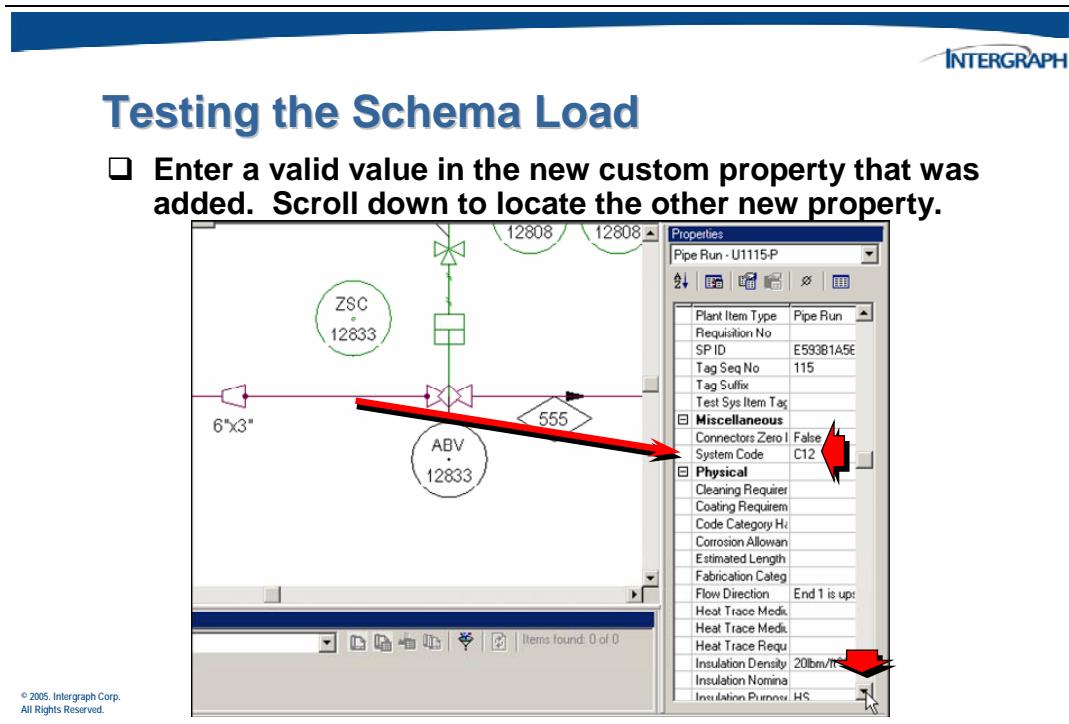
The *SmartPlant P&ID Drawing Manager* window will display. The registered plant to be used for the publish operation is **EFPLANT-SC-2**.



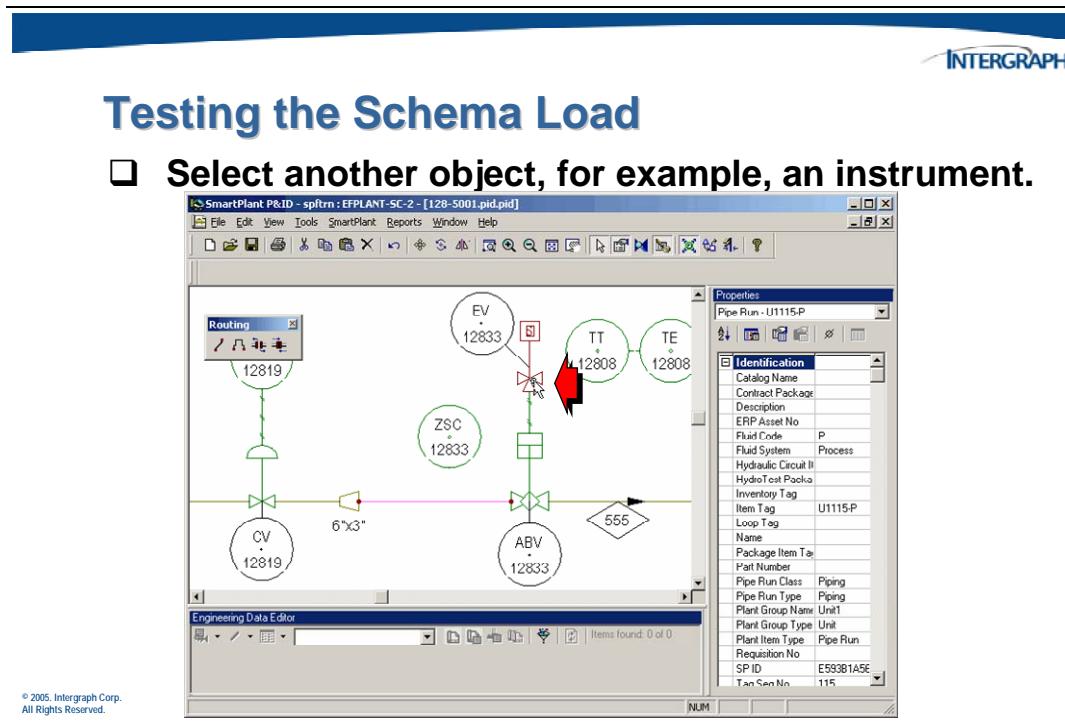
Locate some of the PipeRuns, Instruments and Nozzles that have been placed in this drawing.



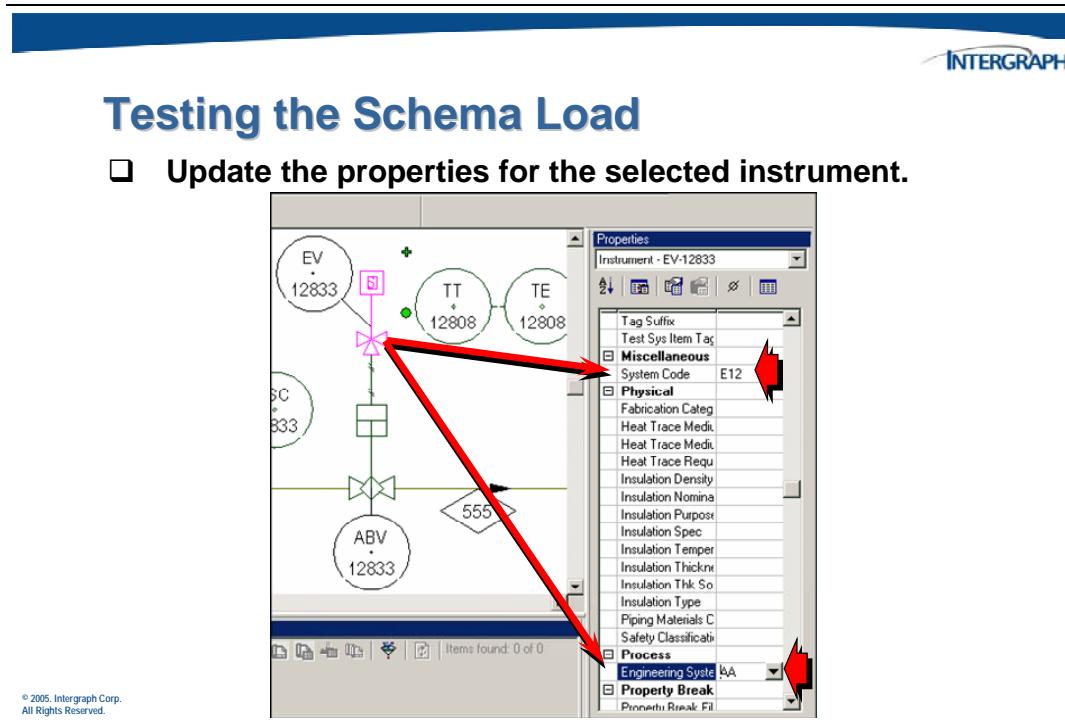
This PipeRun has already been published previously. But two new properties have been added to the *PipeRun* class. Update this existing PipeRun by adding values for these newly added properties.



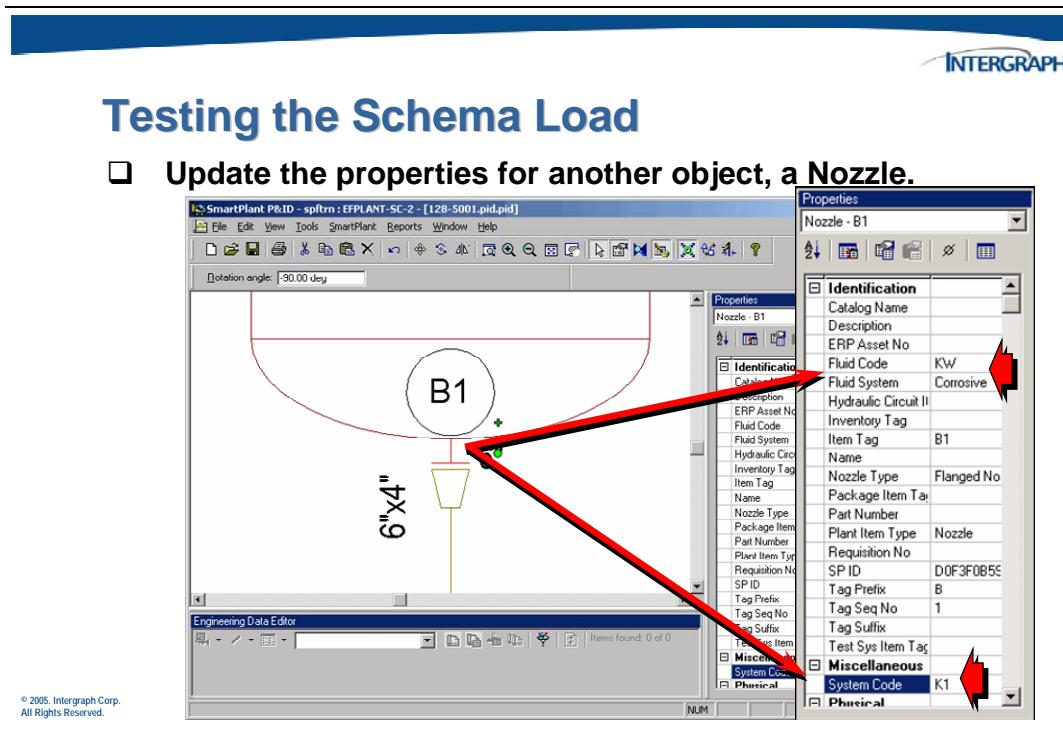
Repeat the update procedure for an instrument object.



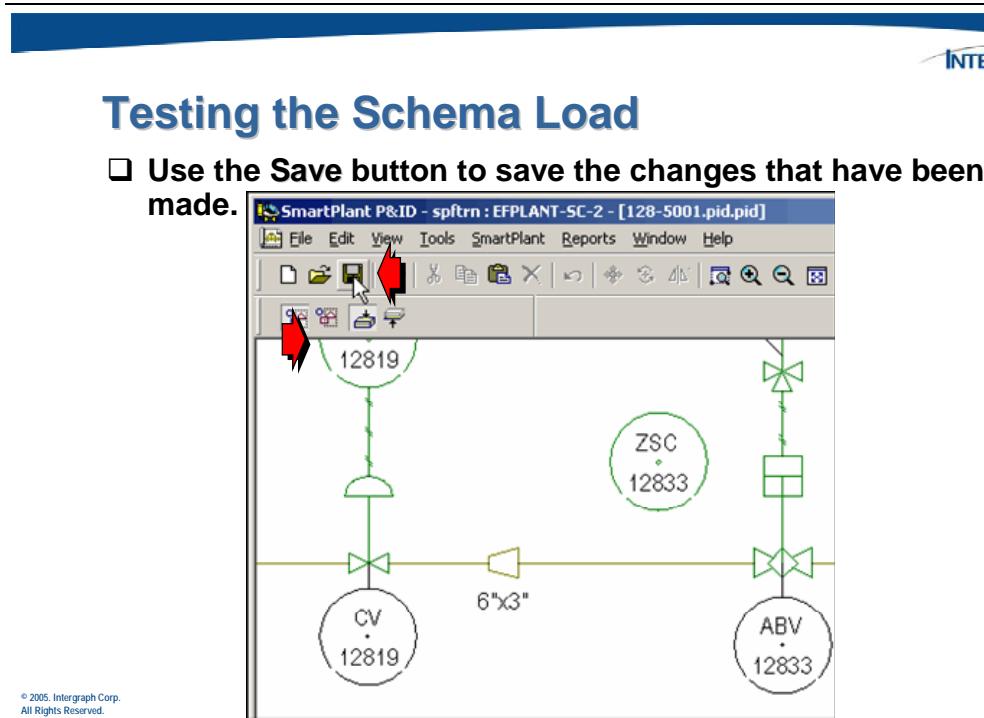
Enter or select some values for the new properties.



Repeat the update for one of the nozzle objects.

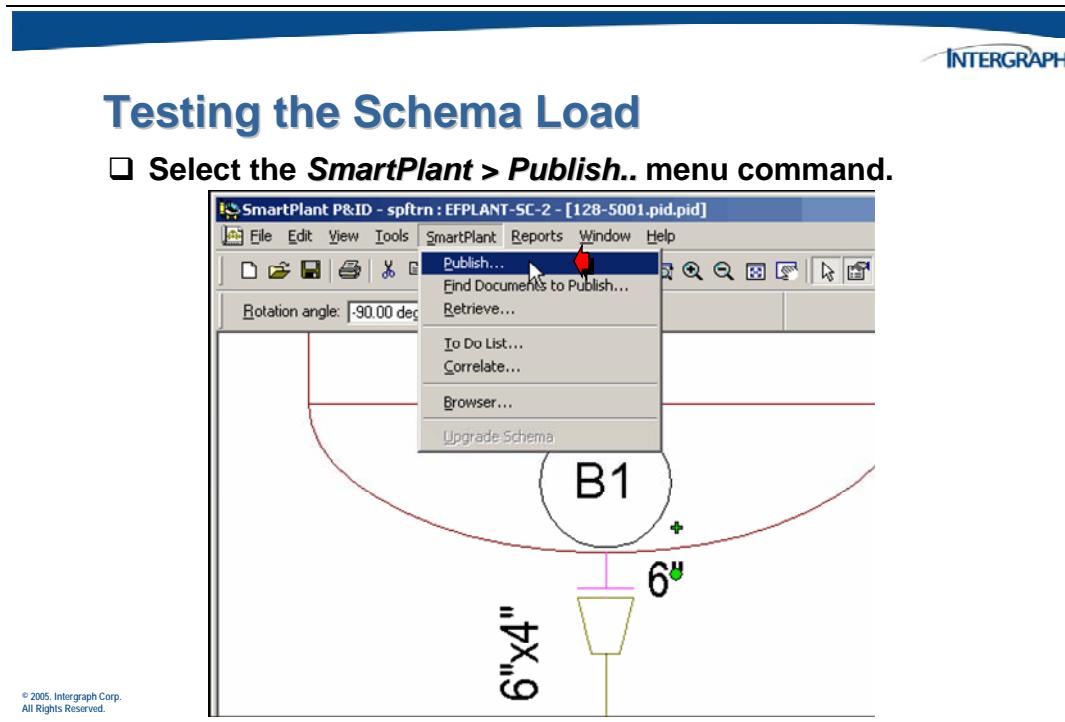


Save the changes that have been made before publishing a new version of this document.



9.2.1 Publishing Documents

To perform a publish operation; use the **Publish** command from SmartPlant P&ID.

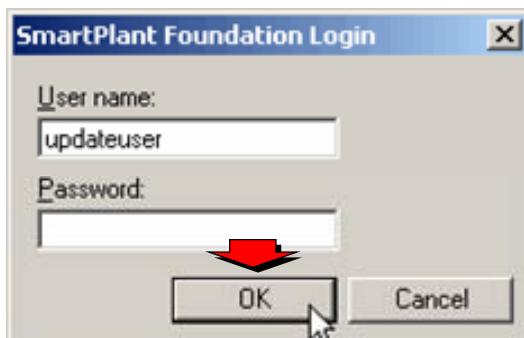


A *SmartPlant Foundation Login* dialog box will be displayed allowing you to specify a user name to be used for the publish operation.



Testing the Schema Load

- Enter a valid SPF user name and password and click **OK** to have the tool log in and publish a document.



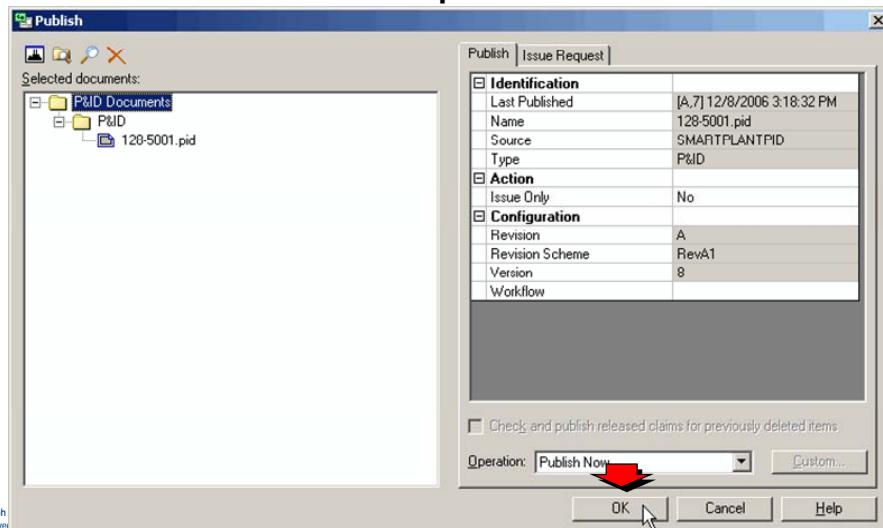
© 2005, Intergraph Corp.
All Rights Reserved.

The *Publish* dialog box will display.



Testing the Schema Load

- Click **OK** to continue and publish the listed documents.

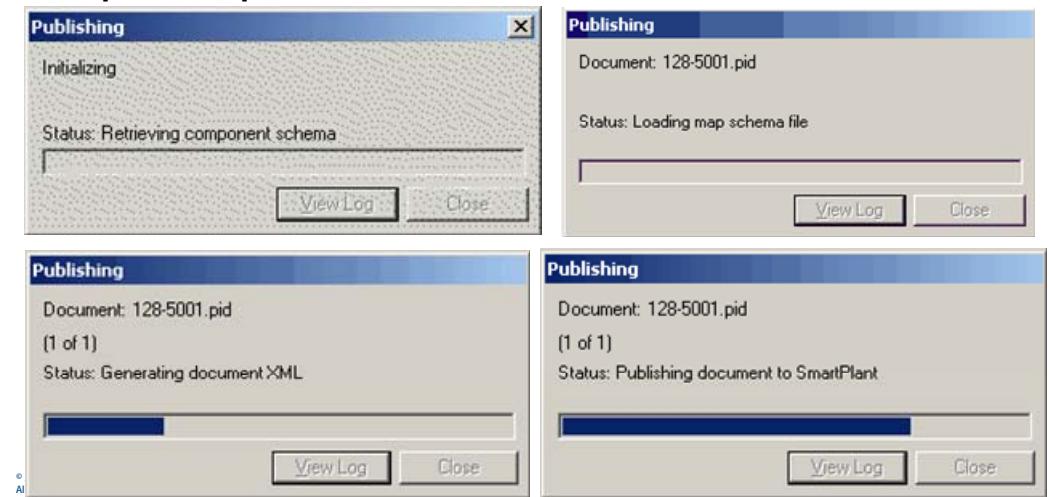


© 2005, Intergraph
All Rights Reserved



Testing the Schema Load

The **Publishing** dialog boxes will display the status of the publish operation.

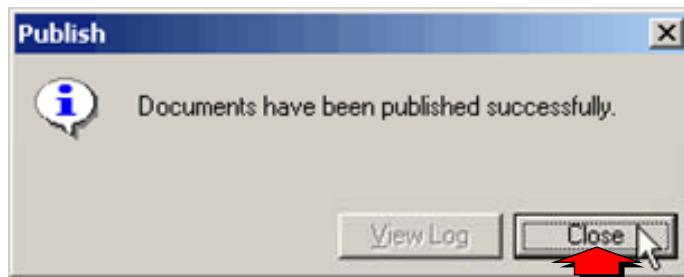


A dialog box will display after some time to report success or failure of the publish.

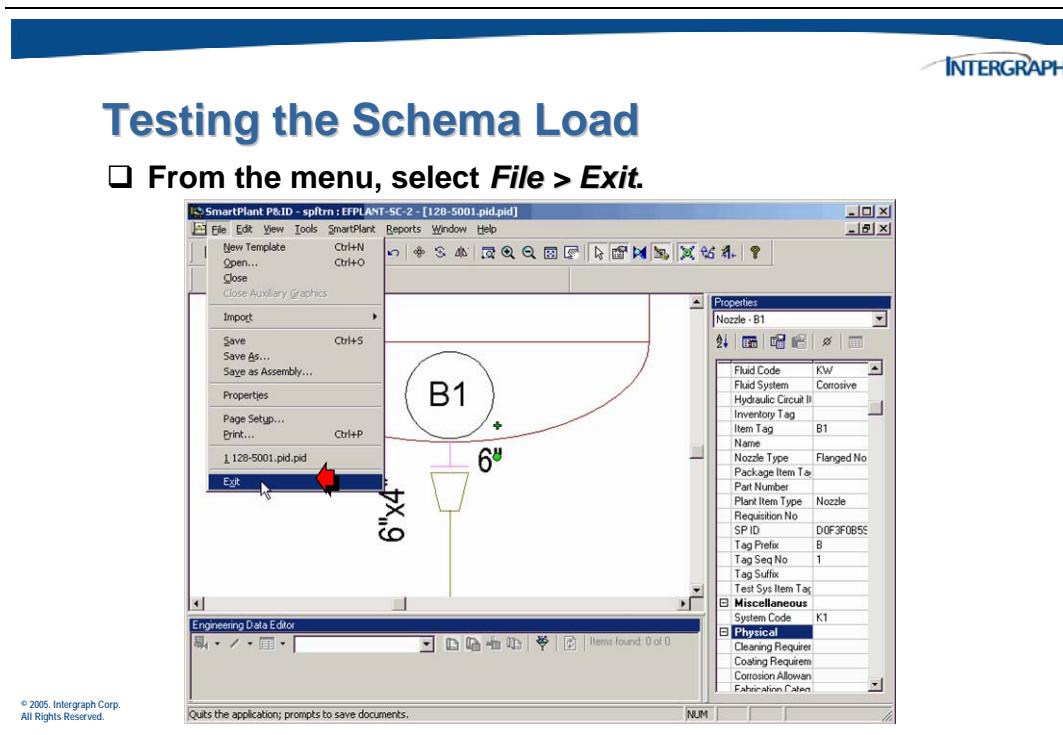


Testing the Schema Load

- When the results dialog box displays, click **Close**.



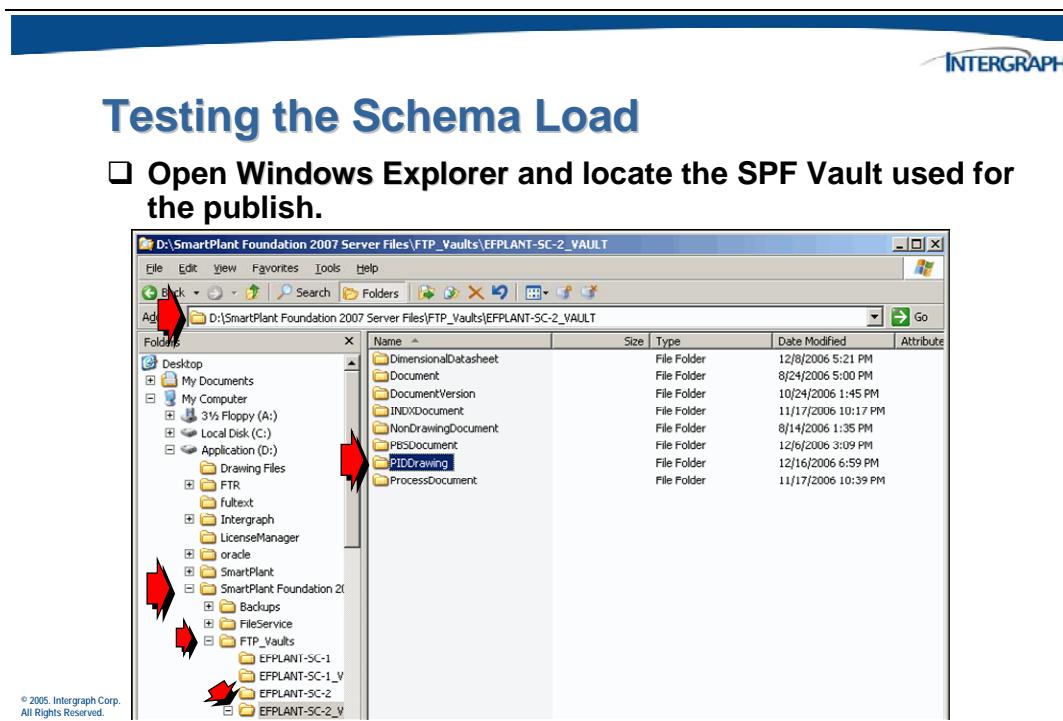
Now that the document has been successfully published, exit from SmartPlant P&ID, locate and review the XML file that has been generated.



9.3 Viewing the Created XML File

The published XML has been sent to the SmartPlant Foundation vault. Use the schema editor to open this XML file and locate the updated pipe run, instrument and nozzle objects. This will allow you to verify that the custom properties are being published.

The default location for the SPF vaults is \SmartPlant Foundation 2007 Server Files\FTP_Vaults\.

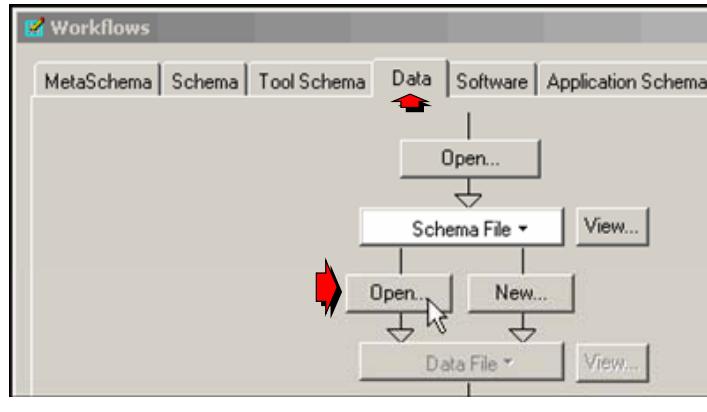


Use the schema editor to review the contents of the published XML files. Once the schema editor has been started, use the **EFSchema.cfg** file to open the schemas. You will also need to use the open data file command (see chapter 6) to open the published xml.



Testing the Schema Load

- In the **Workflows** dialog box, click the **Open** button to open a published **XML** data file.



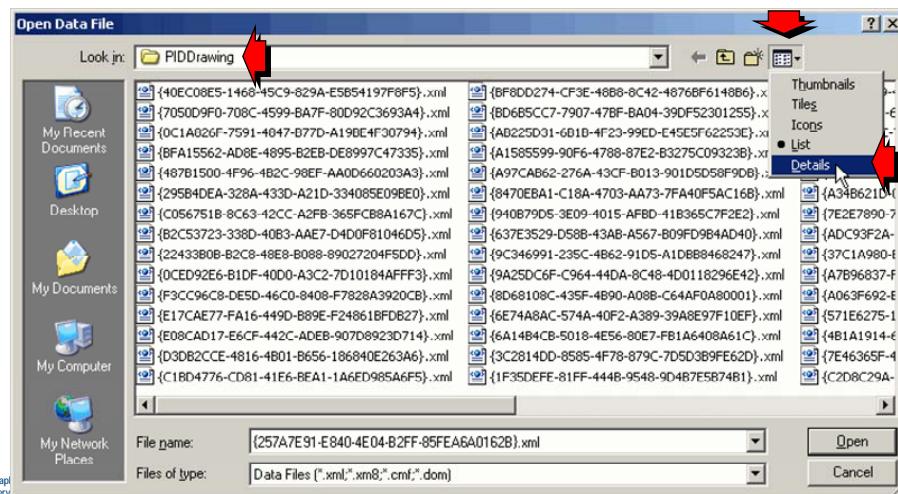
© 2005, Intergraph Corp.
All Rights Reserved.

Open the **PIDDrawing** folder to see a list of available XML files. You can use the Views button on the tool bar to sort the folder contents by date.



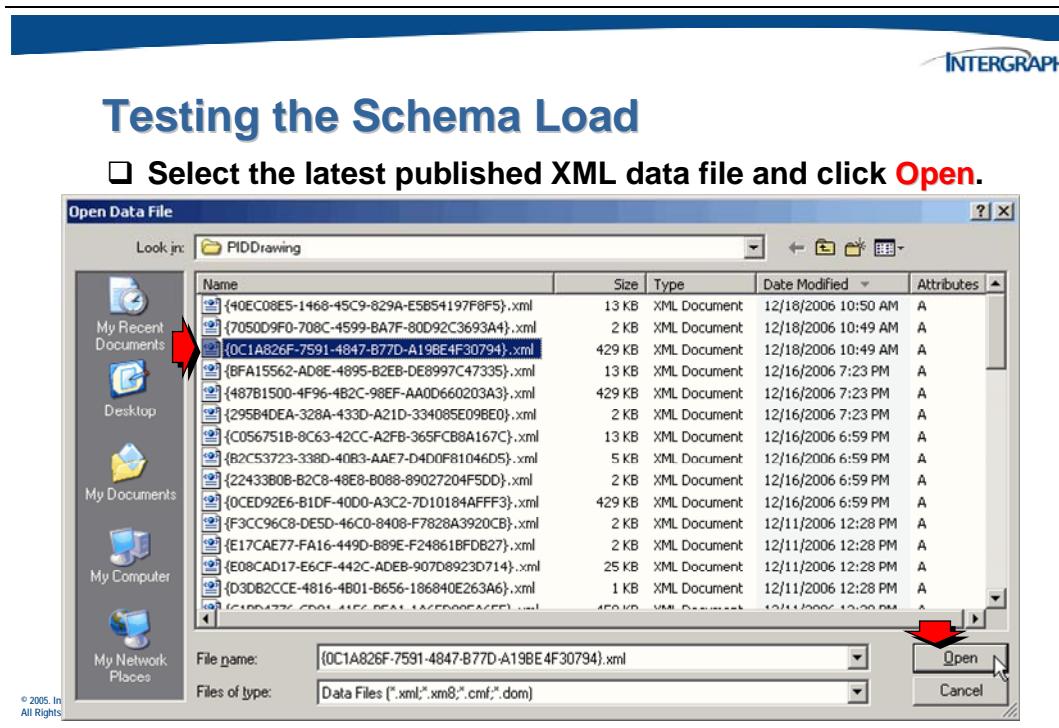
Testing the Schema Load

- In the **Open Data File** dialog box, select the vault folder, click the Views button and choose **Details**.

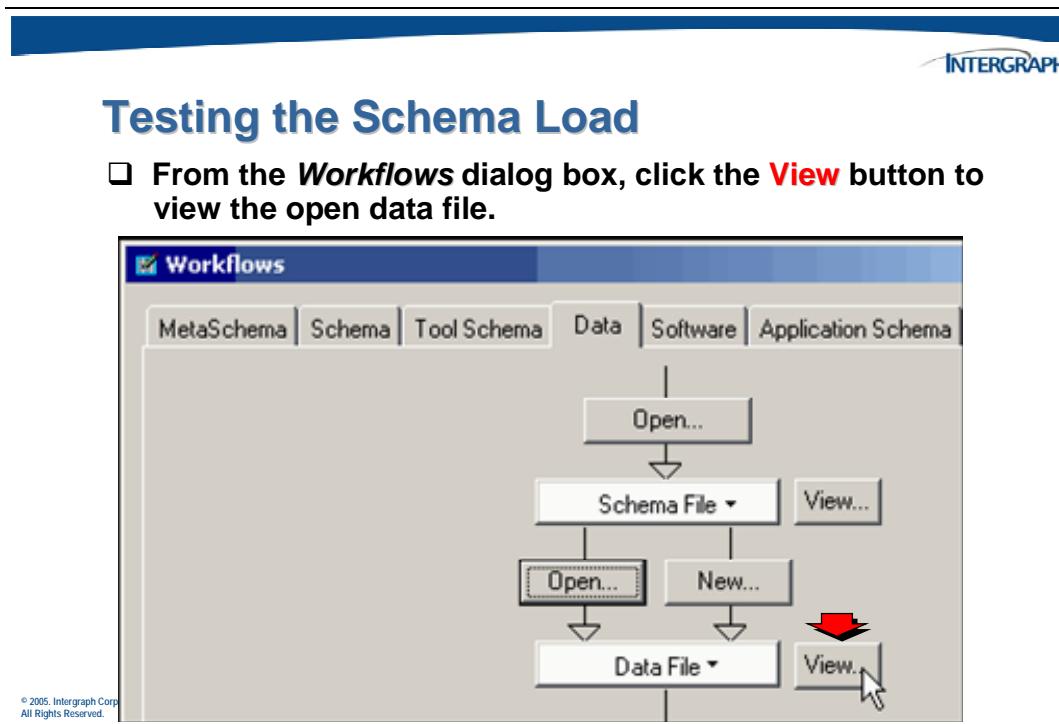


© 2005, Intergraph Corp.
All Rights Reserved.

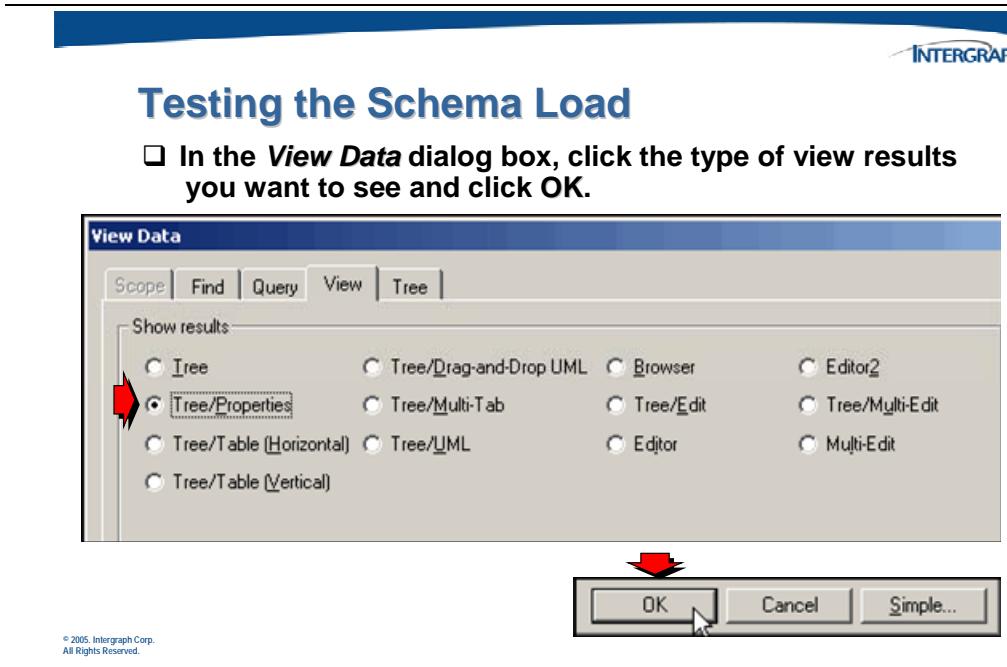
This vault folder, **PIDDrawing**, is the server location where the published XML file is stored. The published file will be one of the latest ones in the folder.



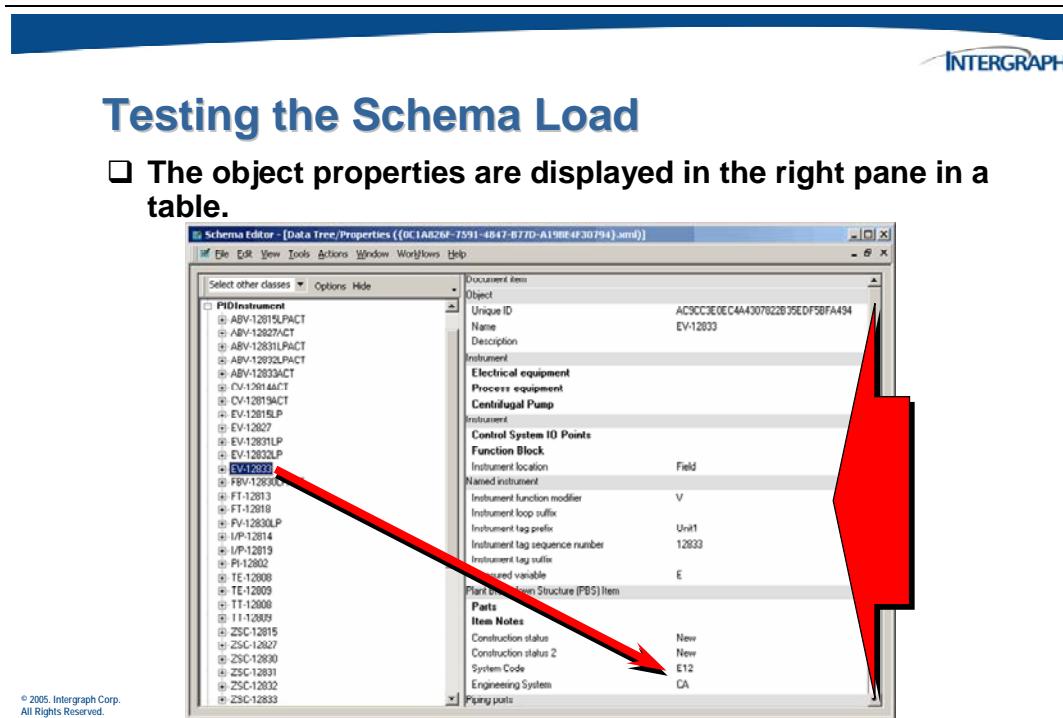
After the published XML file has been opened, use the schema editor to view it.



The schema *Tree/Properties* view allows you to traverse relationships in the tree and see information for items selected in the tree view in a table format.



The *Data Tree/Properties* window displays.



The displayed table shows the properties of the selected object. Note the values for the custom properties.

Review some of the other objects that were published.

The screenshot shows the Schema Editor interface with the title bar "Schema Editor - [Data Tree/Properties ({0C1A826F-7591-4847-B77D-A19BE4F30794}.xml)]". The left pane displays a tree view of objects under "Select other classes", including PIDDrawing, PIDInlineInstrument, PIDInstrument, PIDInstrumentLoop, PIDMechanicalEquipment, PIDNozzle, PIDPipeline, and PIDPipingBranchPoint. A red arrow points from the "Object" section of the right pane to the "System Code" field, which is set to "L12".

Object	
Unique ID	623FD2CA1C944579B49005148D9A07A3
Name	F-12830LP
Description	Plant Breakdown Structure (PBS) Item
Parts	
Item Notes	
Construction status	New
Construction status 2	New
System Code	L12
Engineering System	CA
Planned facility	

© 2005, Intergraph Corp.
All Rights Reserved.

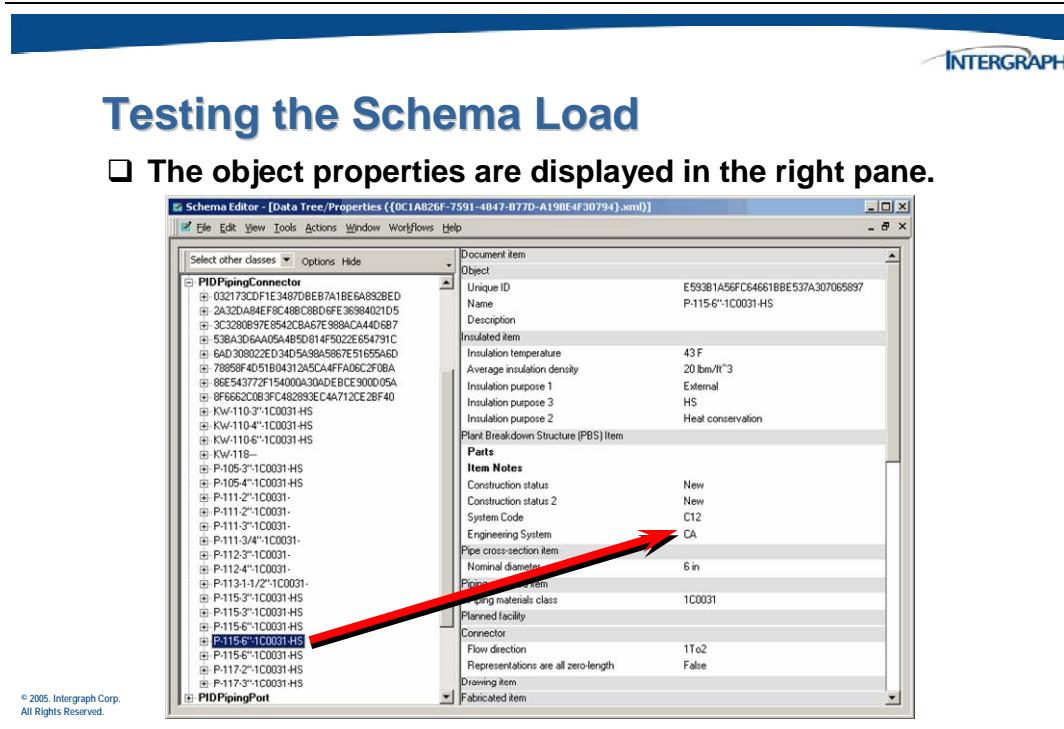
Locate nozzle **B1** and review the custom property values.

The screenshot shows the Schema Editor interface with the title bar "Schema Editor - [Data Tree/Properties ({0C1A826F-7591-4847-B77D-A19BE4F30794}.xml)]". The left pane displays a tree view of objects under "Select other classes", including PIDDrawing, PIDInlineInstrument, PIDInstrument, PIDInstrumentLoop, PIDMechanicalEquipment, PIDNozzle, PIDPipeline, and PIDPipingBranchPoint. A red arrow points from the "Object" section of the right pane to the "System Code" field, which is set to "K1".

Object	
Unique ID	D0F3F0B5914E48DC89A8D0B47F0380E0
Name	B1
Description	
Insulated item	
Average insulation density	20 lbm/ft ³
Insulation purpose 1	External
Insulation purpose 3	HS
Insulation purpose 2	Heat conservation
Insulation temperature	43 F
Plant Breakdown Structure (PBS) Item	
Parts	
Item Notes	
Construction status	New
Construction status 2	New
System Code	K1
Engineering System	CA

© 2005, Intergraph Corp.
All Rights Reserved.

In the following example, the custom property values for the pipe run are shown.

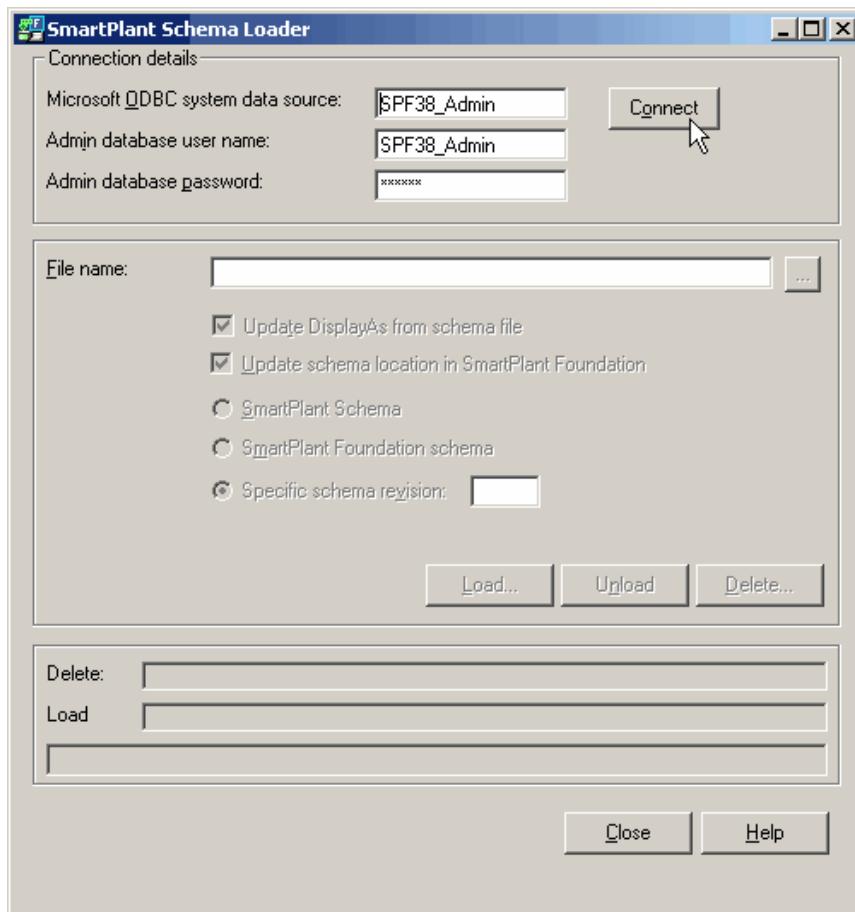


9.4 Activity – Loading and Testing the Schema Changes

The objective of this activity is to use the Schema Loader to load the extensions created for SmartPlant P&ID into SmartPlant Foundation and test the publish functionality. Under the SPF root directory (D:\smartplant\foundation\2007) double click the file **SmartPlantSchemaLoader.exe**. Once the load is complete you will login to SmartPlant P&ID to test the model by publishing some updated data values.

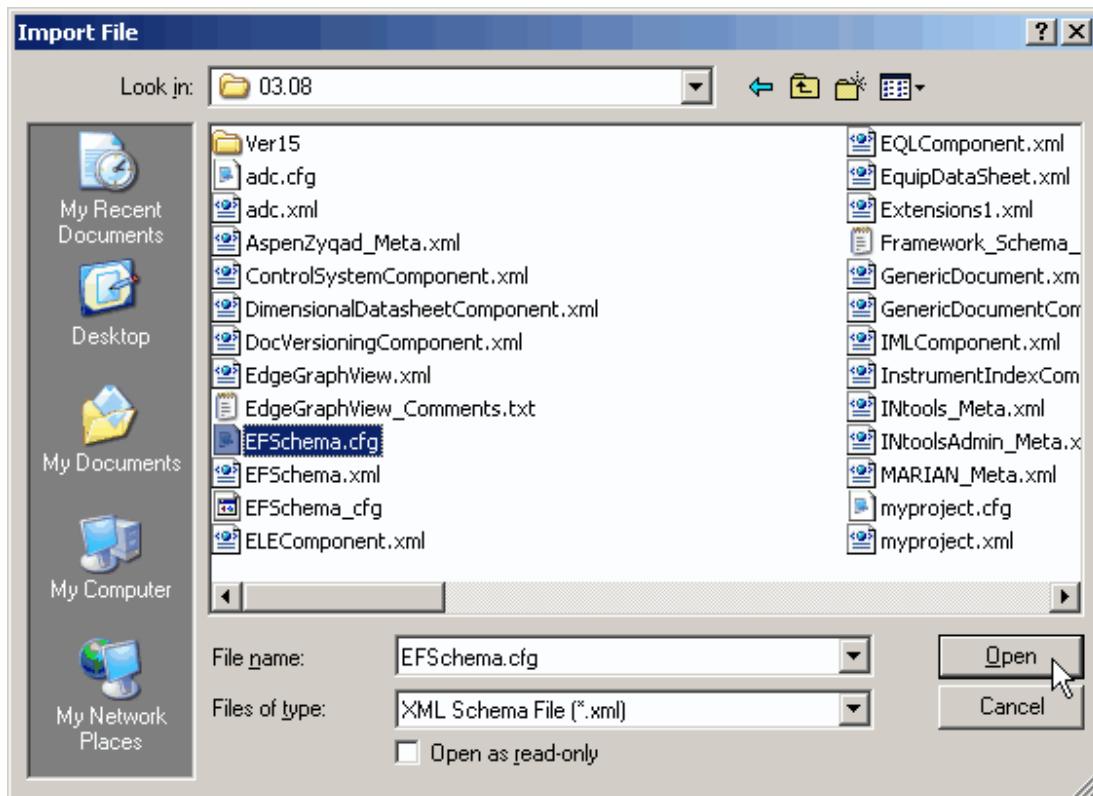
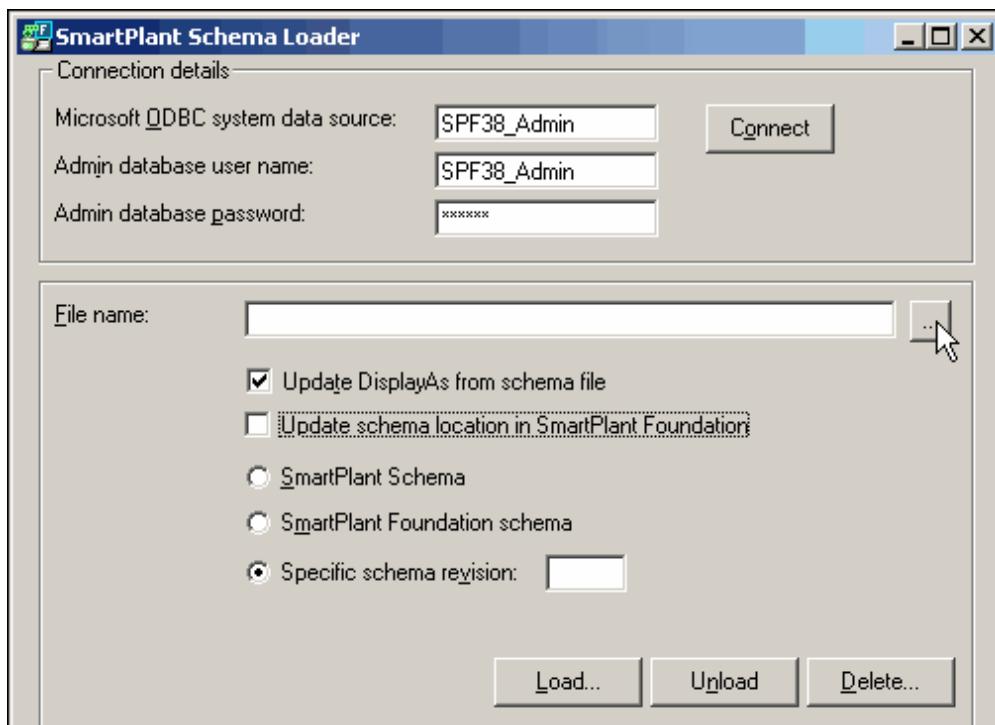
Load the SmartPlant Schema files into SPF

1. Verify the login values are correct in the dialog.

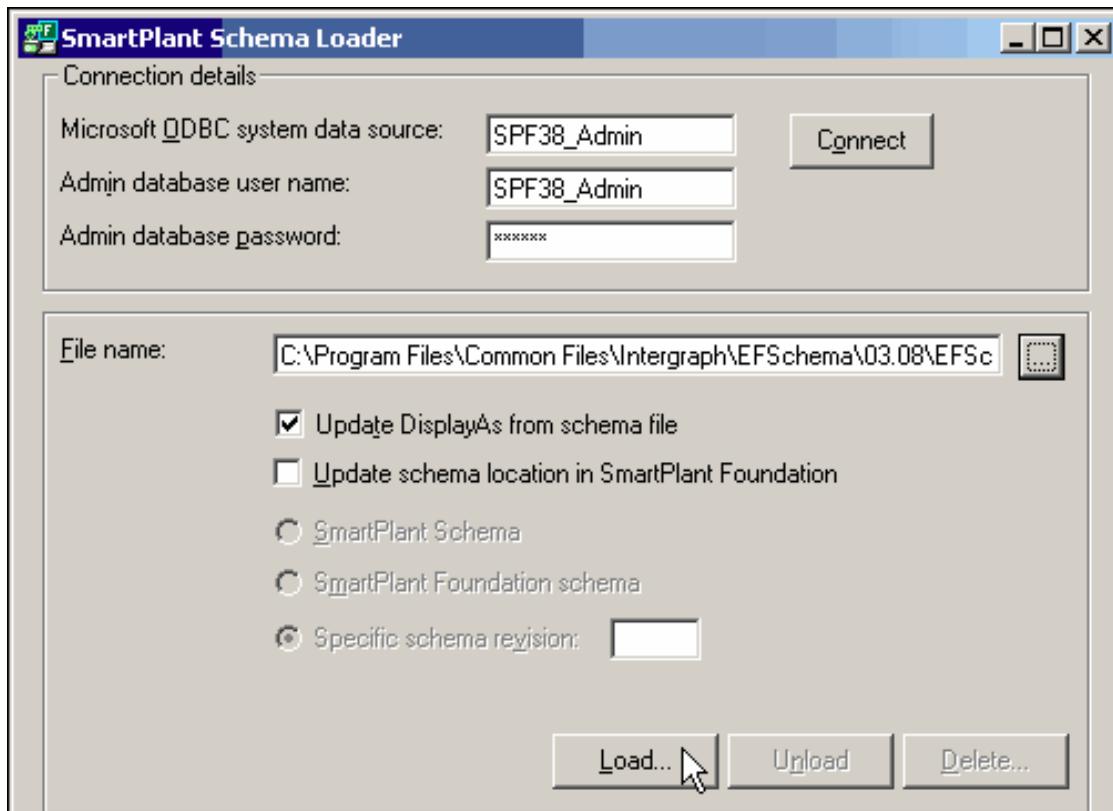


2. Click the **Connect** button to login

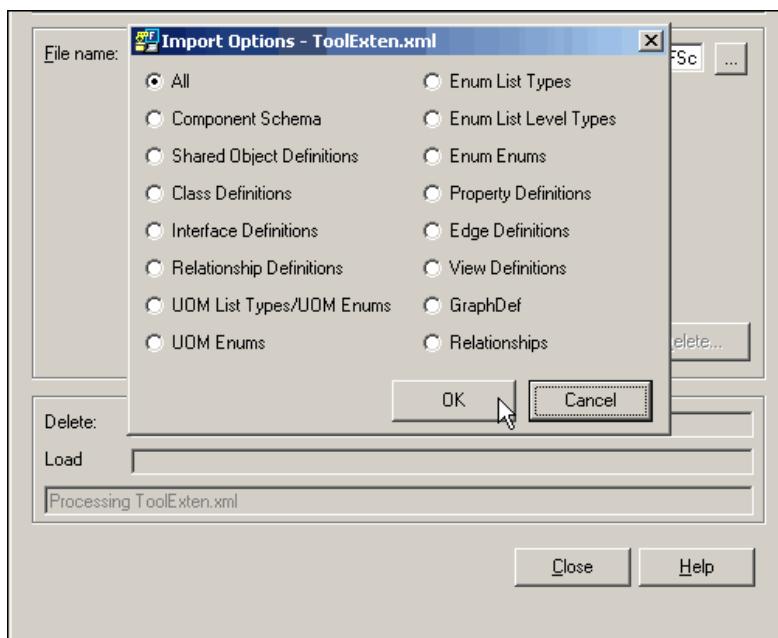
3. Use the **File name:** browse button (...) to query for the file **EFSchema.cfg** used previously in this class. The default share, \\Pimdemo1\\EFSchema is OK to use.



4. Unselect the *Update schema location in SmartPlant Foundation* toggle and click the **Load** button to load the SmartPlant Schema data.



5. Select **All** in the option dialog which will display and then click **OK**.



6. Open Windows Explorer and clean up the temporary cache folder, **C:\Documents and Settings\spfuser\Local Settings\Temp\EFSchemaCache\03.08\pimdemo1_spf38asp**. Delete all the files in this folder.
7. Open the schema editor using the *EFSchema.cfg* configuration and generate a new set of component schemas.
 - Click **File Configurations > Open Configuration....**
 - Browse to **C:\Program Files\Common\Intergraph\EFSchema\03.08** in the *Open Configuration File* dialog box.
 - Select **EFSchema.cfg** file and click **Open**.
 - On the *Schema Editor* menu, click **Tools > Schema > Generate Component Schemas**.
8. Use the examples beginning on page 9-10 to open SmartPlant P&ID and update some of the placed pipe runs, instruments and nozzles. Be sure and update Pipe Run **137-P** (just to the left of the vertical tank and instrument ABV-12833, upper left of center in the drawing). Do NOT change the *Fluid System/Fluid Code* for this Pipe Run.
9. Once the objects have been updated, use SmartPlant to publish the changed values.
10. Use the Schema Editor to view the results of the publish by opening the published XML file from the vault.
11. When you have finished, take a short break until the other students have finished their activity.

10

C H A P T E R

Mapping with SmartPlant Instrumentation

10. Mapping with SmartPlant Instrumentation

This chapter will demonstrate how to further extend the SmartPlant schema for SmartPlant Foundation along with the authoring tools (SmartPlant P&ID and SmartPlant Instrumentation) that integrate with it to create an integrated engineering system.



Extending the SmartPlant Schema

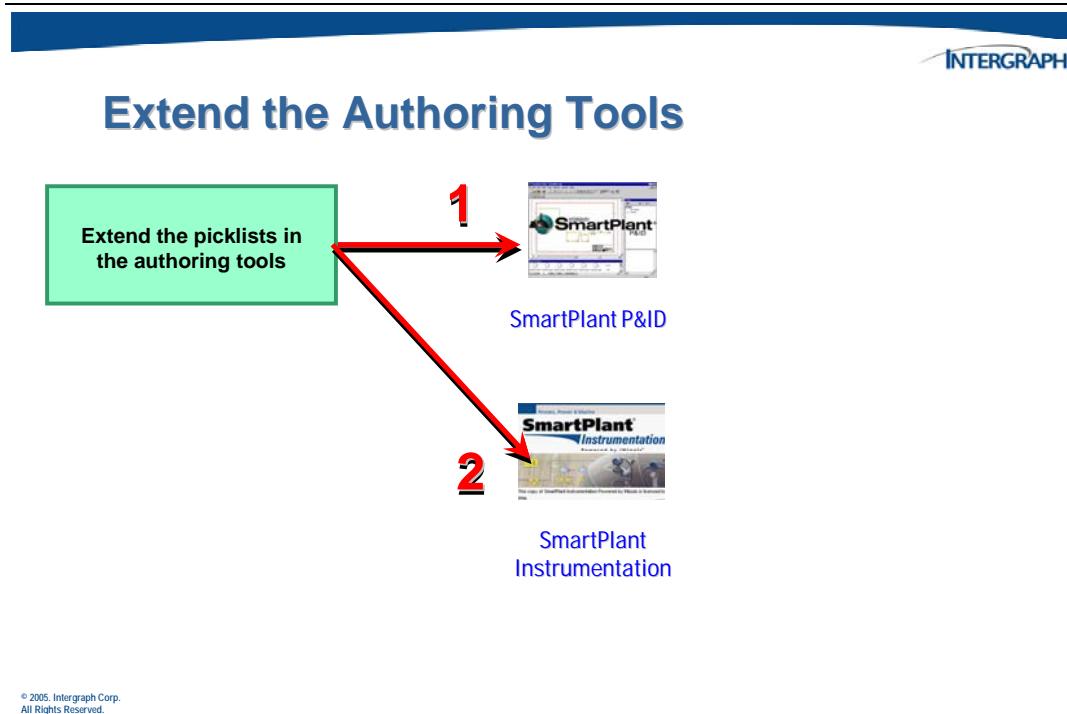
This example will add the “**preconstruction**” status to the existing **Construction Status** enumerated list.

The example includes **SmartPlant Foundation** and the following authoring tools: **SmartPlant P&ID** and **SmartPlant Instrumentation**



10.1 Modifying the Authoring Tools

There is no specific order to follow when extending the SmartPlant system. It may be easier and more logical to add the new picklist values to the authoring tool first. Once that has been accomplished, when extending the schema files, you will know exactly what values must be added there. It is critical that the value added to the tool schema match what has been added to the tool meta data.



© 2005. Intergraph Corp.
All Rights Reserved.

In this example, two authoring tools are being extended so both will be modified in order to complete the extensions.

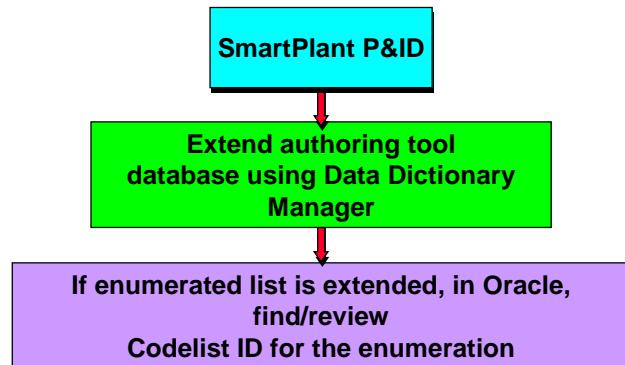
10.2 Extending an Enumerated List in SmartPlant P&ID

The first step to extending the SmartPlant system is to add the new information in one of the authoring tools. In the example in this chapter, the SmartPlant P&ID and SmartPlant Instrumentation authoring tools are demonstrated.



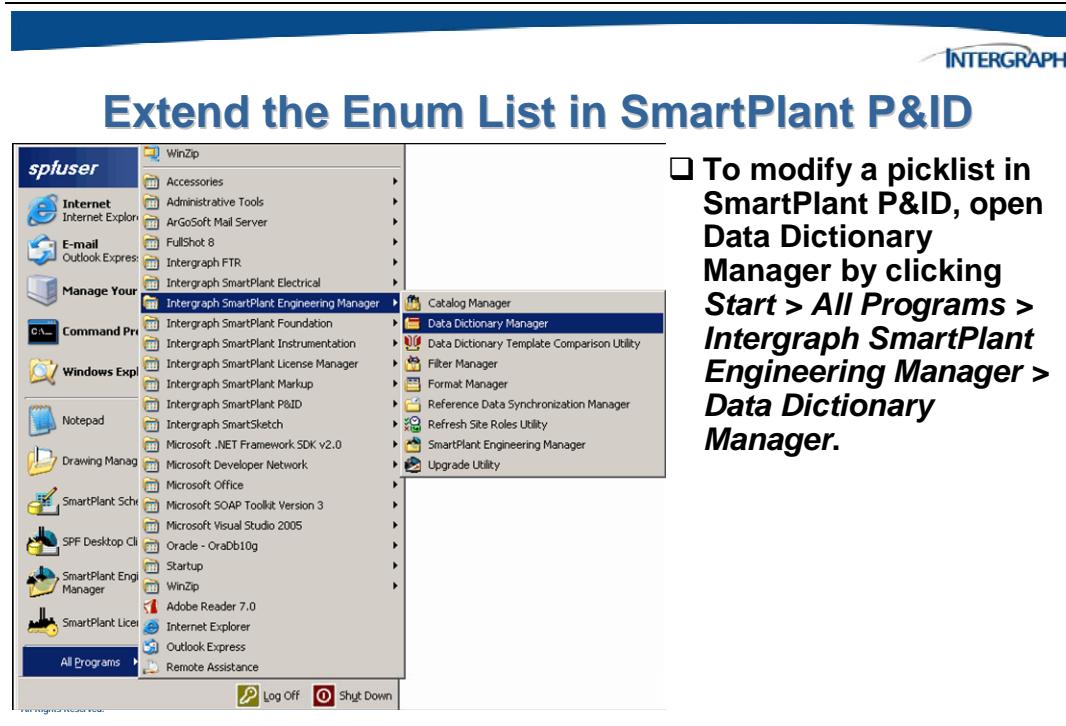
Extend the Enum List in SmartPlant P&ID

The following workflow describes the steps necessary to extend the SmartPlant P&ID authoring tool.



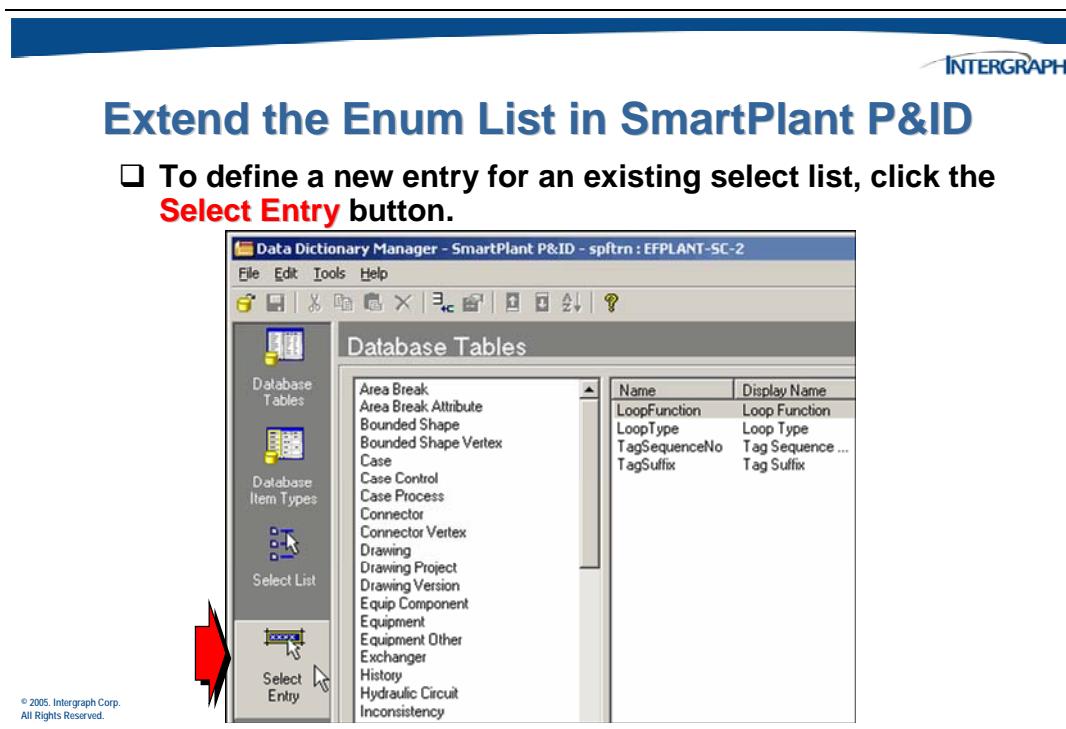
© 2005. Intergraph Corp.
All Rights Reserved.

In SmartPlant P&ID, start Data Dictionary Manager (DDM) for the plant where you want to extend the construction status.

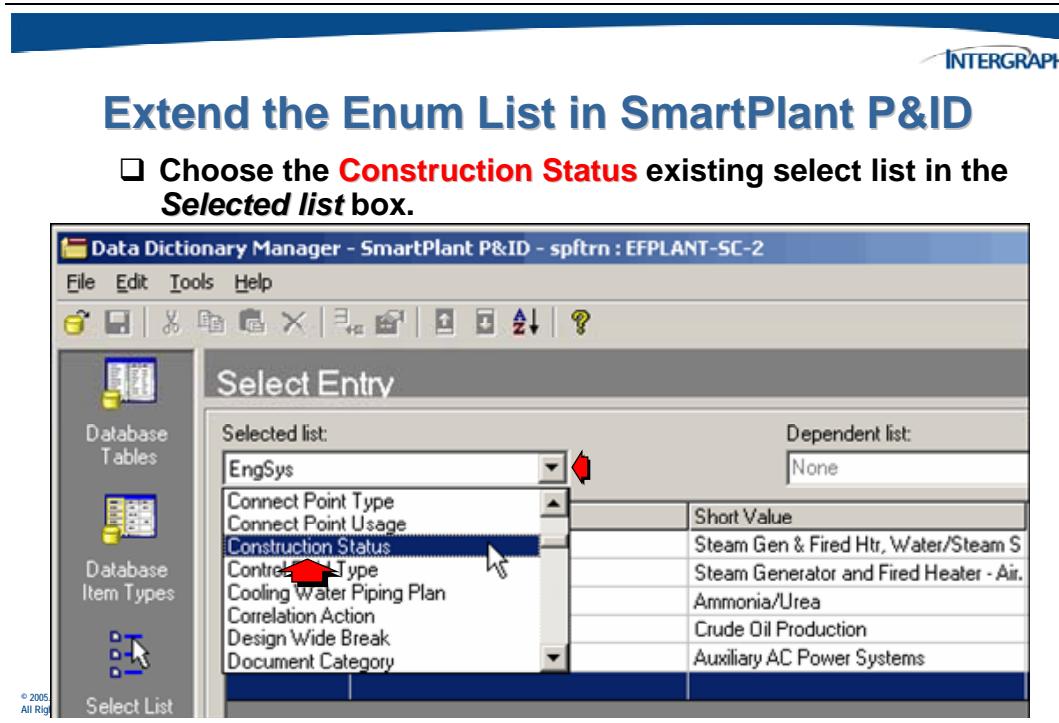


- To modify a picklist in SmartPlant P&ID, open Data Dictionary Manager by clicking Start > All Programs > Intergraph SmartPlant Engineering Manager > Data Dictionary Manager.

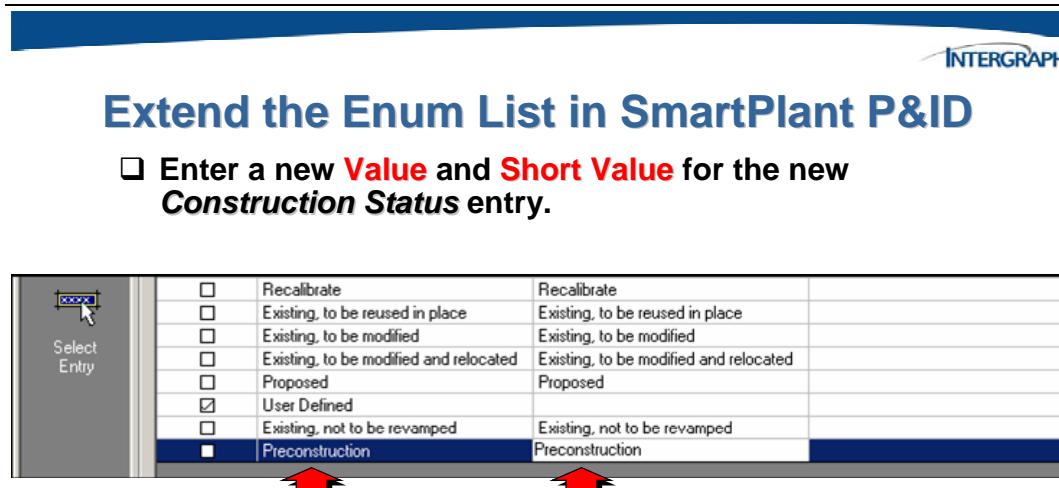
The *Data Dictionary Manager* application window will display.



The *Select Entry* dialog will display.



Select the last entry in the list view and type “Preconstruction” as the *Value* and *Short Value*.

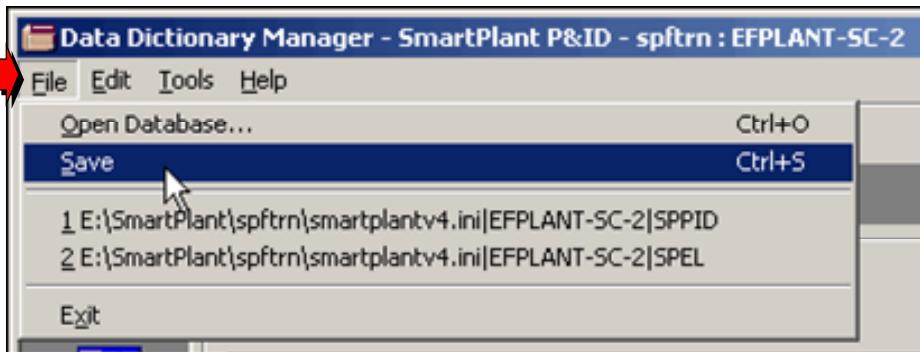


Save the changes and close Data Dictionary Manager.



Extend the Enum List in SmartPlant P&ID

- From the menu, select **File > Save**.



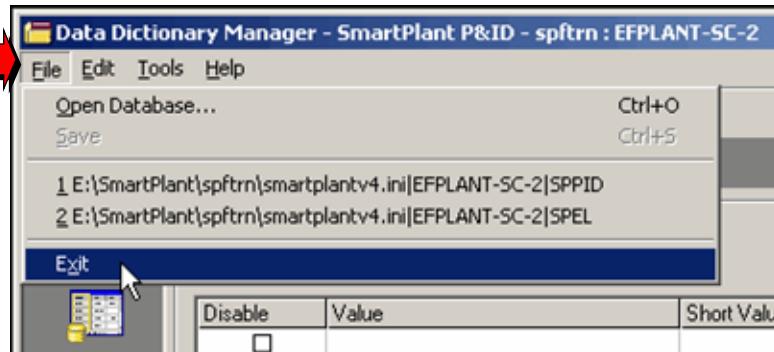
© 2005, Intergraph Corp.
All Rights Reserved.

You may be prompted to save the changes.



Extend the Enum List in SmartPlant P&ID

- Select **File > Exit** from the menu.



© 2005, Intergraph Corp.
All Rights Reserved.

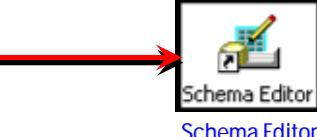
10.2.1 Extending the SPPID Tool Map Schema

Before you can define mapping for these new entries in the tool map schema, the enumerated list must first be extended. This is done automatically when the tool map schema and the SmartPlant schema are synchronized. The steps necessary to modify the master schema involves using the schema editor to add new entries to an existing enumerated list. After the entry has been added, the schema must be saved and before publishing, new copies of the component schemas generated.

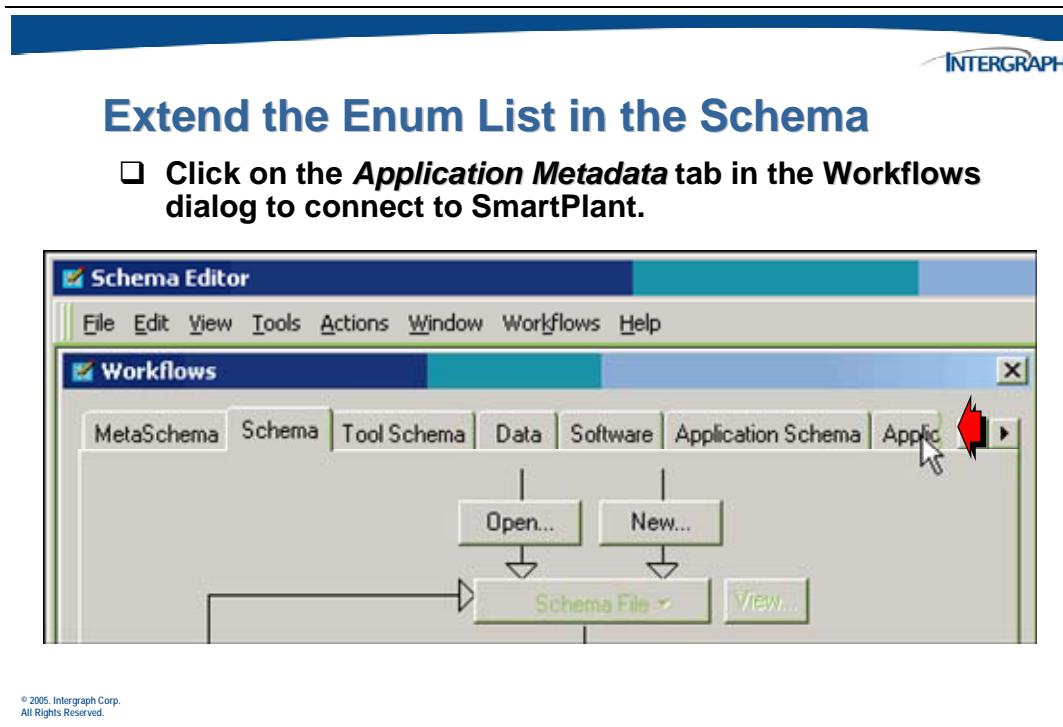


Extend the SmartPlant Schema

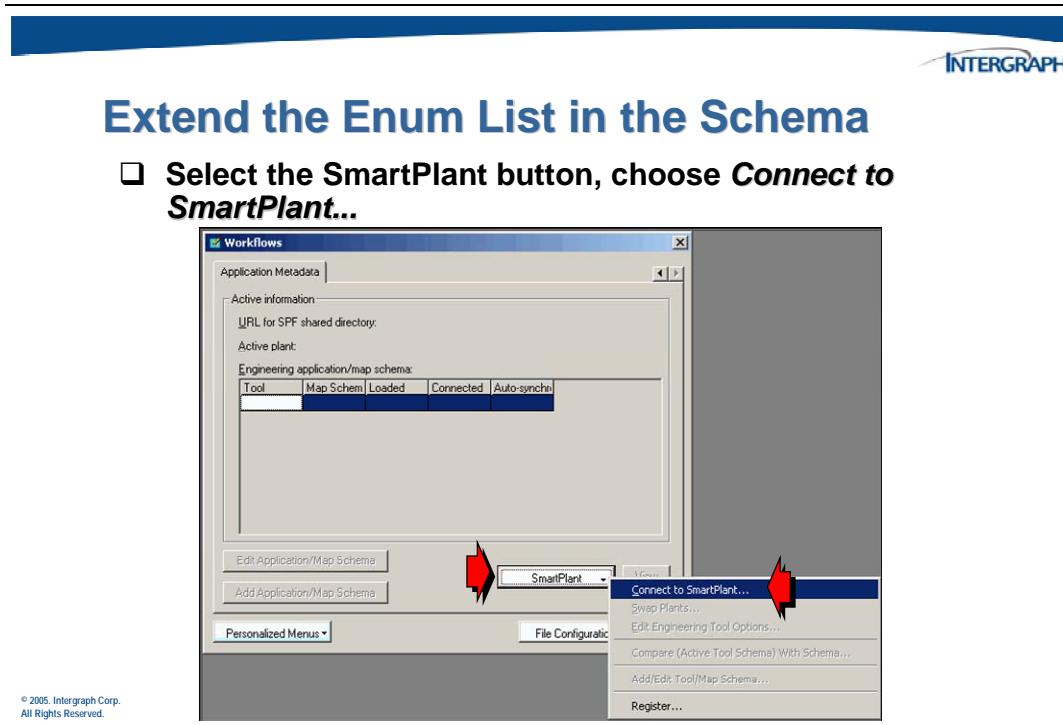
Add the new enumerated entry to the SPPID tool map schema and the SmartPlant schema and map it



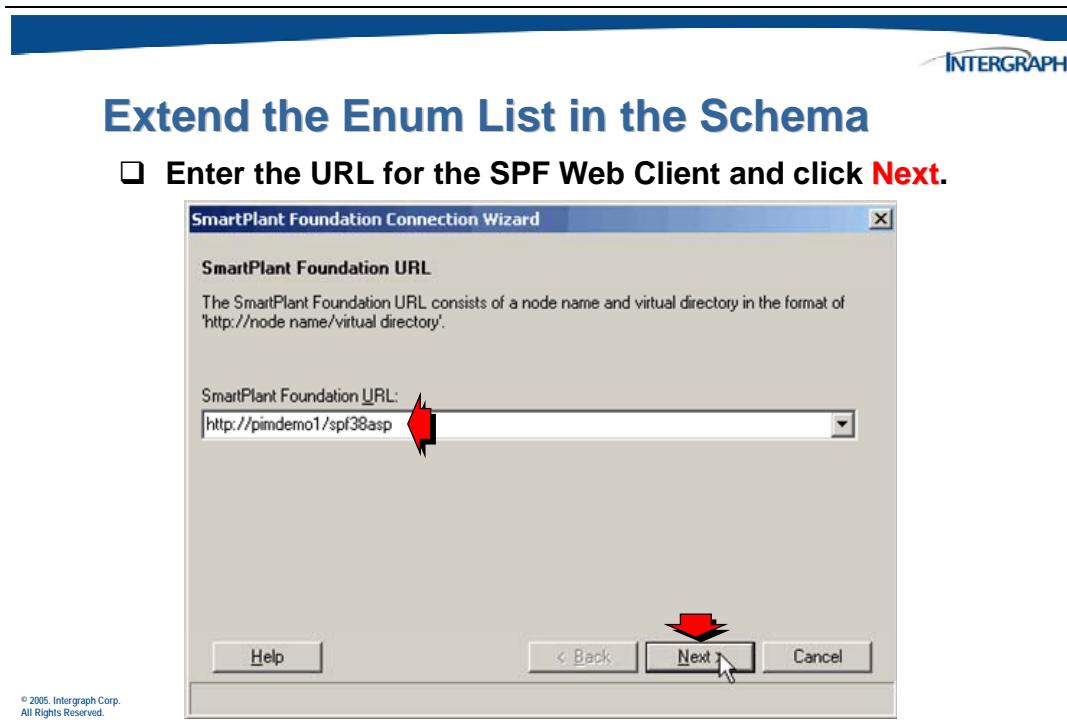
Begin by starting the schema editor and selecting the **Application Metadata** tab.



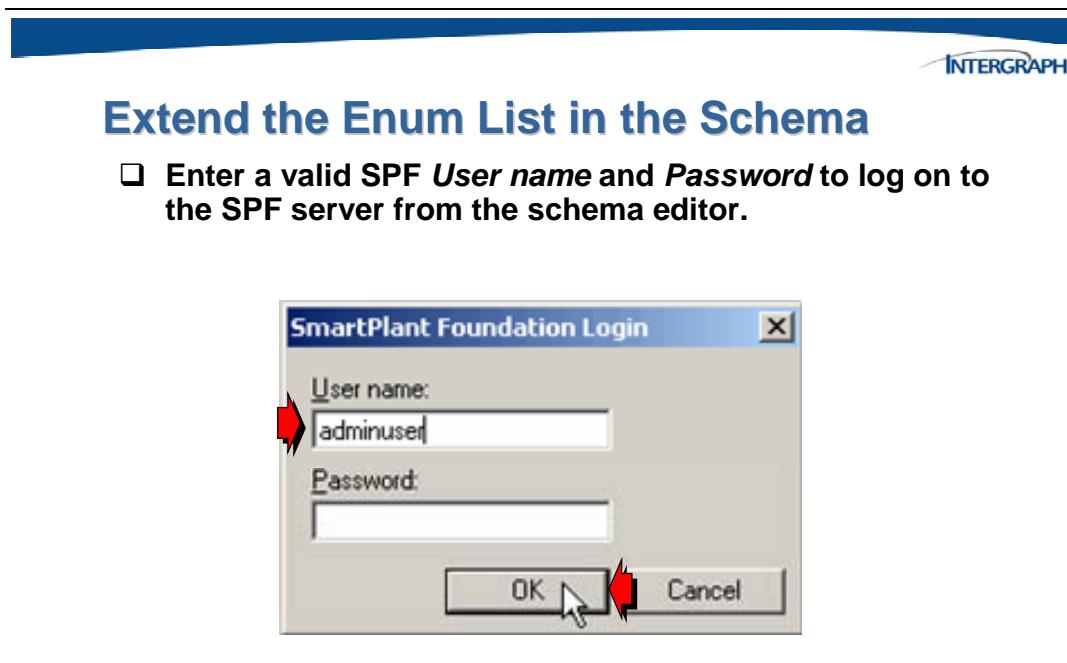
The *Application Metadata* dialog will display.



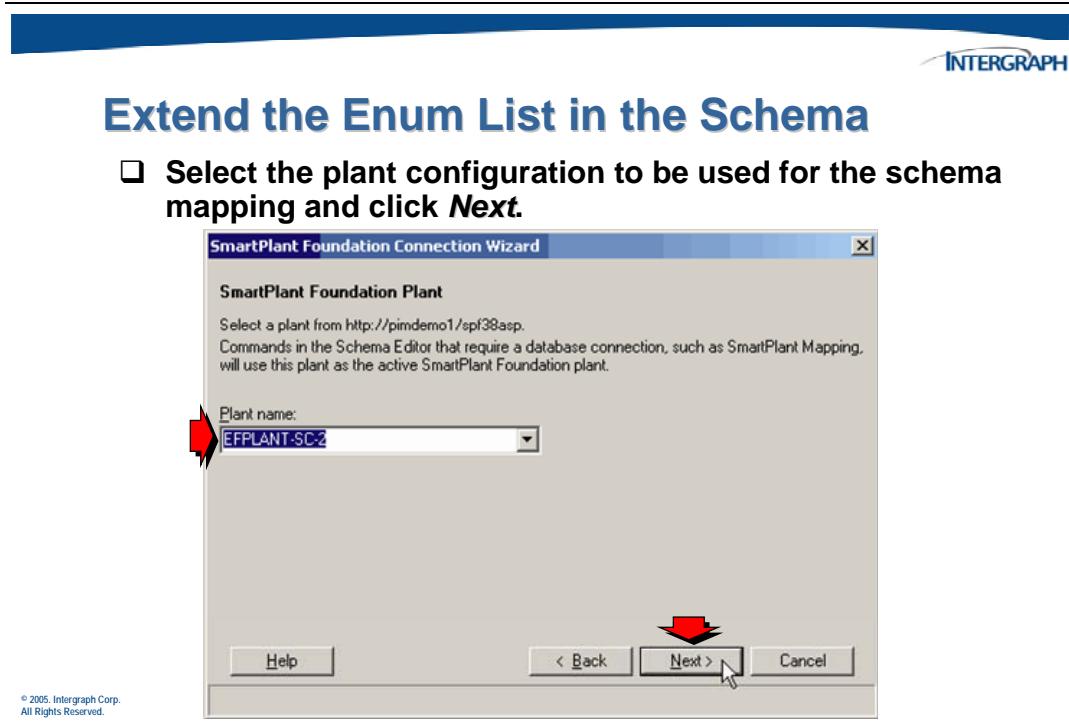
The *SmartPlant Foundation Connection Wizard* will be displayed.



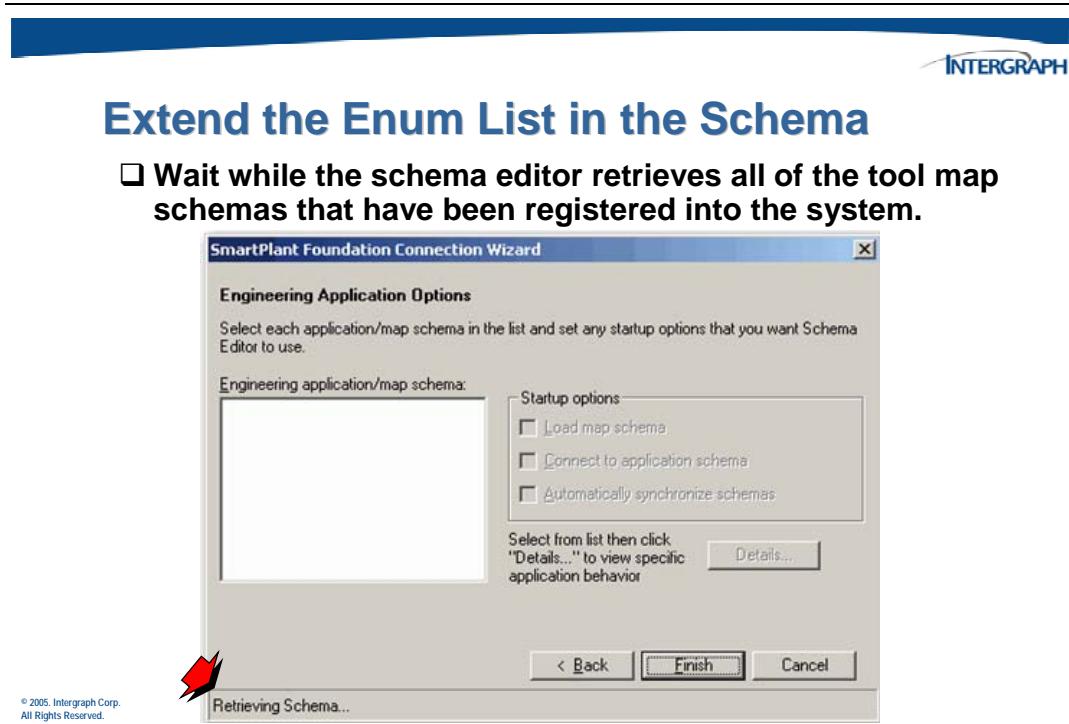
In the *SmartPlant Foundation URL* field, type the node name and virtual directory of the SmartPlant Foundation database with which you want to connect.



For the next step, you will need to know the SmartPlant Foundation plant configuration that you will use for mapping.



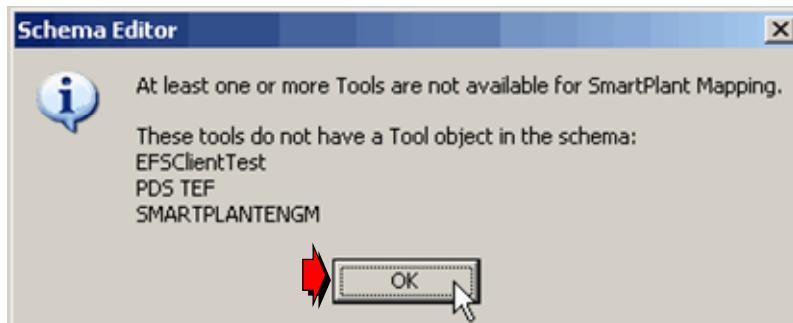
The Schema Editor will go and find the SmartPlant schema and all **registered** tool map schemas.





Extend the Enum List in the Schema

- When a Schema Editor informational window displays, click OK to continue.



© 2005, Intergraph Corp.
All Rights Reserved.

In the above example, the *EFSClientTest* tool has registered with SmartPlant for this plant but no Tool object exists for it. A list of available tool map schemas will be displayed.



Extend the Enum List in the Schema

- Select the tool map schema to be loaded and click Finish.



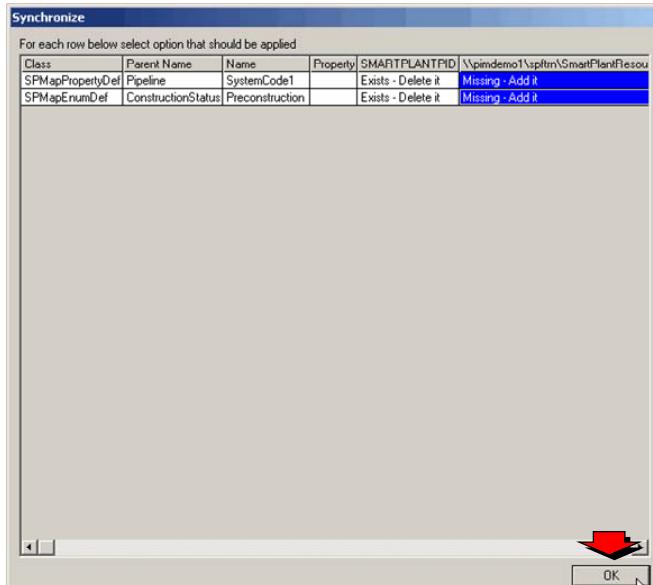
© 2005, Intergraph Corp.
All Rights Reserved.

Selecting a Tool/ToolSchema entry from the list control enables one (or more) of the toggle boxes on the right of the dialog.

After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema.

Extend the Enum List in the Schema

Click OK from the Synchronize dialog to have the schema editor read the tool meta-schema and automatically add the new enum extension to the tool map schema.



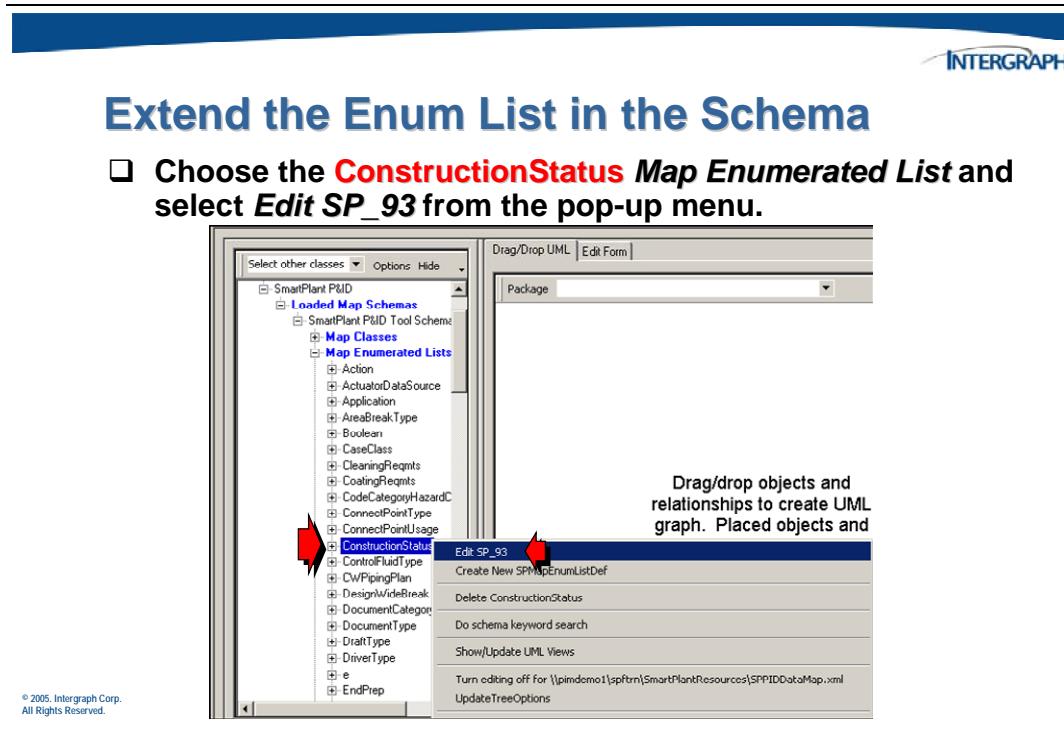
The screenshot shows a 'Synchronize' dialog box. At the top, it says 'For each row below select option that should be applied'. Below is a table with two rows:

Class	Parent Name	Name	Property	SMARTPLANTPID \\spidemo1\\spitm\\SmartPlantResou
SPMapPropertyDef	Pipeline	SystemCode1	Exists - Delete it	Missing - Add it
SPMapEnumDef	ConstructionStatus	Preconstruction	Exists - Delete it	Missing - Add it

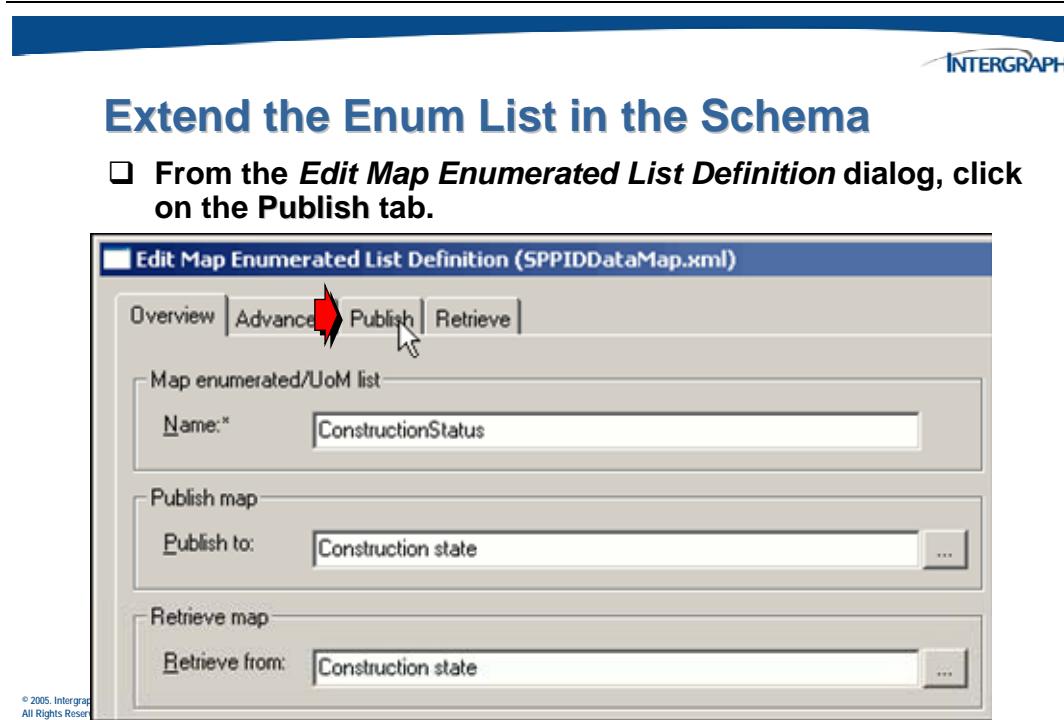
At the bottom right of the dialog, there is a red arrow pointing to the 'OK' button.

The differences between the two map schemas are displayed in the **Synchronize** form. For each difference the user must select (each one has a system-determined pre-selection) which synchronization action to perform.

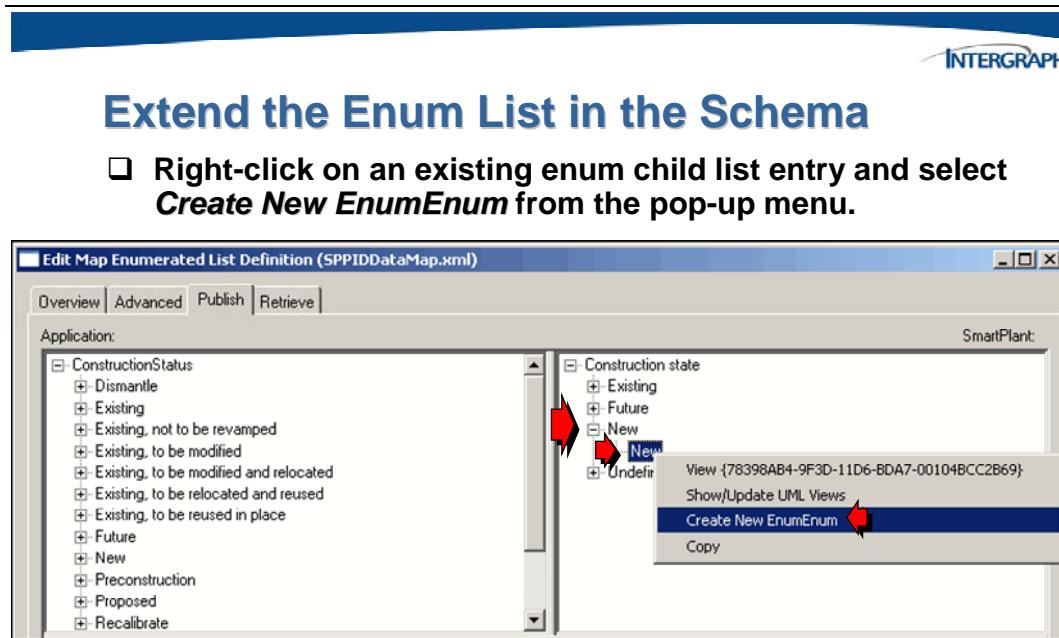
Since the enum entry “Preconstruction” is automatically added to the tool map schema, it must also be manually added to the SmartPlant schema but in the extension file (ToolExten.xml).



The *Edit Map Enumerated List Definition* dialog will display.

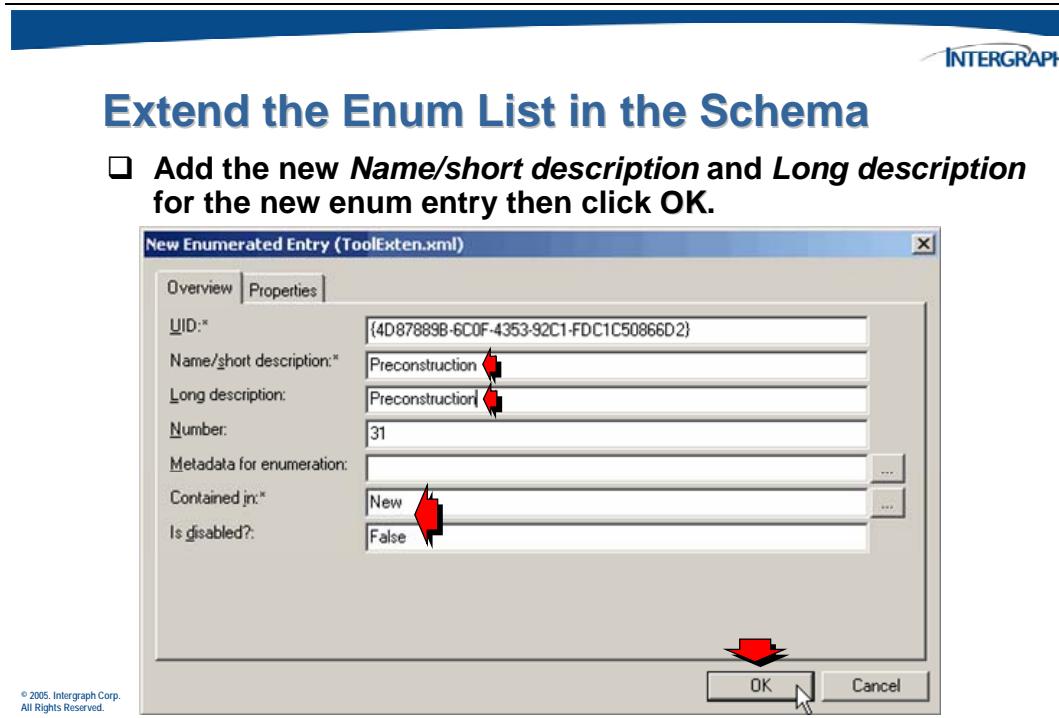


To add the entry in the correct enum child list, expand the list entry *New* and highlight the enum value **New**.



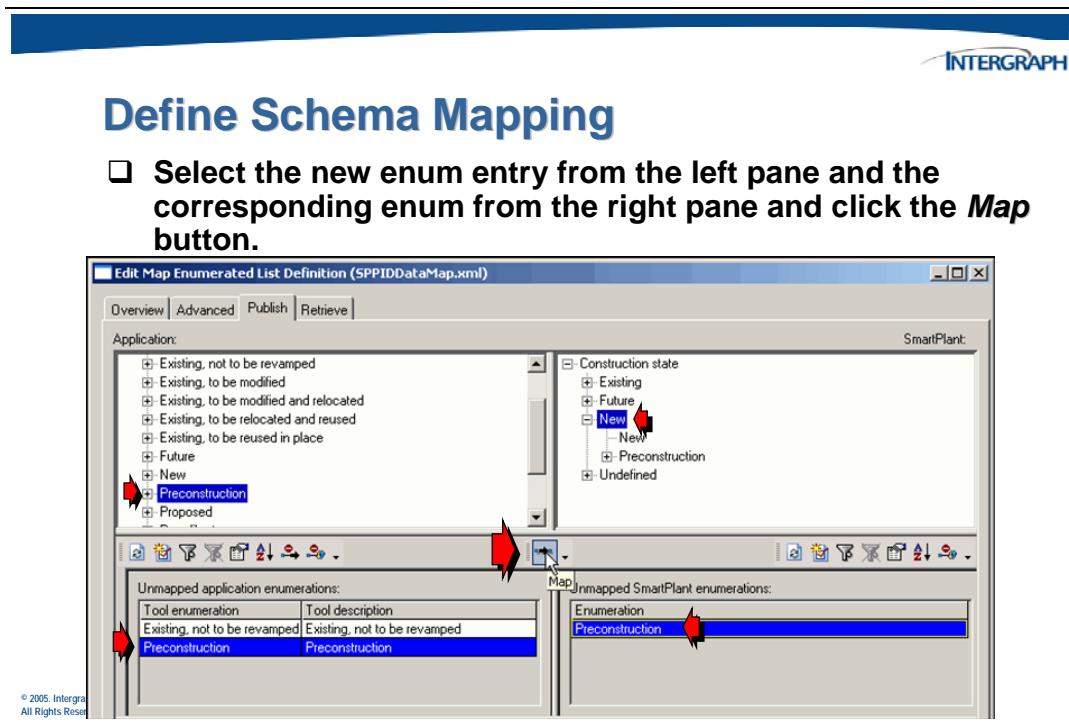
© 2005, Intergraph Corp.
All Rights Reserved.

Enter the necessary information to create a new child entry (Contained in **New**).



10.2.2 Mapping New Enum Entries for SPPID

After you define the new enumerated entry in the SmartPlant schema, you will do the mapping for publish and retrieve using the Map Environment.

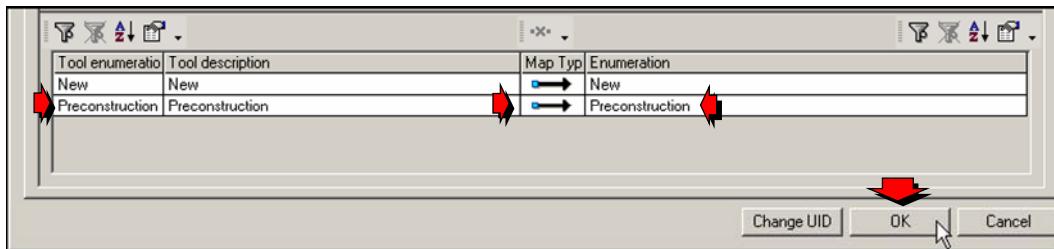


The **Map** button is used to map an unmapped enum value to an unmapped SmartPlant enum value.



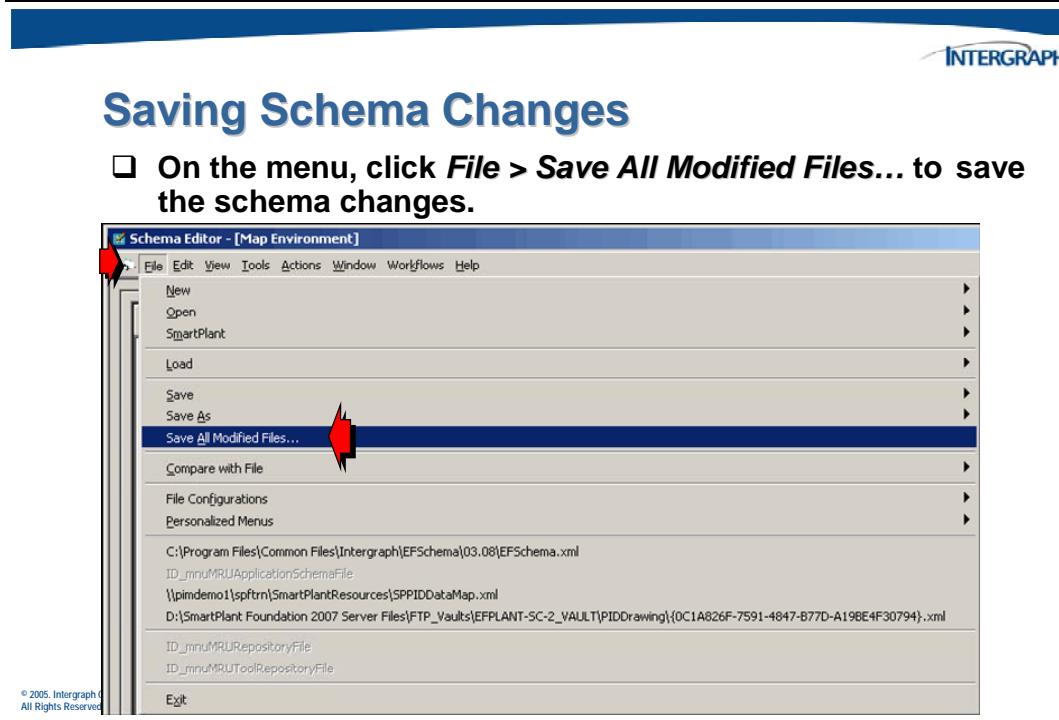
Define Schema Mapping

- ❑ Verify that the enum entries from the two schemas are mapped at the bottom of the dialog and click OK.



10.2.3 Saving SPPID Schema Changes

At this point, the additions to the SmartPlant schema and the mapping information is stored in memory will need to be saved to the respective xml files.



You will be prompted to save the changes to the tool map schema.



Saving Schema Changes

- Verify the tool map schema name for the mapping changes and choose **Yes**.



© 2005, Intergraph Corp.
All Rights Reserved.

You will also be prompted to save the changes to the SmartPlant extension schema.



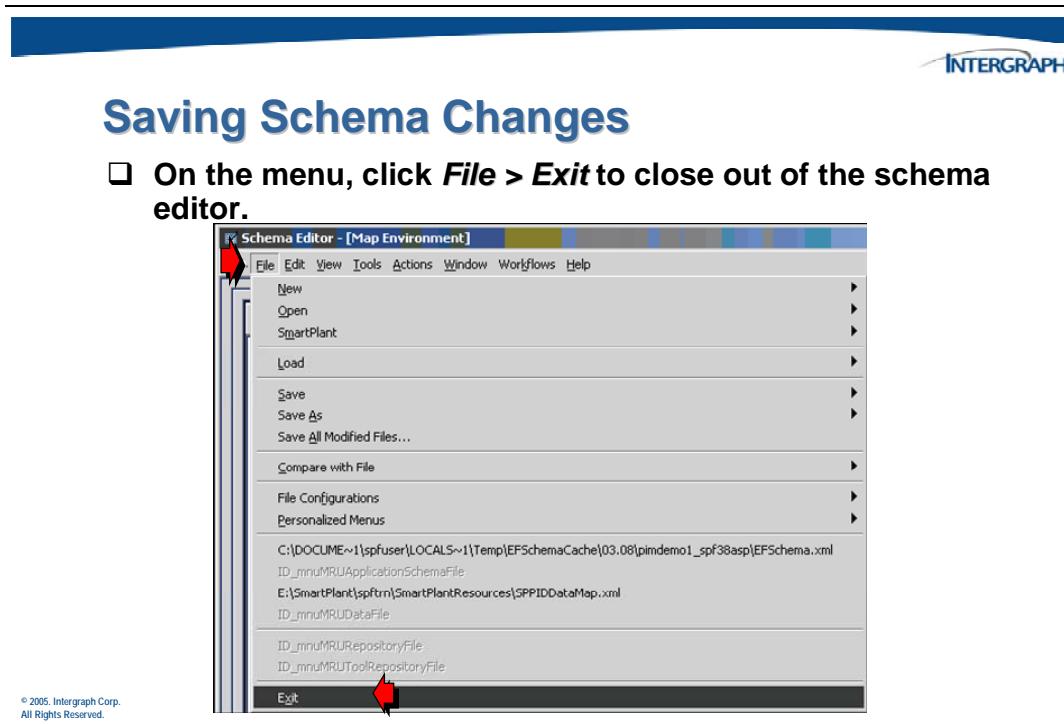
Saving Schema Changes

- Verify the extension schema name for the enum additions and choose **Yes**.

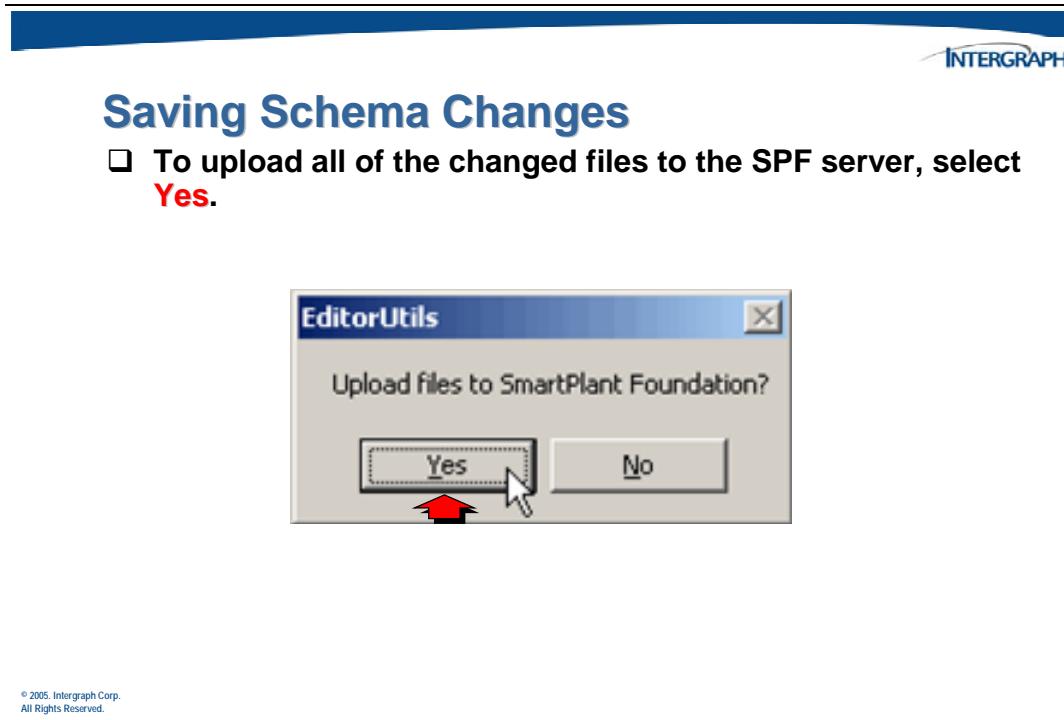


© 2005, Intergraph Corp.
All Rights Reserved.

Once all of the schema files have been saved exit the Schema Editor.



The last step is to upload these files to the schema location on the SmartPlant Foundation server.



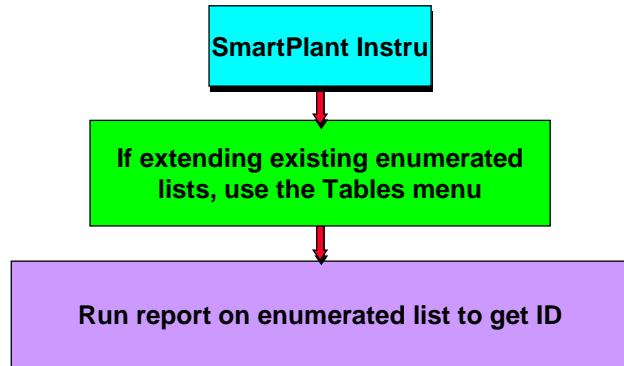
10.3 Extending an Enumerated List in SmartPlant Instrumentation

The next step to extending the system is to add the same information in the next authoring tool such as SmartPlant Instrumentation.

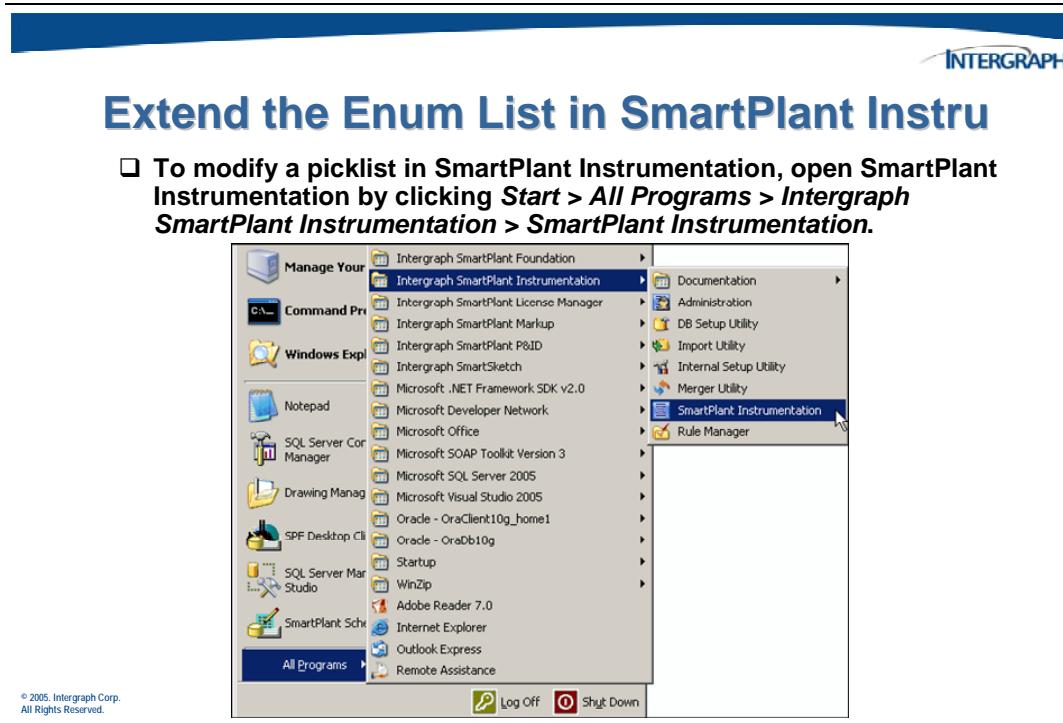


Extend the Enum List in SmartPlant Instru

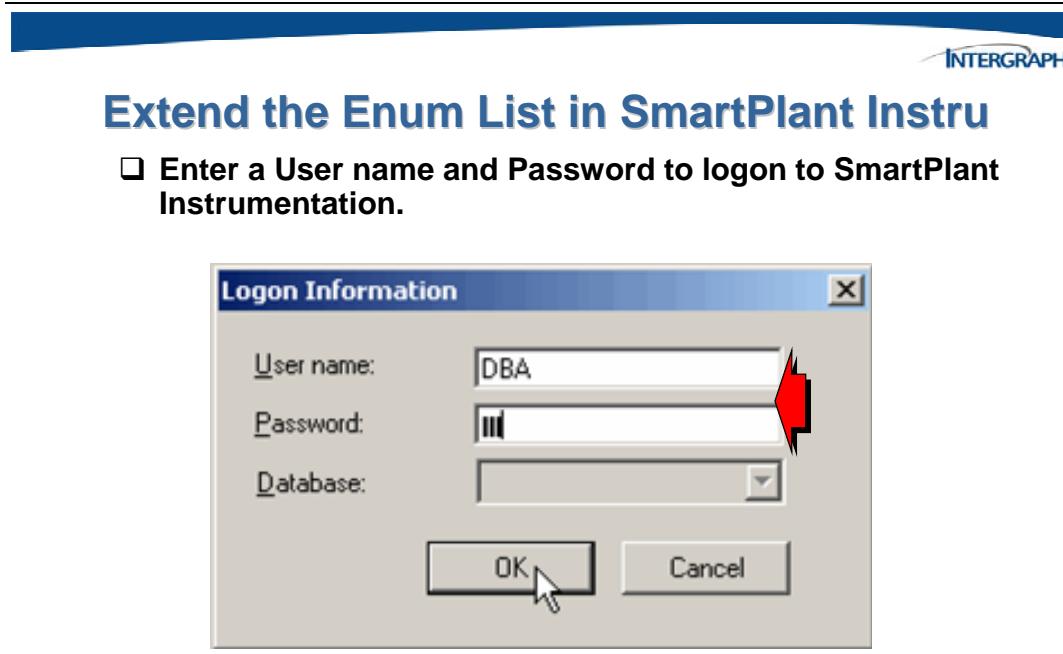
The following workflow describes the steps necessary to extend the SmartPlant Instrumentation authoring tool.



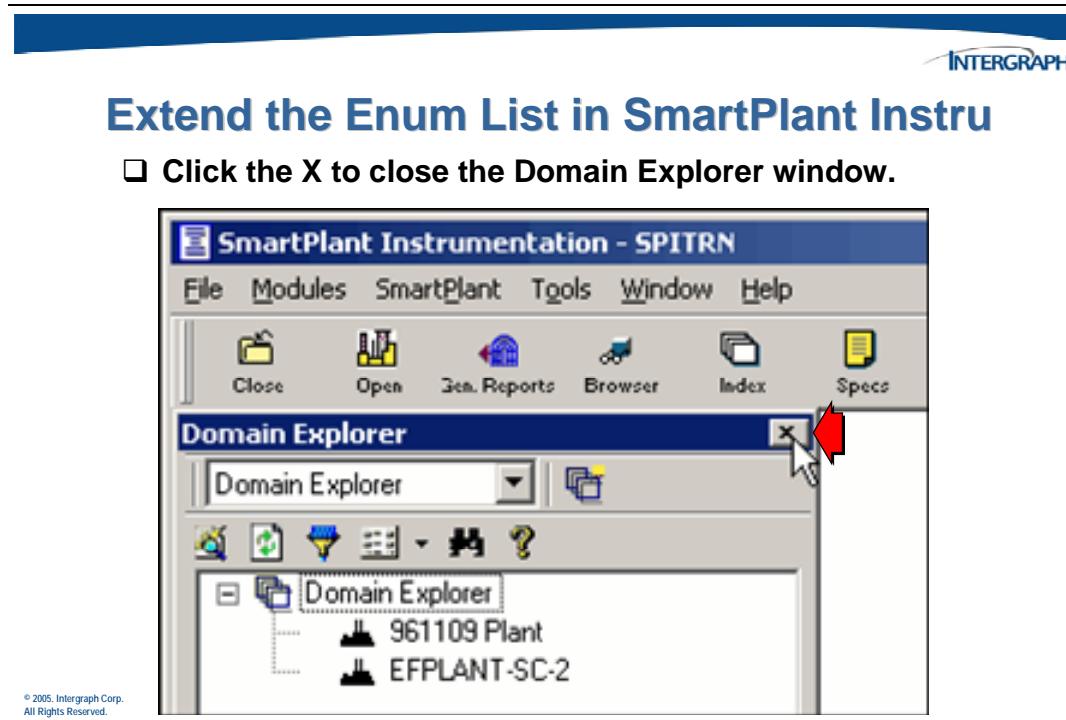
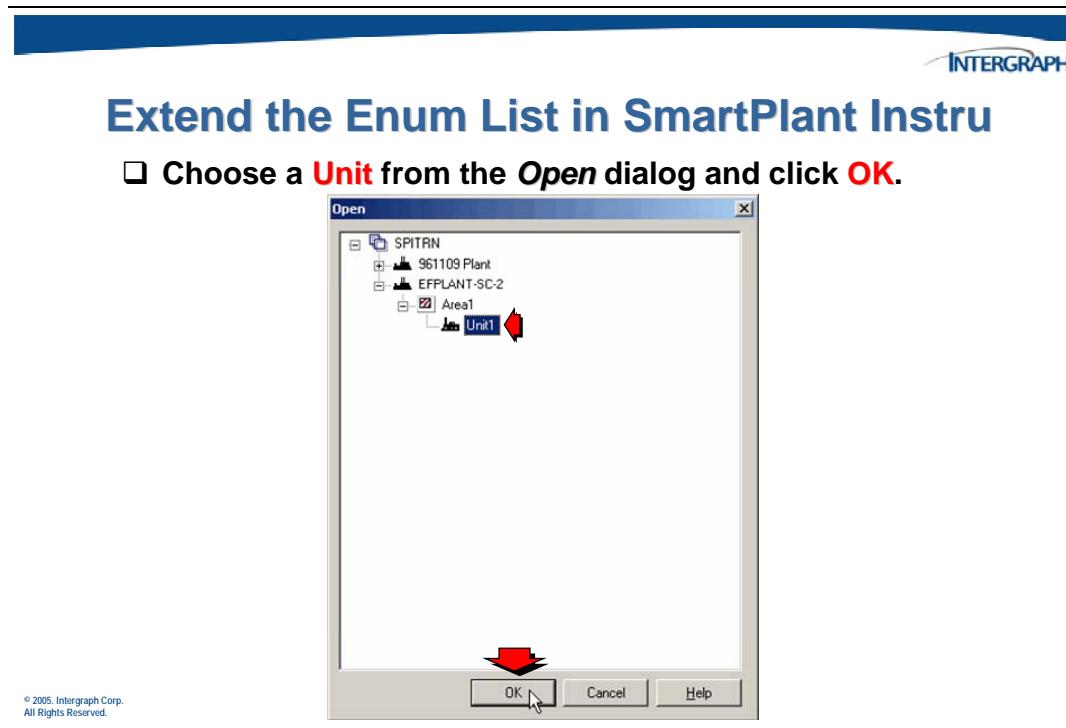
Start SmartPlant Instrumentation in order to add an extension to an existing picklist.



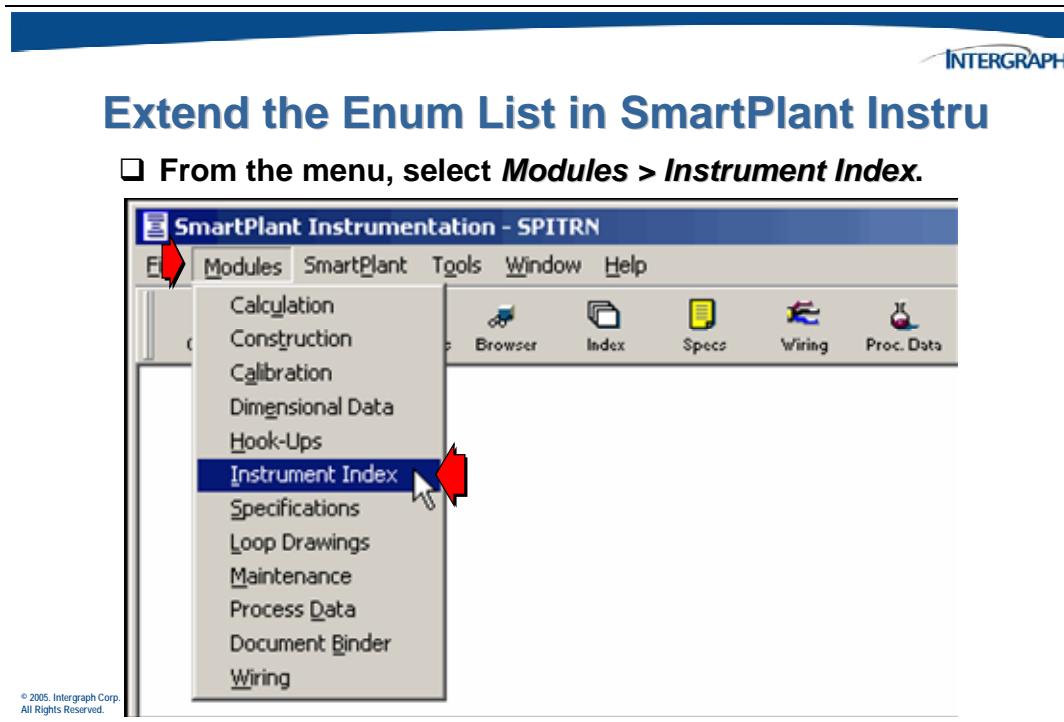
The login dialog will display.



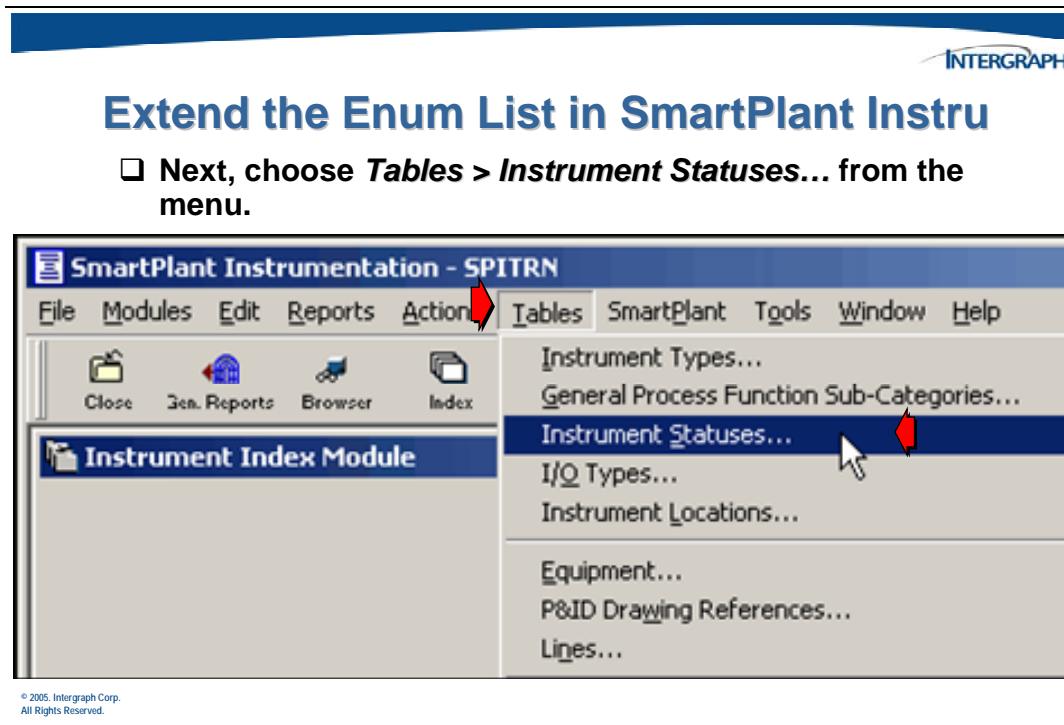
When the *Open* window displays, choose a plant/unit from the displayed tree.



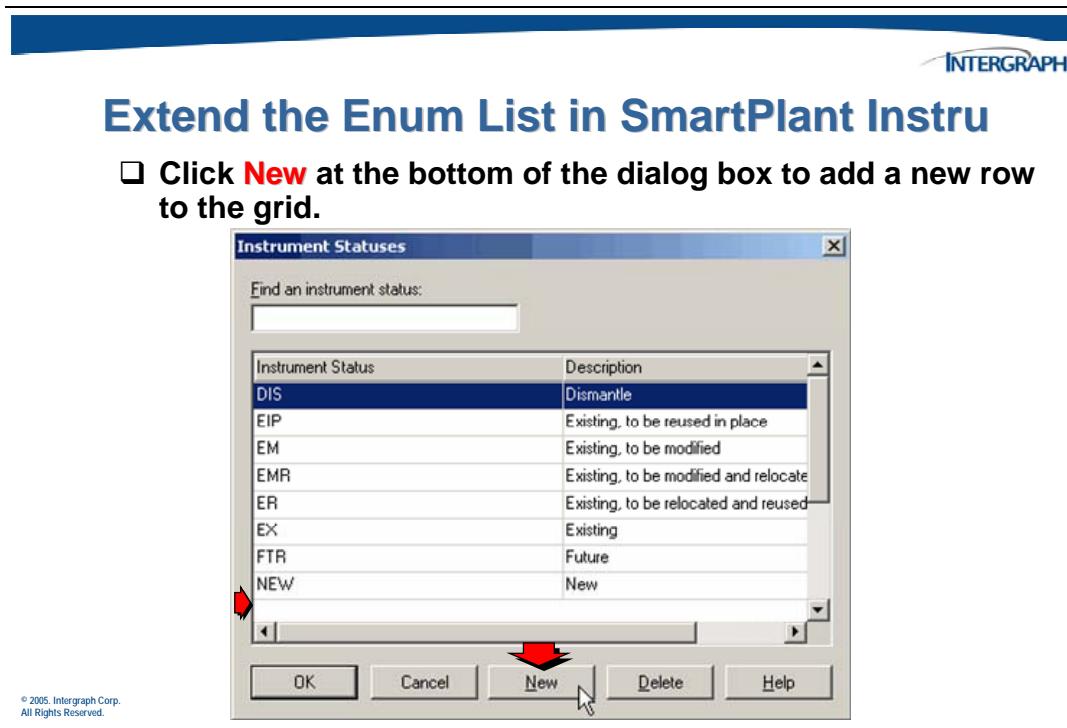
From the SmartPlant Instrumentation application window, open the *Instrument Index* module.



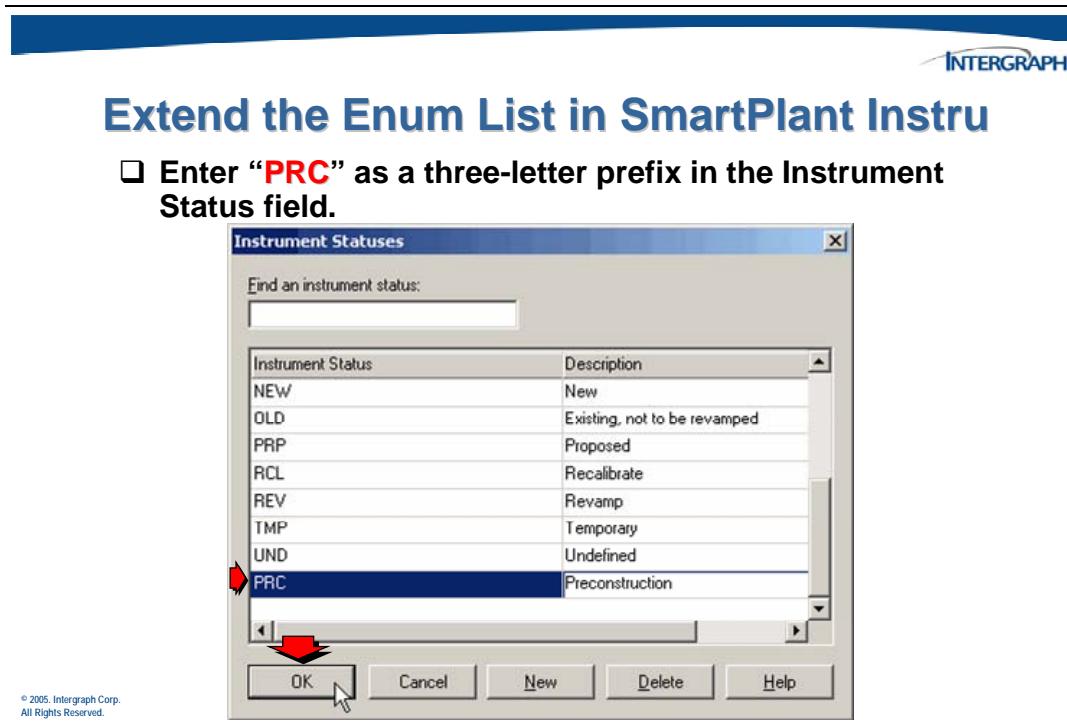
The *Instrument Index Module* window will display.



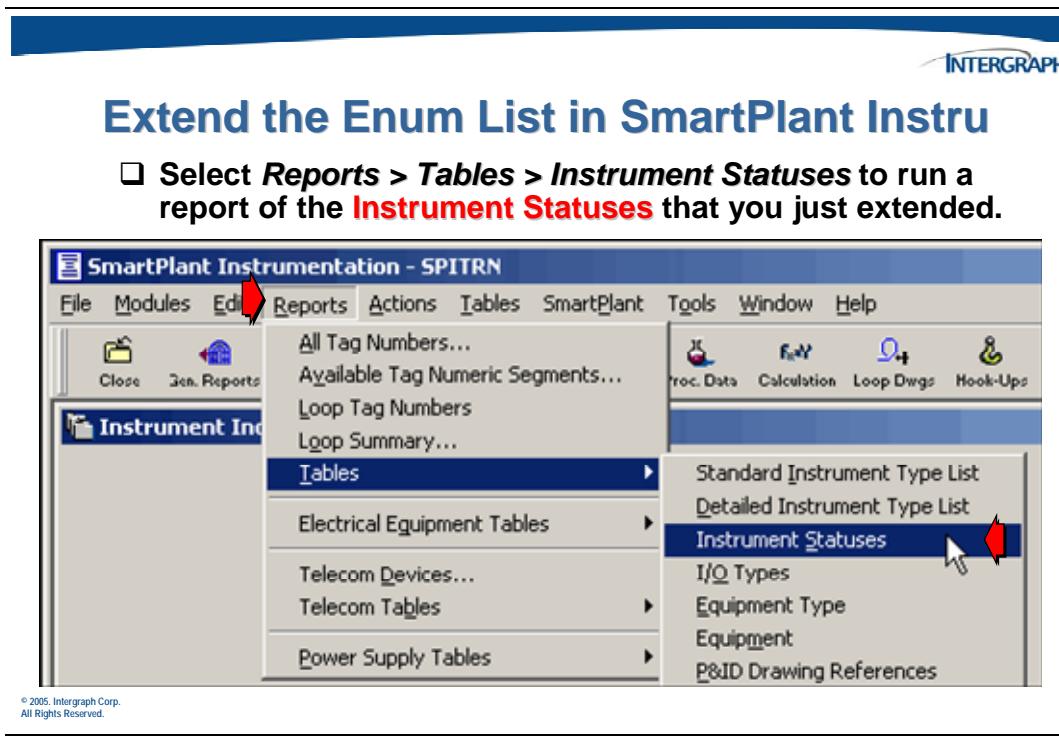
The *Instrument Statuses* dialog will display.



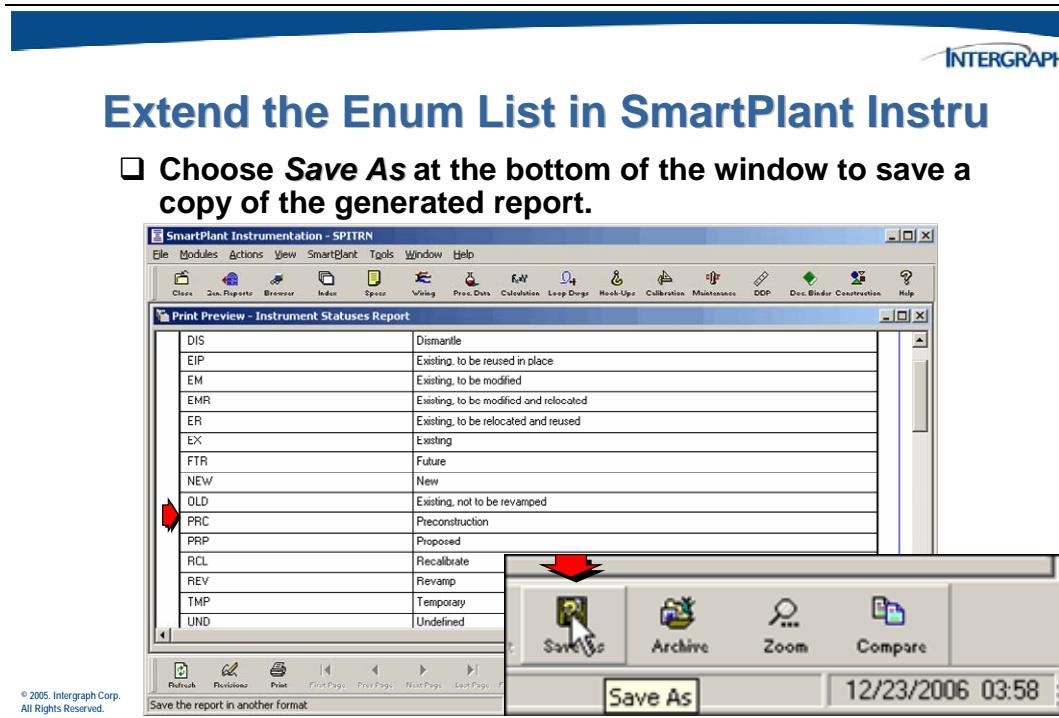
Enter the new picklist information in the new row at the bottom of the dialog.



To see the new *Instrument Status* that has been added, generate an **Instrument Statuses** report.



This displays the print preview of the report.

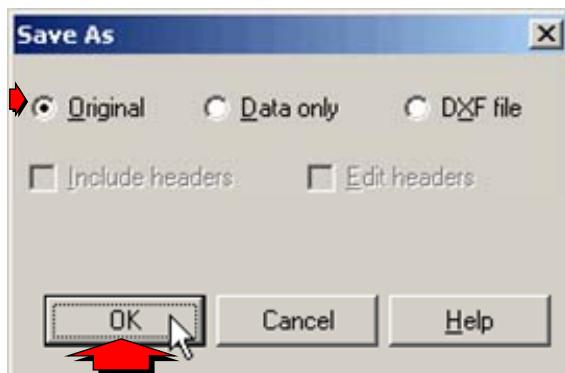


A Save As dialog will display.



Extend the Enum List in SmartPlant Instru

- ❑ Select the *Original* option, and click **OK**.



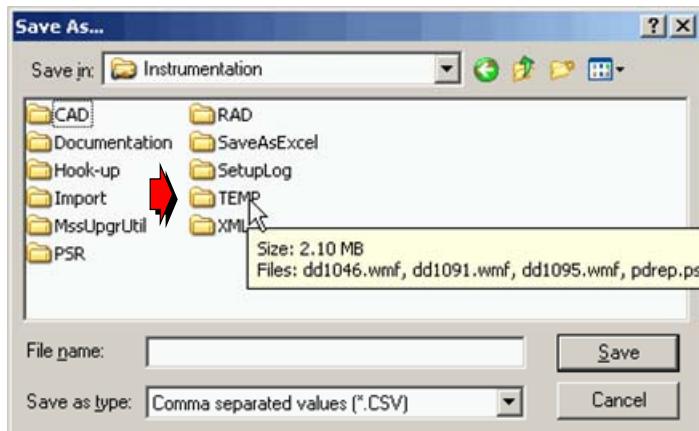
© 2005, Intergraph Corp.
All Rights Reserved.

Select a convenient location such as the TEMP folder in order to save the report file,



Extend the Enum List in SmartPlant Instru

- ❑ Double-click on the TEMP sub-folder as the location for the saved report.

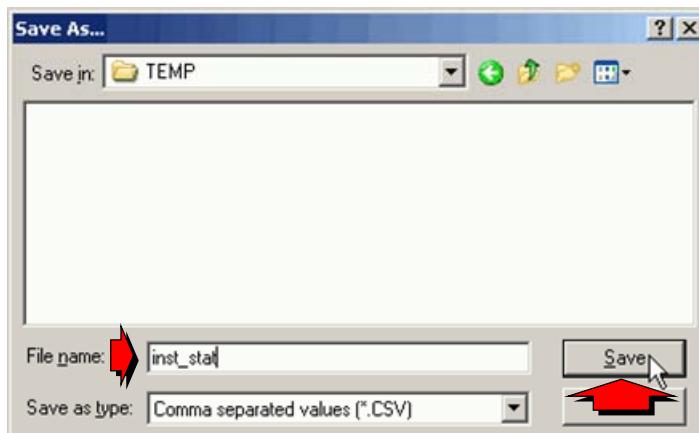


© 2005, Intergraph Corp.
All Rights Reserved.

Key in a name for the report (it will be saved with a .csv extension). The software saves the report in a comma-delimited format.

Extend the Enum List in SmartPlant Instru

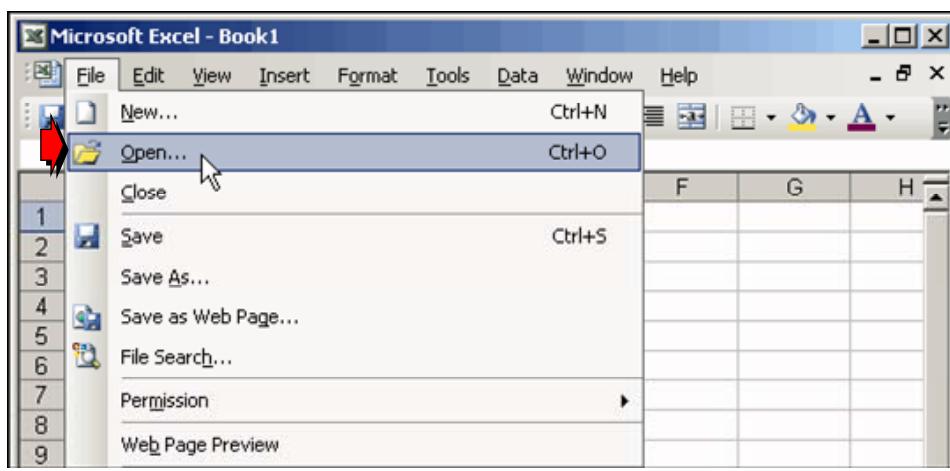
- Name the report **inst_stat.CSV**, and save it to the TEMP folder.



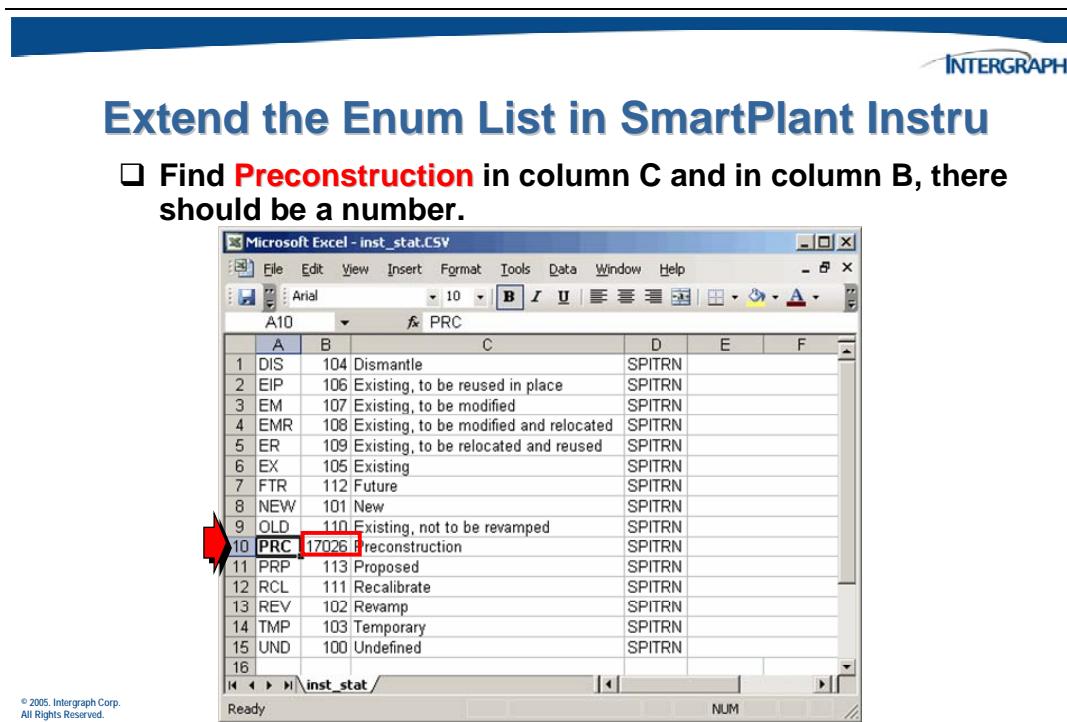
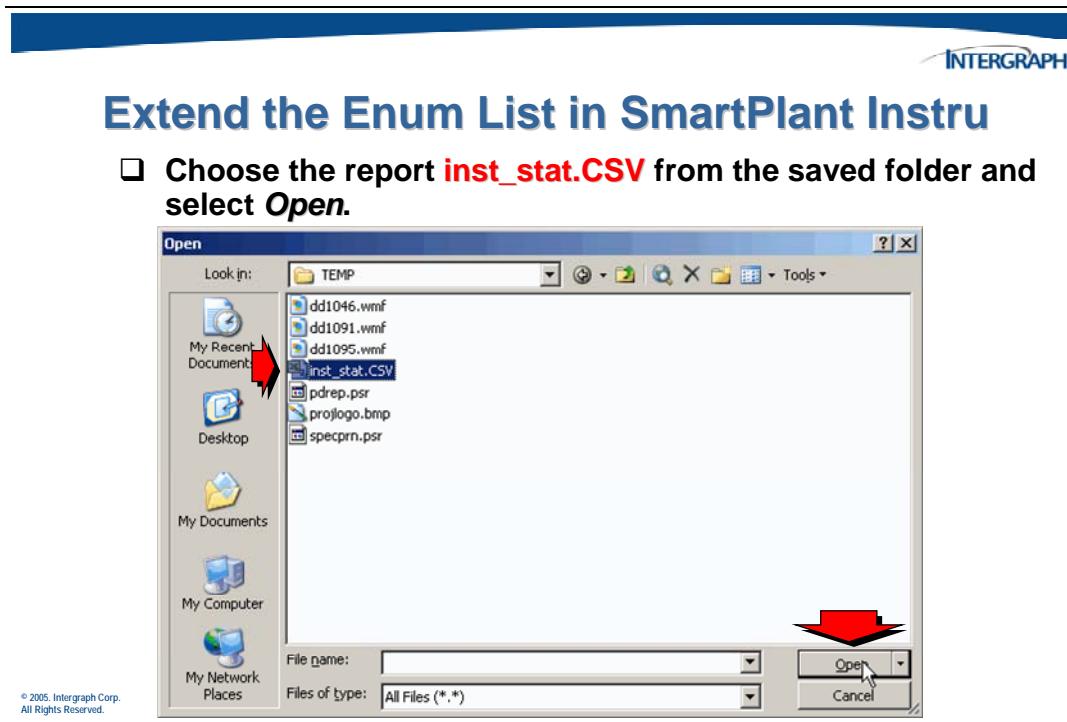
Start Microsoft Excel and view the generated report file.

Extend the Enum List in SmartPlant Instru

- Open the saved **inst_stat.CSV** report with Microsoft Excel.



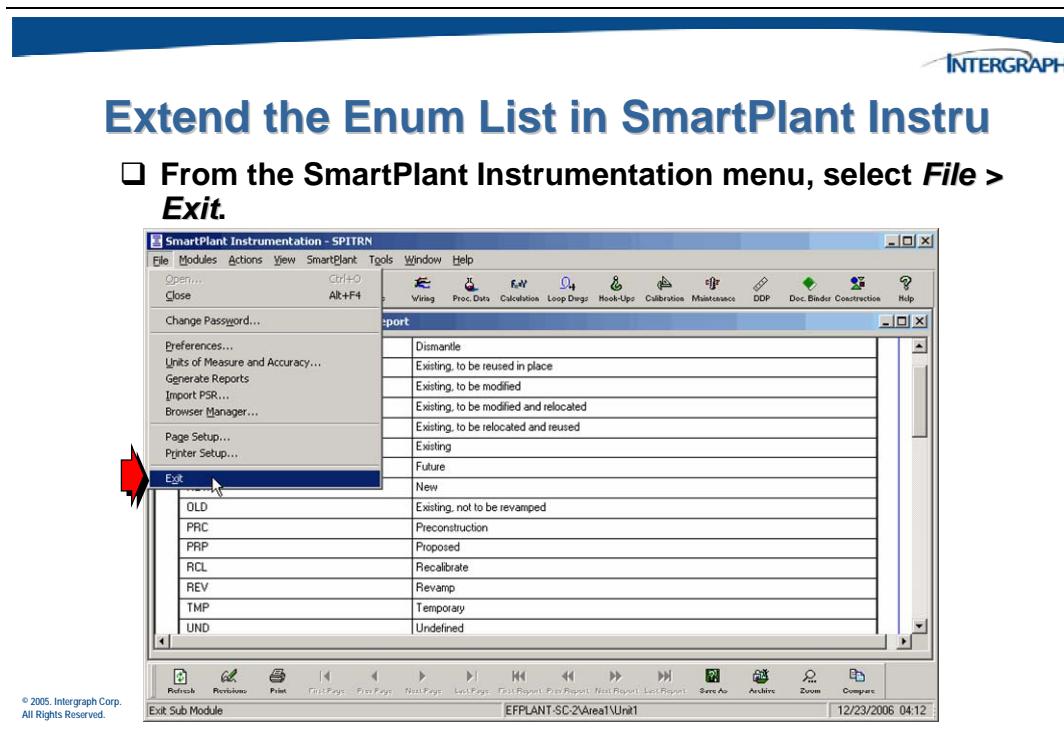
An *Open* dialog will be displayed.



Find the value **PRC** in column A. In column B, there should be a number (in this case, it is 17026). And in column C, it should display **Preconstruction** for the value.

This information will be used for extending the SmartPlant Instrumentation Tool Schema map file. The value **17026** equals the *Name* and **Preconstruction** equals *Description* for the enumerated list extension. Make a note of this information or refer back to this report. A suggestion would be to print this report and have it available when adding the enumerated list extension in the Schema Editor. This value can be useful in troubleshooting possible mapping problems.

Exit out of Microsoft Excel and also from SmartPlant Instrumentation.

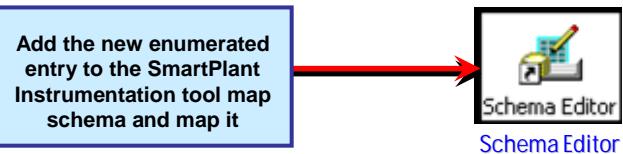


10.3.1 Extending the SPI Tool Map Schema

Again, before you can define mapping for these new entries in the tool map schema, the enumerated list must first be extended. This is done automatically when the tool map schema and the SmartPlant schema are synchronized.



Extend the SmartPlant Schema



A list of available tool map schemas will be displayed. This time, select the tool map schema for SmartPlant Instrumentation (INtools).



Map the Enum List in the Tool Schema

- Select the tool map schema to be loaded and click Finish.

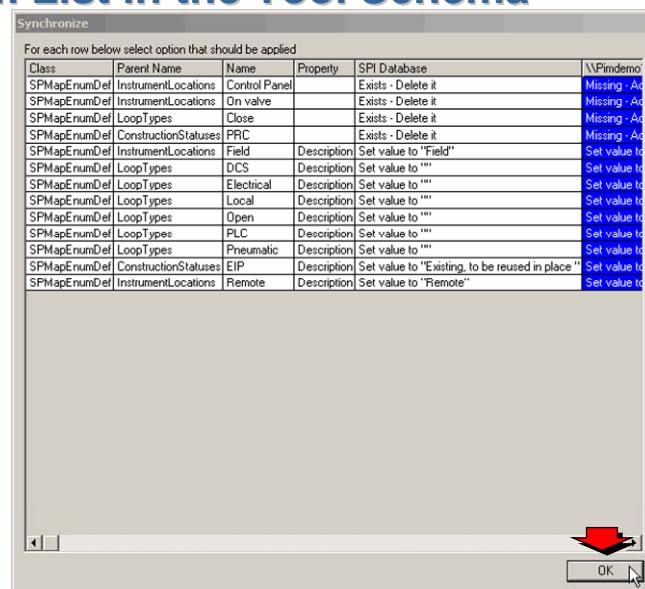


The differences between the two map schemas are displayed in the **Synchronize** form.



Map the Enum List in the Tool Schema

- Click OK from the Synchronize dialog to have the schema editor read the tool meta-schema and automatically add the new enum extension to the tool map schema.



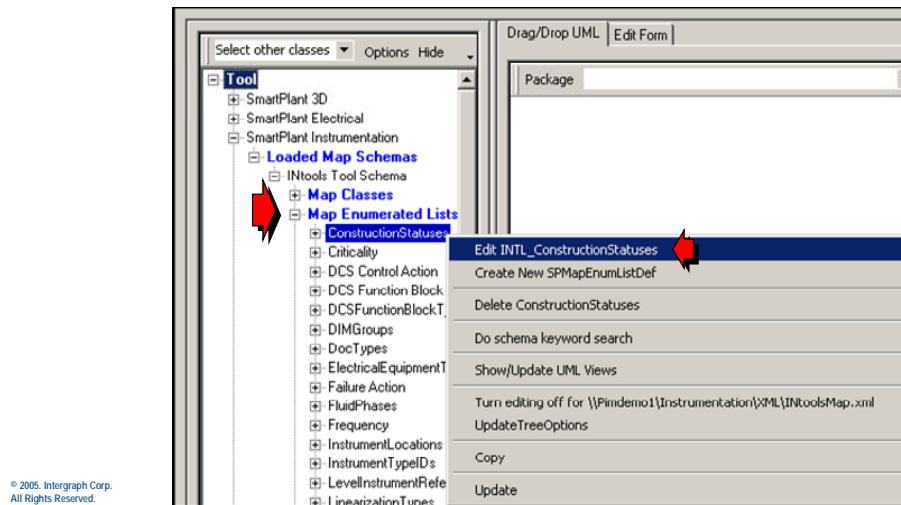
10.3.2 Mapping New Enum Entries for SPI

The enum entry “PRC” is automatically added to the tool map schema. Because the value for Preconstruction has already been added to the extension schema (ToolExten.xml) all that remains to be done is the mapping.



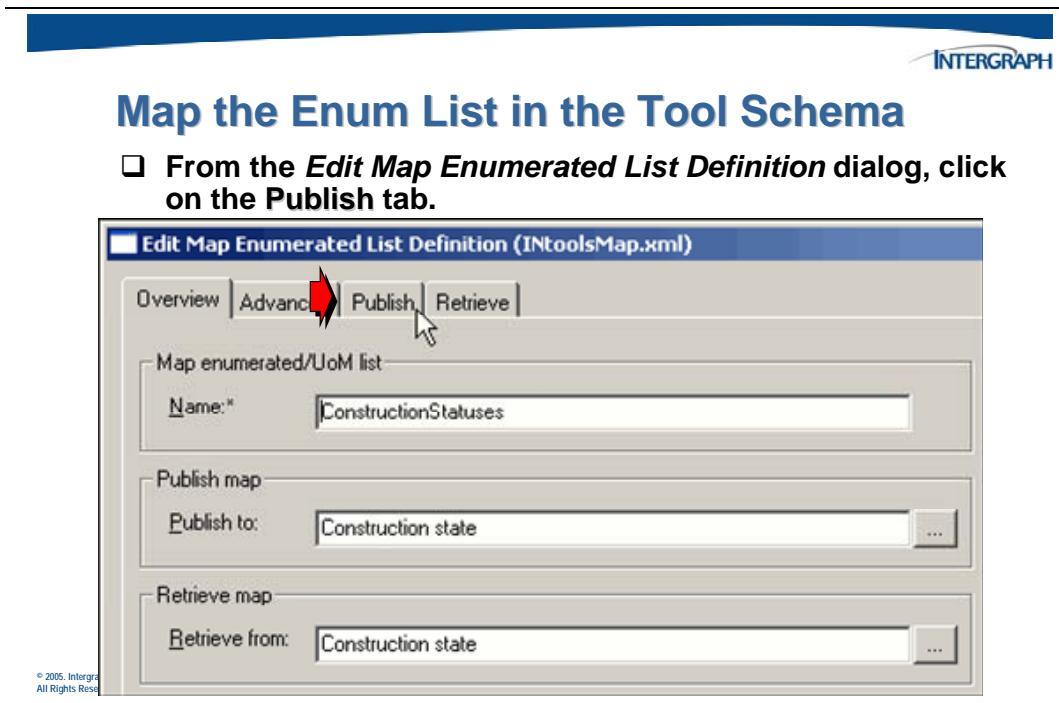
Map the Enum List in the Tool Schema

- Choose the **ConstructionStatuses Map Enumerated List** and select **Edit INTL_ConstructionStatuses** from the menu.

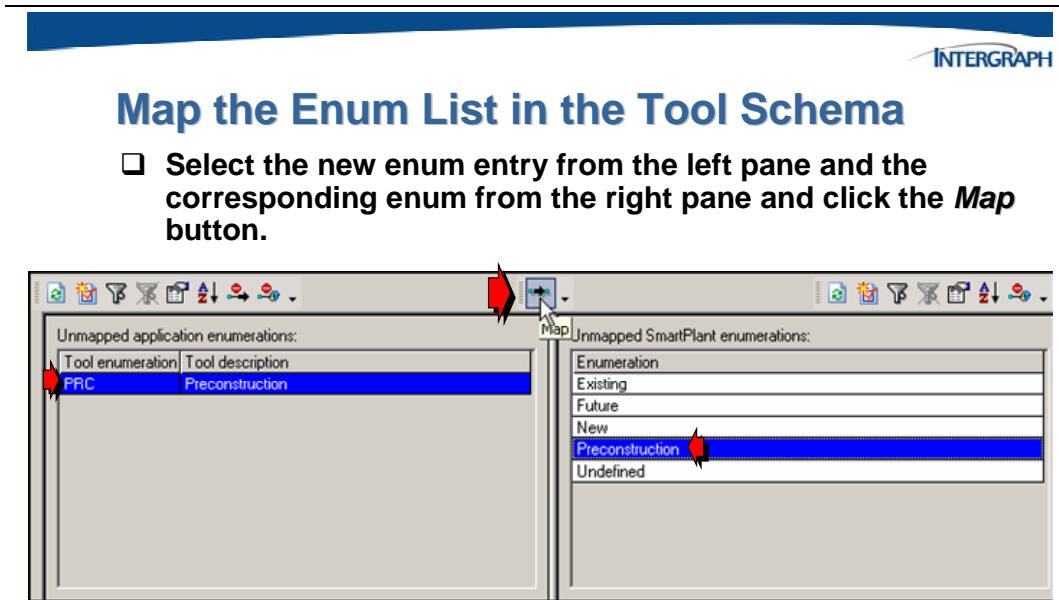


© 2005, Intergraph Corp.
All Rights Reserved.

The *Edit Map Enumerated List Definition* dialog will display.



After you define the new enumerated entry in the SmartPlant schema, you will do the mapping for publish and retrieve using the Map Environment.



The **Map** button is used to map an unmapped enum value to an unmapped SmartPlant enum value.



Define Schema Mapping

- Verify that the enum entries from the two schemas are mapped at the bottom of the dialog.

The screenshot shows a table with four columns: Tool enumeration, Tool description, Map Typ, and Description. Red arrows point to the bottom row of the table, which contains the entries 'PRC' and 'Preconstruction' under the 'Tool enumeration' and 'Tool description' columns respectively, and 'Preconstruction' under both 'Map Typ' and 'Description'.

Tool enumeration	Tool description	Map Typ	Description
DIS	Dismantle	→	Dismantle
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 relocated	→	Existing, to be modified and relocated
ER	Existing, to be relocated and reused	→	Existing, to be relocated and reused
EX	Existing	→	Existing
FTR	Future	→	Future
NEW	New	→	New
OLD	Existing, not to be revamped	→	Existing, not to be revamped
PRP	Proposed	→	Proposed
RCL	Recalibrate	→	Recalibrate
REV	Revamp	→	Revamp
TMP	Temporary	→	Temporary
UND	Undefined	→	Undefined
PRC	Preconstruction	→	Preconstruction

© 2005, Intergraph Corp.
All Rights Reserved.

Next, perform the mapping needed to retrieve the *Construction Status* information from SPF.



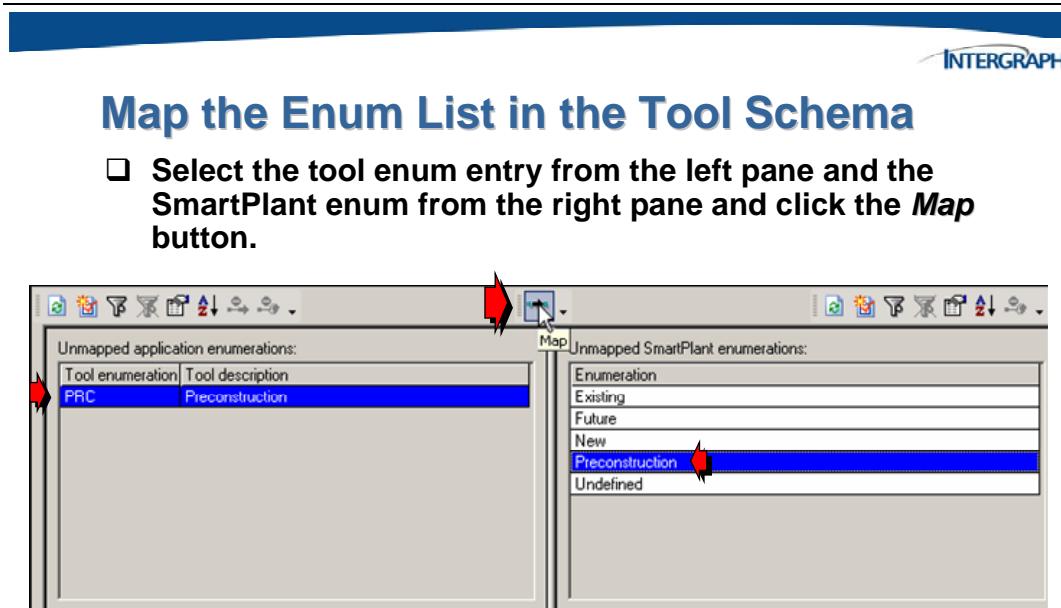
Map the Enum List in the Tool Schema

- At the top of the dialog, click on the Retrieve tab to perform the retrieval mapping.

The screenshot shows the 'Edit Map Enumerated List Definition (IToolsMap.xml)' dialog. The tabs at the top are Overview, Advanced, Put, and Retrieve, with 'Retrieve' being the active tab. Red arrows point to the 'Retrieve' tab and the 'Construction state' node in the tree view on the right. The left pane shows a tree structure of 'ConstructionStatuses' with various status codes like DIS, EIP, EM, etc. The right pane shows a detailed view of the 'Construction state' node, which includes 'Existing', 'Future', 'New', 'Preconstruction', and 'Undefined'.

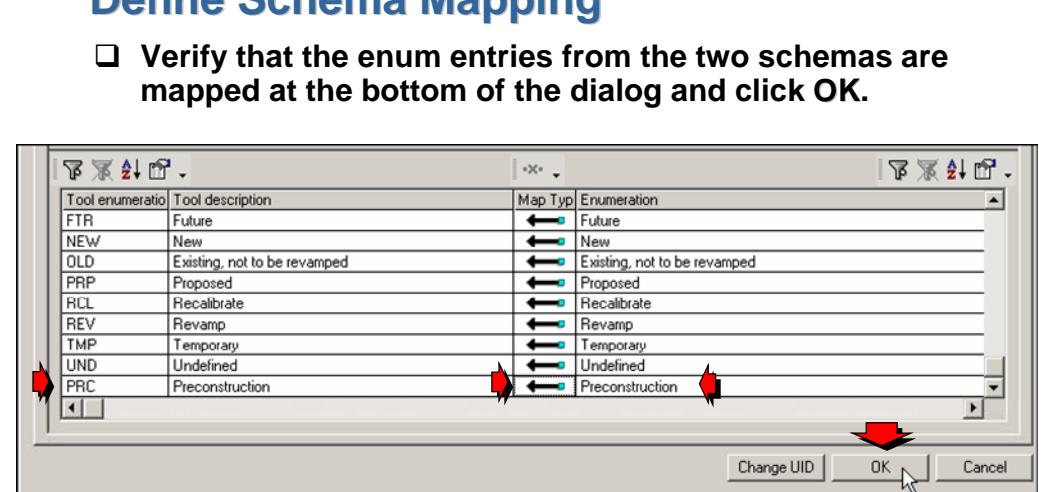
© 2005, Intergraph Corp.
All Rights Reserved.

The **Map** button is used to map an unmapped tool enum value to an unmapped SmartPlant enum value.



© 2005, Intergraph Corp.
All Rights Reserved.

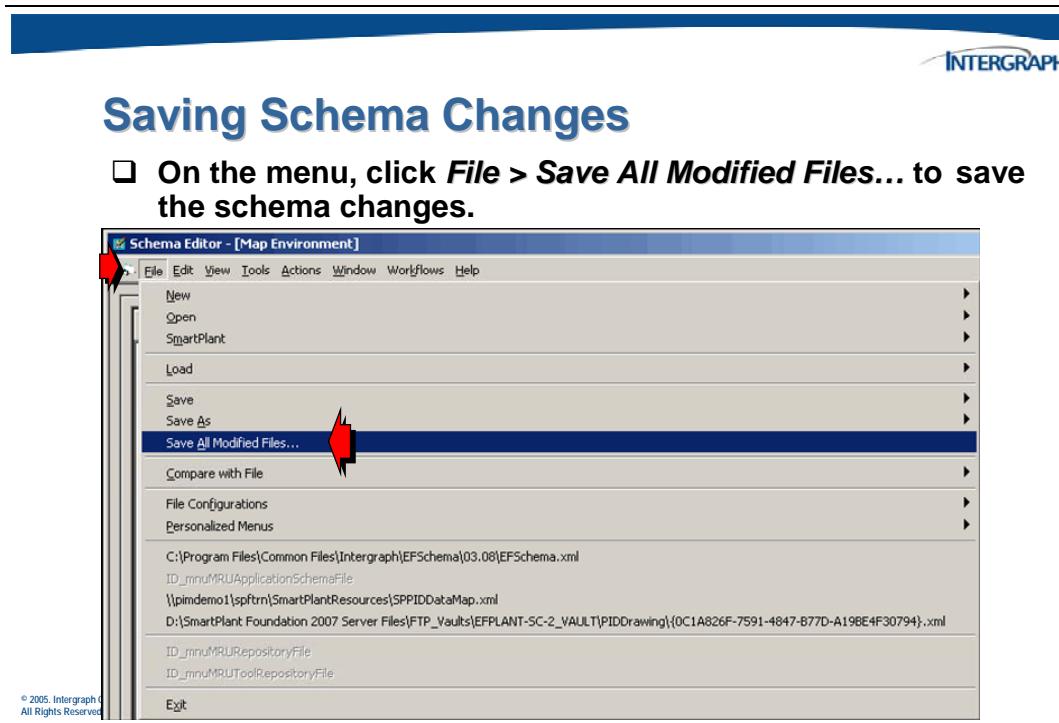
This completes the mapping for the new enumerated entry for both Publish and Retrieve operations.



© 2005, Intergraph Corp.
All Rights Reserved.

10.3.3 Saving SPI Schema Changes

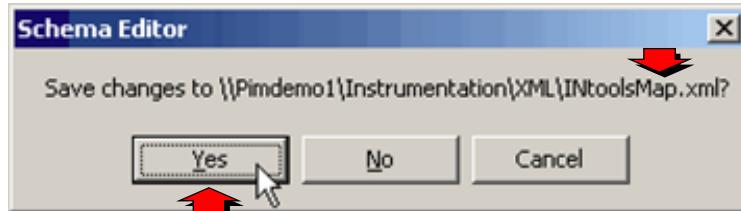
All of the mapping information is stored in memory will need to be saved to the respective xml files.



You will be prompted to save the changes to the tool map schema.

Saving Schema Changes

- Verify the tool map schema name for the mapping changes and choose **Yes**.



© 2005, Intergraph Corp.
All Rights Reserved.

Once all of the schema files have been saved exit the Schema Editor.

Saving Schema Changes

- On the menu, click **File > Exit** to close out of the schema editor.

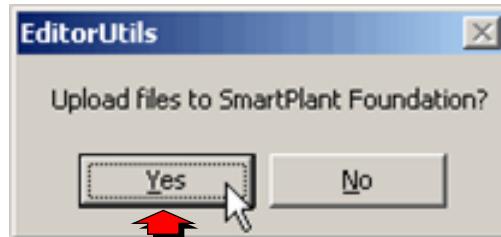
© 2005, Intergraph Corp.
All Rights Reserved.

Finally, upload these files to the schema location on the SmartPlant Foundation server.



Saving Schema Changes

- To upload all of the changed files to the SPF server, select **Yes**.



10.4 Modifying View Definitions

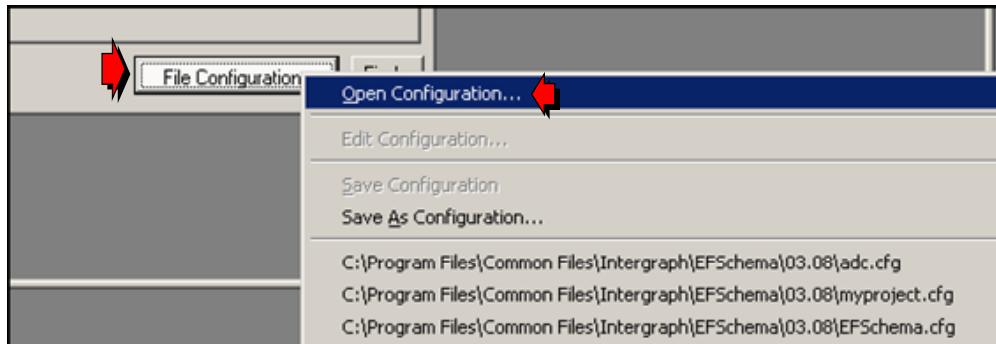
Depending on the property that you want to add to the SmartPlant Schema, you may have to extend the *view definitions*, *graph definitions* and *edge definitions*. When an enumerated list is being extended to add an entry, nothing additional has to be done to the Edge Definition if the property is already in the View Definition. In the case for construction status enumeration, a new entry, “Preconstruction”, was added to **ConstructionStatus2** or the second level of the construction status enumerations. The delivered View Definitions only have the first level of construction status enumeration in the definition. If you want to see **Preconstruction** when you select a piece of equipment, you will have to add *Construction Status2* to the view definitions which is the second level of the Construction Status Hierachal Enumeration.

Since the delivered View Definitions are contained in the extension file **EdgeGraphView.xml**, the configuration must be modified to allow changes.



Modifying View Defs

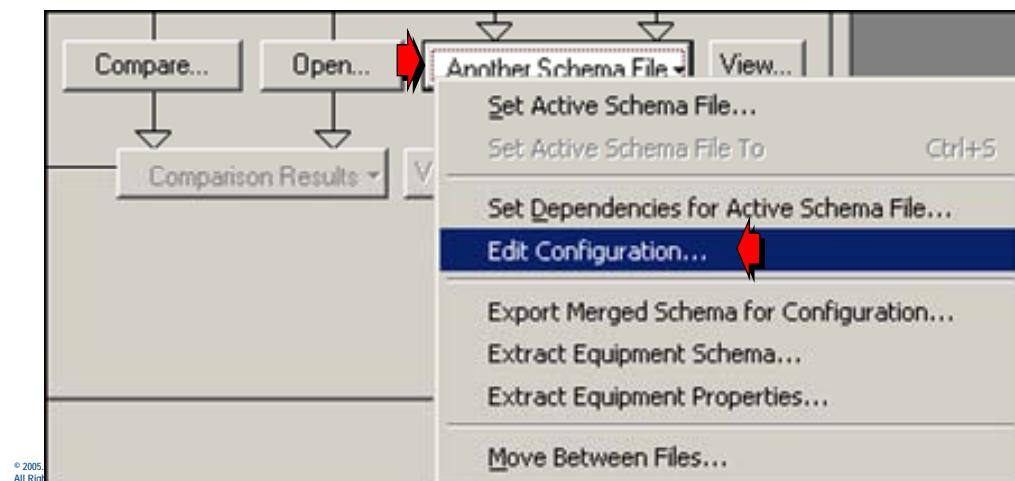
- From the Workflows dialog box, choose the File Configurations button and select **Open Configuration..**





Modifying View Defs

- Click on the bolded Another Schema File button, then select **Edit Configuration...**

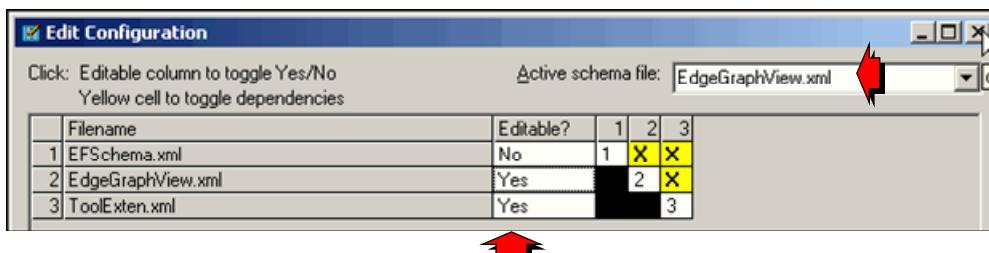


Make the *Active schema file* **EdgeGraphView.xml** and also change the *Editable?* field to **Yes** to allow changes to this extension file.



Modifying View Defs

- Click in the **Editable?** Field to set the permissions on the schema files in this configuration.

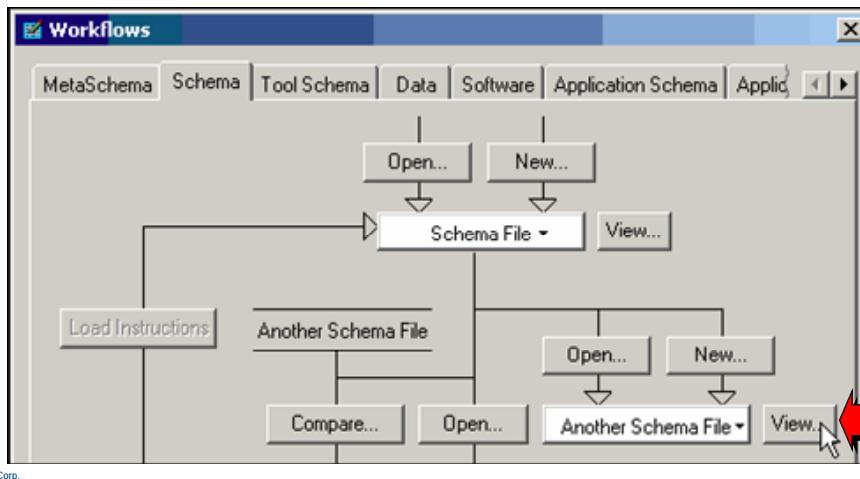


Now that the configuration has been changed, make the modifications to the view definitions.



Modifying View Defs

- In the **Workflows** dialog box, click the **View** button to view the navigation schema file (**EdgeGraphView.xml**).

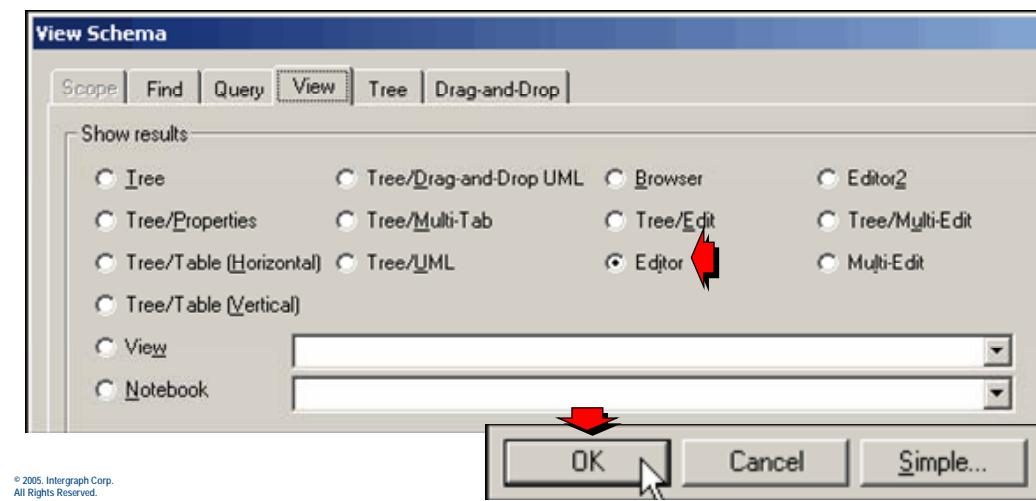


Use the *Editor* view to make the necessary changes.

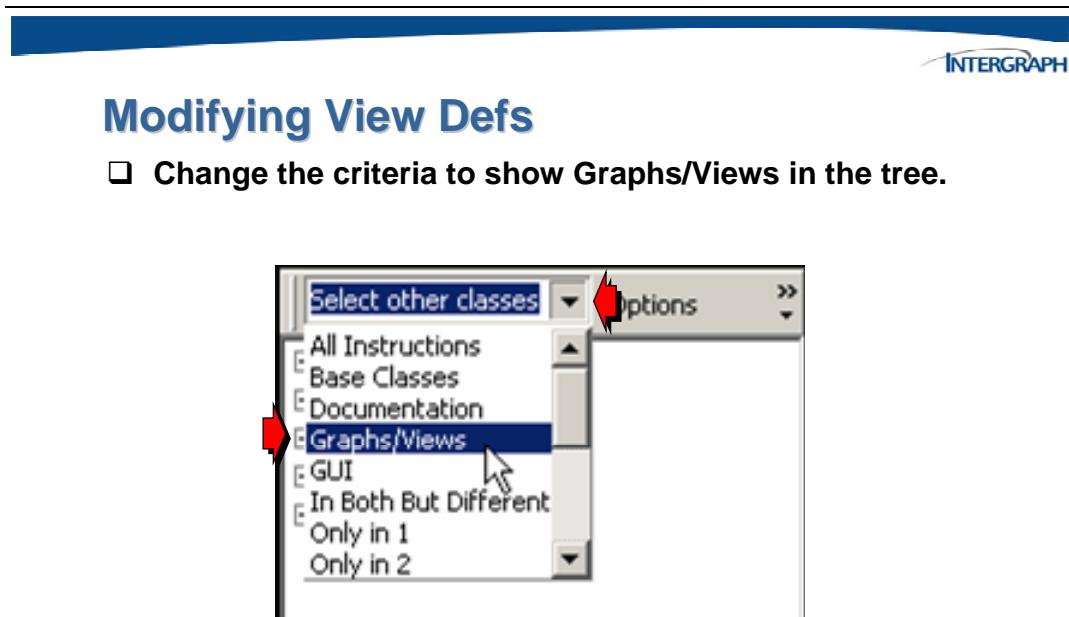


Modifying View Defs

- In the **View Schema** dialog box, select the **Editor** view and click **OK**.

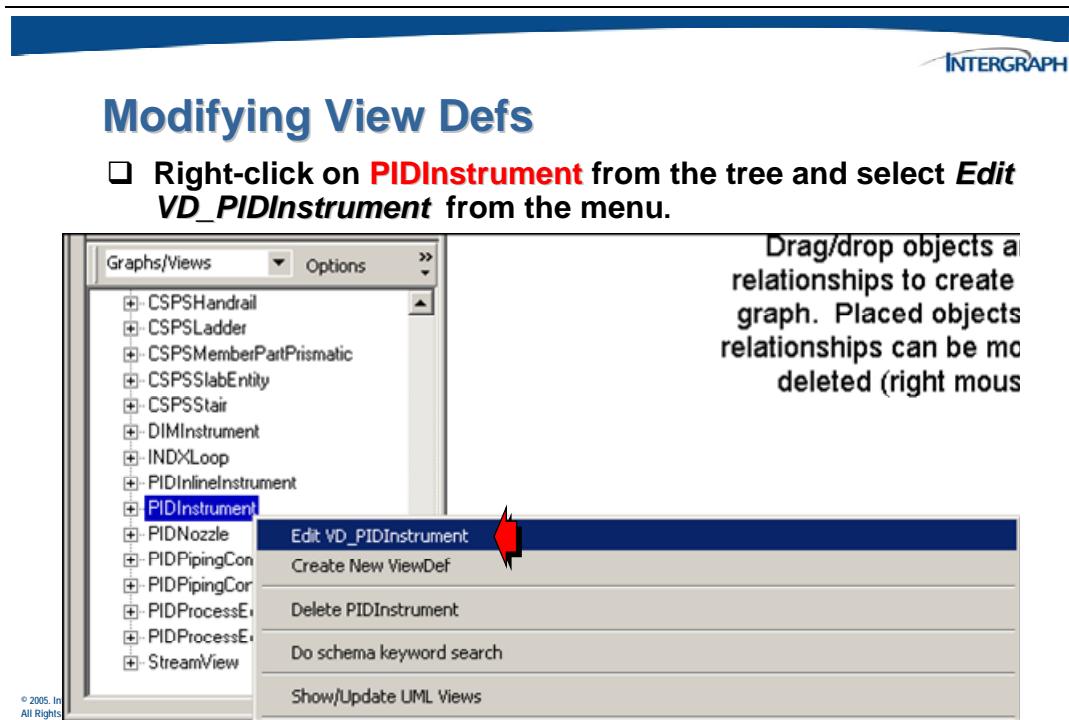


Change the display from the Base Classes to be able to see the **Graphs/Views**.

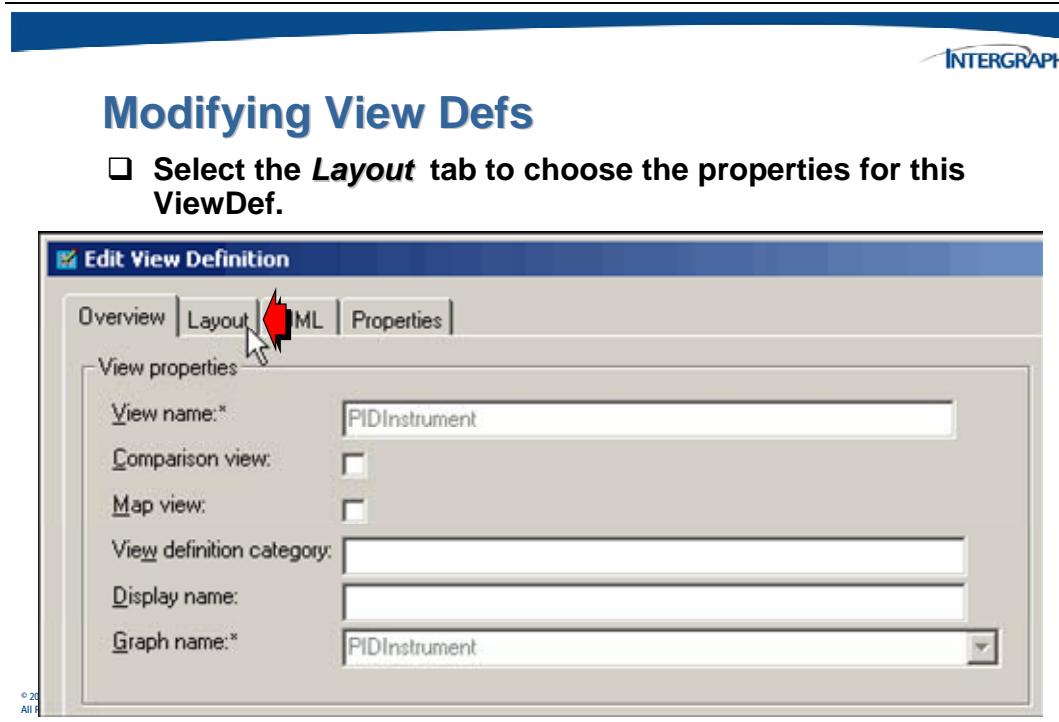


© 2005. Intergraph Corp.
All Rights Reserved.

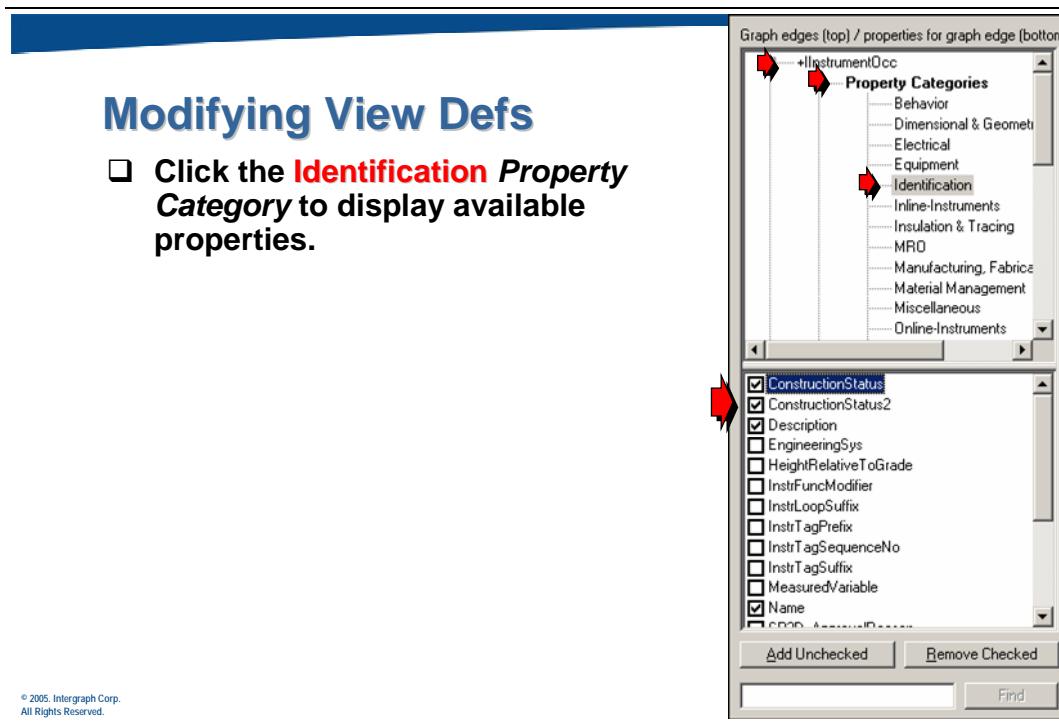
Expand the tree to display the **View Definition** objects. Locate and highlight the view definition to be changed.



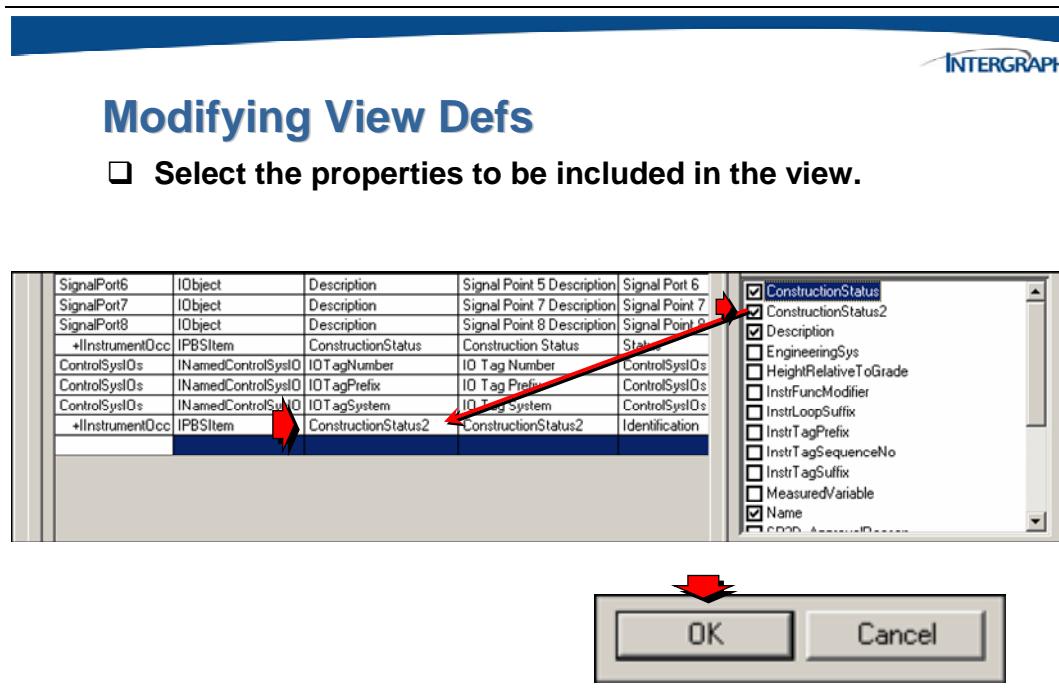
Since this is an existing View Definition, switch to the Layout screen to add additional properties.



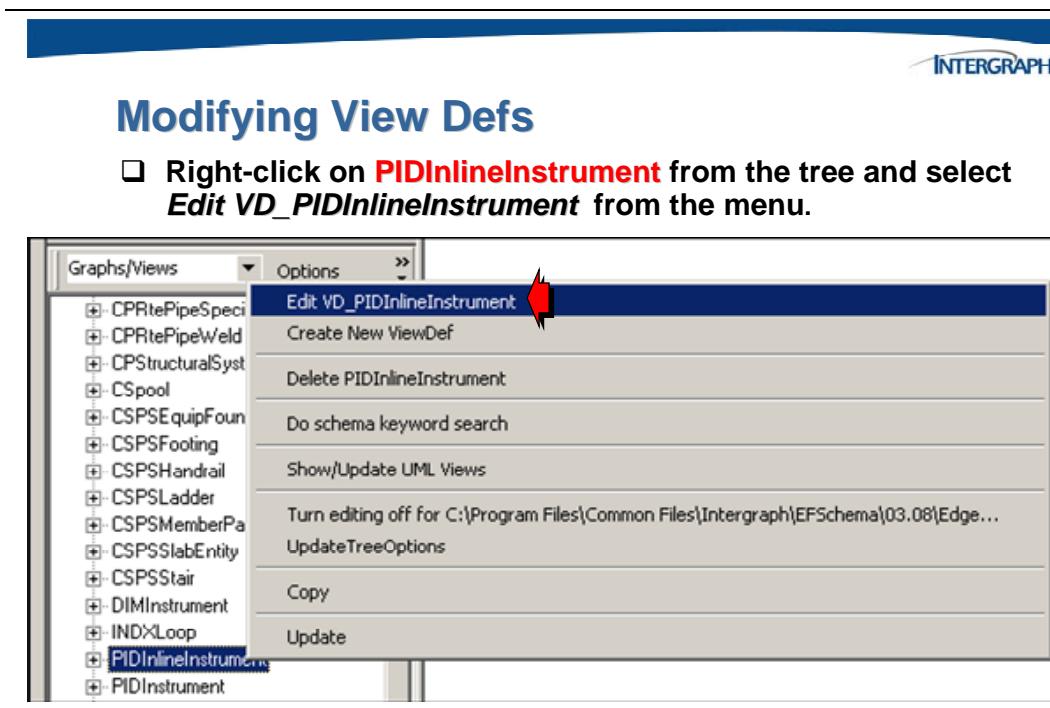
Expand the **IInstrumentOcc** interface and locate **ConstructionStatus2** in the list.



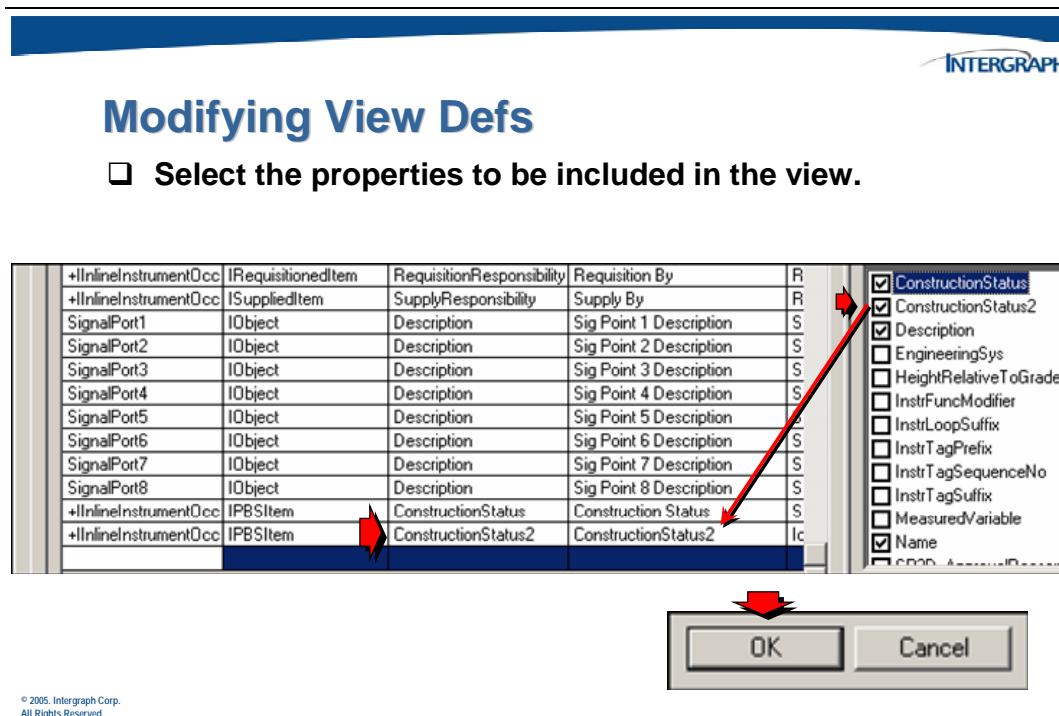
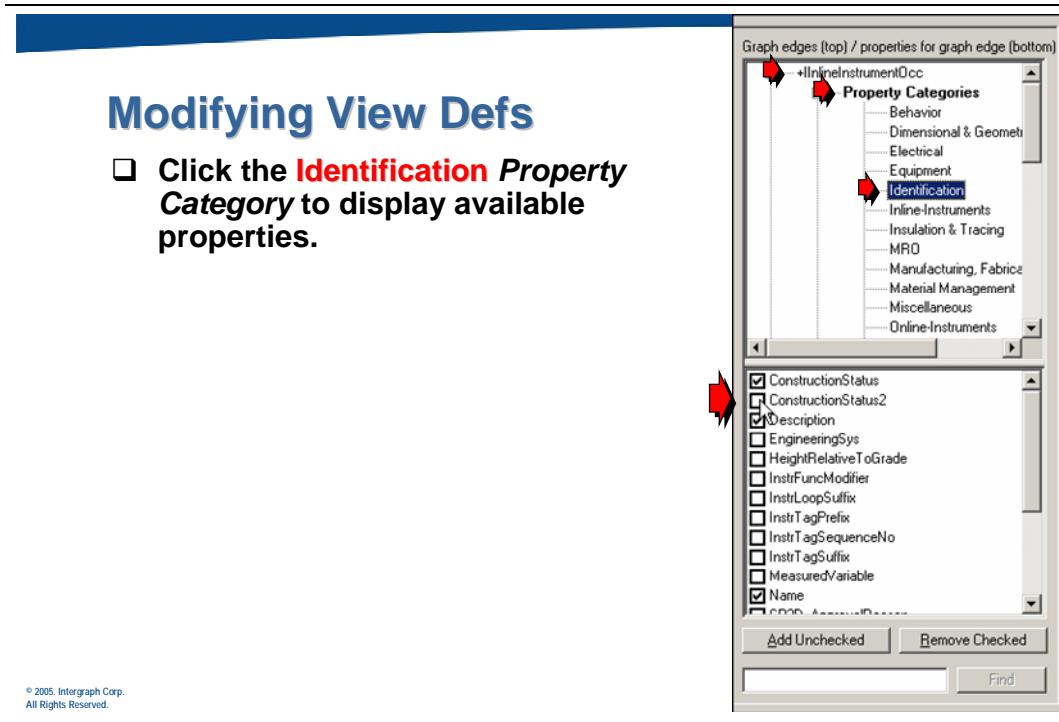
Enable the toggle box to add *ConstructionStatus2* to the selected properties of the View Definition.



Save the changes that have been made to this view definition. Then repeat this process for other view definitions, such as *PIDInlineInstrument* where you might want to see *ConstructionStatus2*.



Enable the toggle box to add *ConstructionStatus2* to the selected properties of the View Definition.

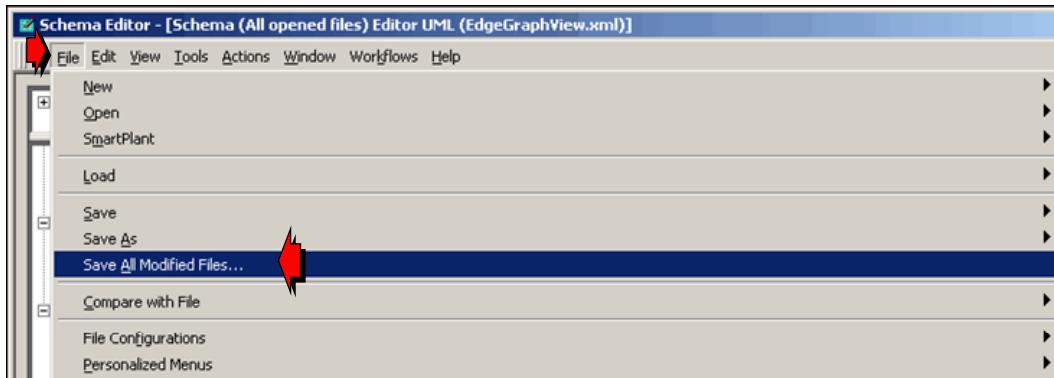


After you add the new property to the view definitions in the SmartPlant schema, save the modifications to the schema file.



Saving Schema Changes

- On the menu, click **File > Save All Modified Files...** to save the additions to the extension schema.

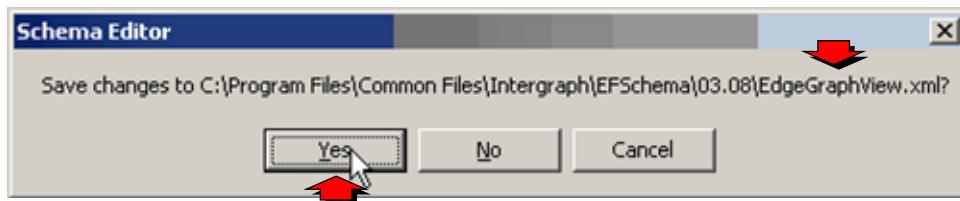


Save the changes to the **EdgeGraphView.xml** file using. This is necessary for the schema load to process successfully.

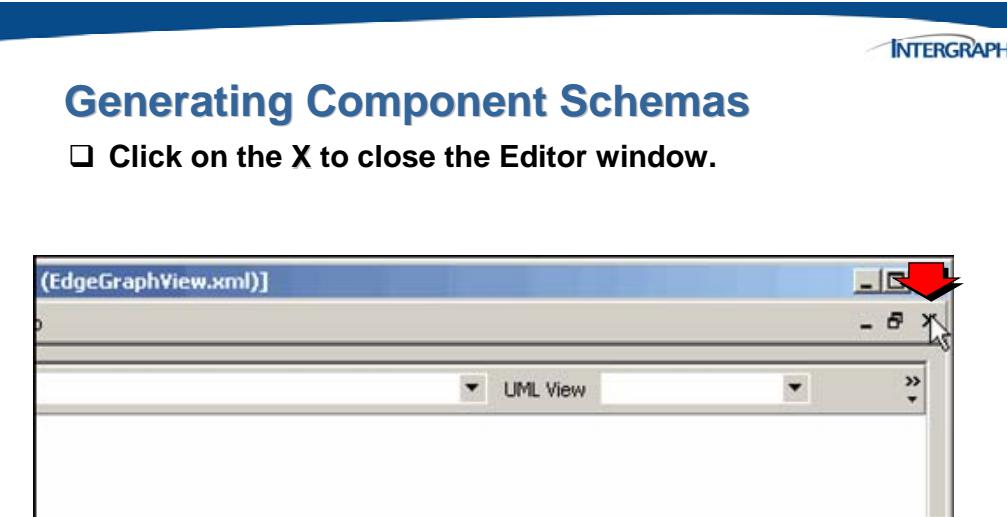


Saving Schema Changes

- Verify the extension schema name for the ViewDef additions and choose **Yes**.



Now that the view definition changes are complete, close the Editor view window.

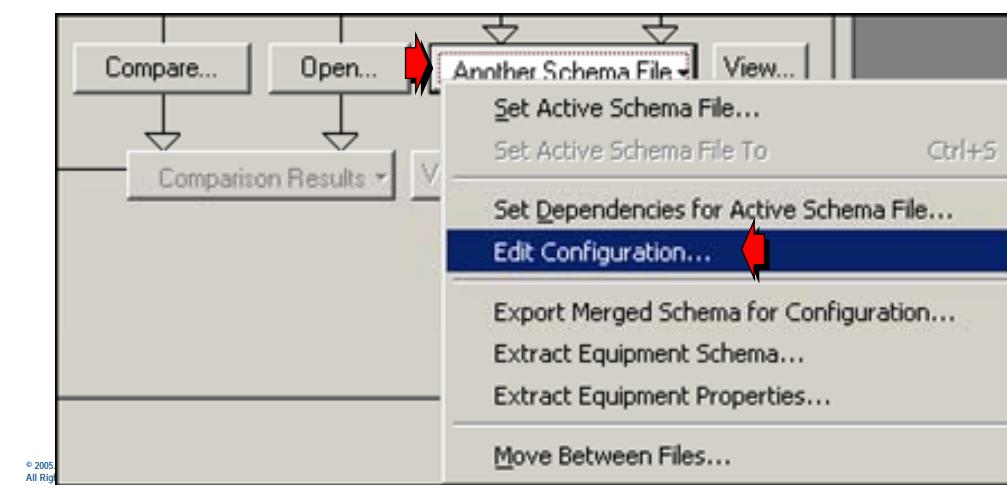


© 2005. Intergraph Corp.
All Rights Reserved.

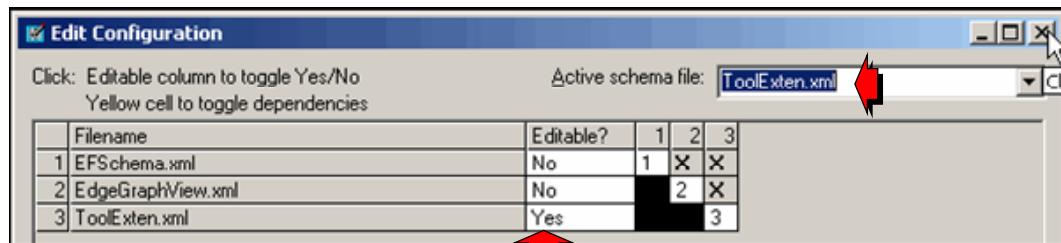
Since the new enum value for construction status has been added to the extension schema, **ToolExten.xml**, the configuration must be edited before generating new component schemas.

Generating Component Schemas

- Click on the bolded Another Schema File button, then select *Edit Configuration...*



This will allow you to make the extension schema the **Active** schema file.



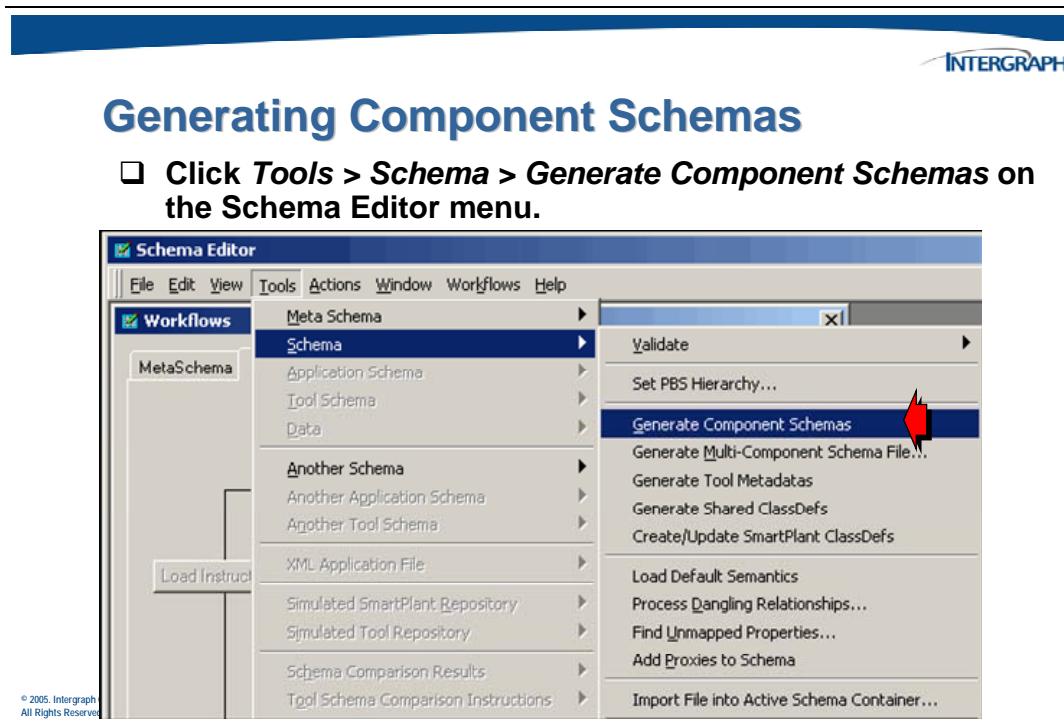
The screenshot shows a Windows application window titled "Edit Configuration". Inside, there's a table with three rows and four columns. The first column is "Filename", the second is "Editable?", and the last two are numbered 1, 2, and 3. A red arrow points to the header of the dependency columns (1, 2, 3). Another red arrow points to the "Active schema file" dropdown at the top right, which is set to "ToolExten.xml".

Filename	Editable?	1	2	3
1 EFSchema.xml	No	1	X	X
2 EdgeGraphView.xml	No	2	X	
3 ToolExten.xml	Yes		3	

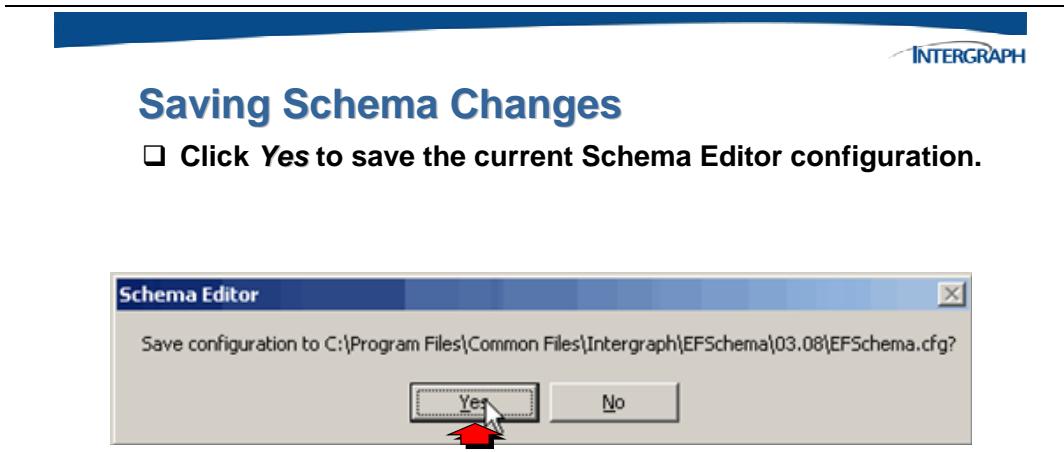
© 2005, Intergraph Corp.
All Rights Reserved.

If you modify the SmartPlant schema with any extensions, you will need to generate a new component schema in order to have the extensions available in the component schemas.

The purpose of generating new component schemas is for the updated structure that includes the modified enumerated list to be available during the publish operation. As part of the publish, a copy of the component schema specific to the publishing tool is retrieved from the server to the tool client.



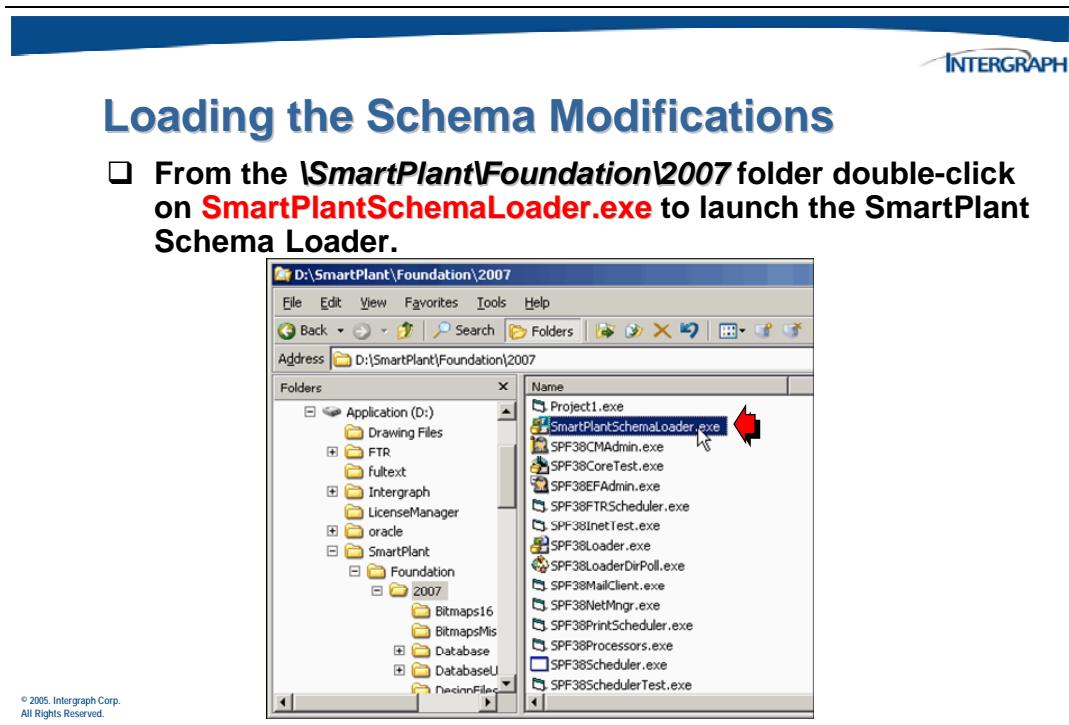
After the **Generate Component Schemas** command has completed, **Exit** from the schema editor.



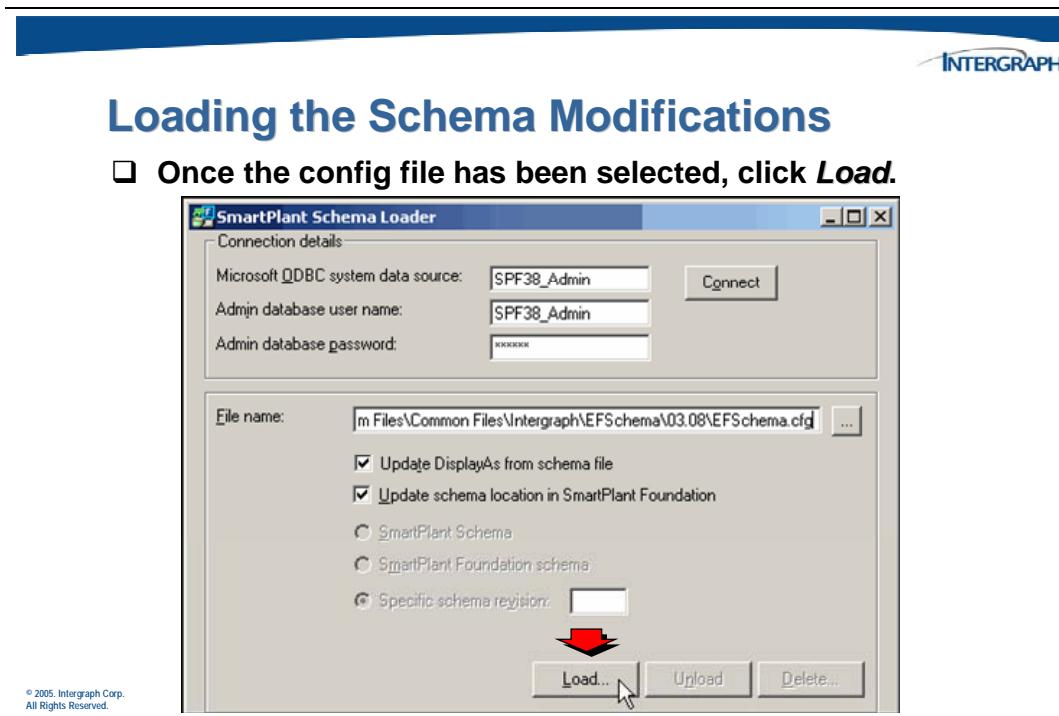
10.5 Loading the Extensions into SmartPlant Foundation

Remember from chapter 9 that the SmartPlant Schema Loader is used to load the SmartPlant schema extensions into SPF.

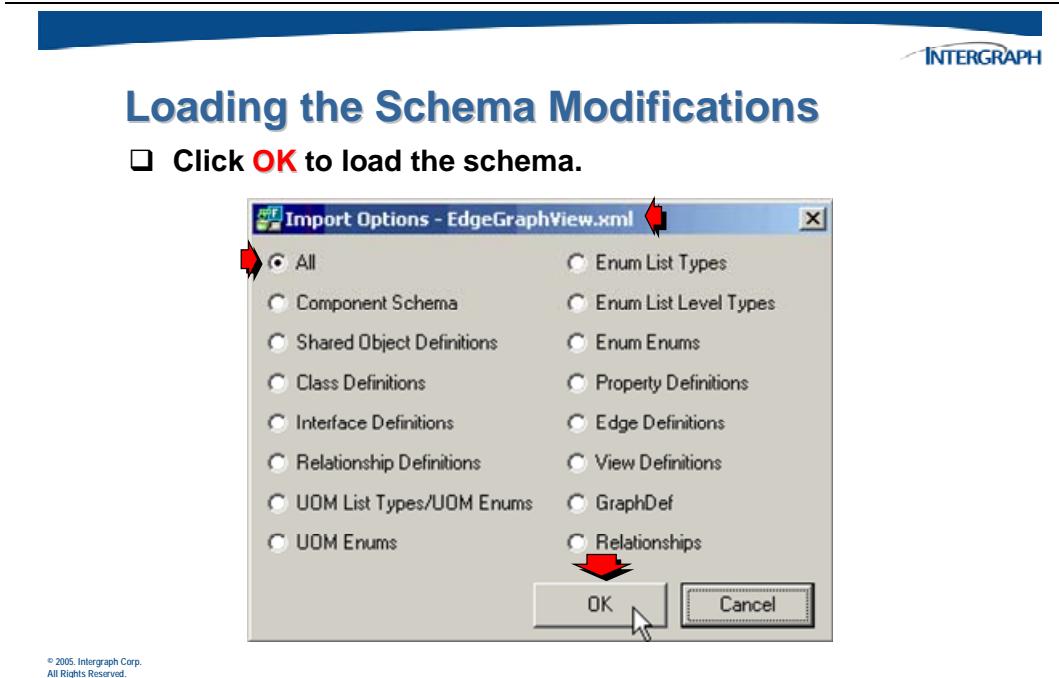
To launch the SmartPlant Schema Loader you can double-click **SmartPlantSchemaLoader.exe** or you can open a *Command Prompt* window and **cd** to the location of the executable.



The *SmartPlant Schema Loader* window displays.



The location for the saved and modified SmartPlant config file, **EFSchema.cfg**, is *C:\Program Files\Common Files\Intergraph\EFSchema\03.08*.



Due to changes in the view definitions, the EdgeGraphView file will be processed first.



Loading the Schema Modifications

The software displays the results of the load below the Load progress bar.

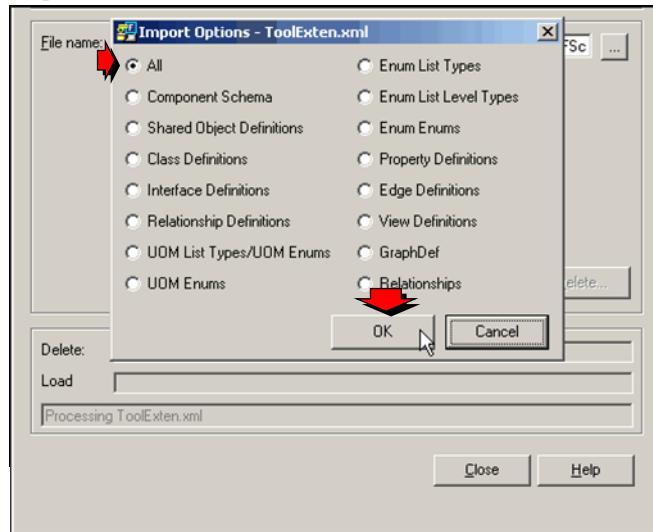


Once the EdgeGraphView file has been loaded, the **ToolExten** file will be processed.



Loading the Schema Modifications

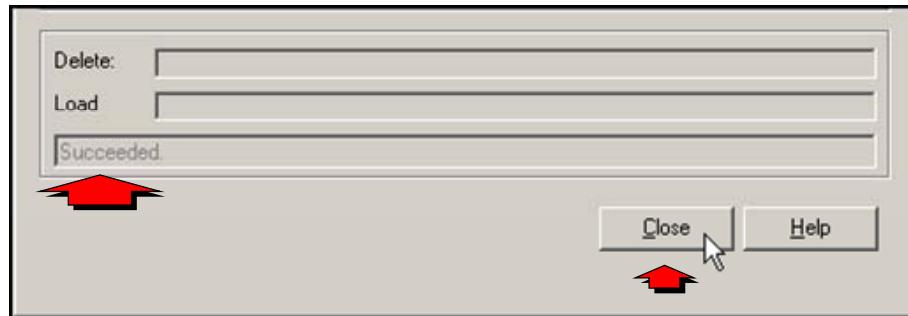
- Click **OK** to load the schema.



After both changed extension files have been loaded, exit from the schema loader.

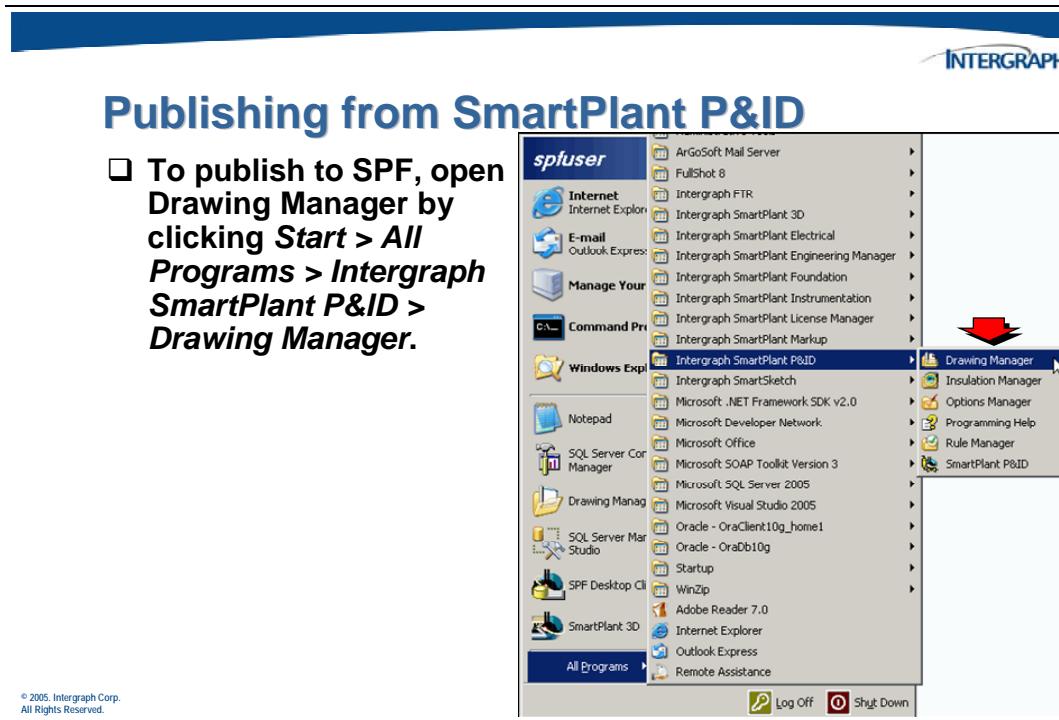
Loading the Schema Modifications

- ❑ Click the **Close** button once the schema has been loaded.

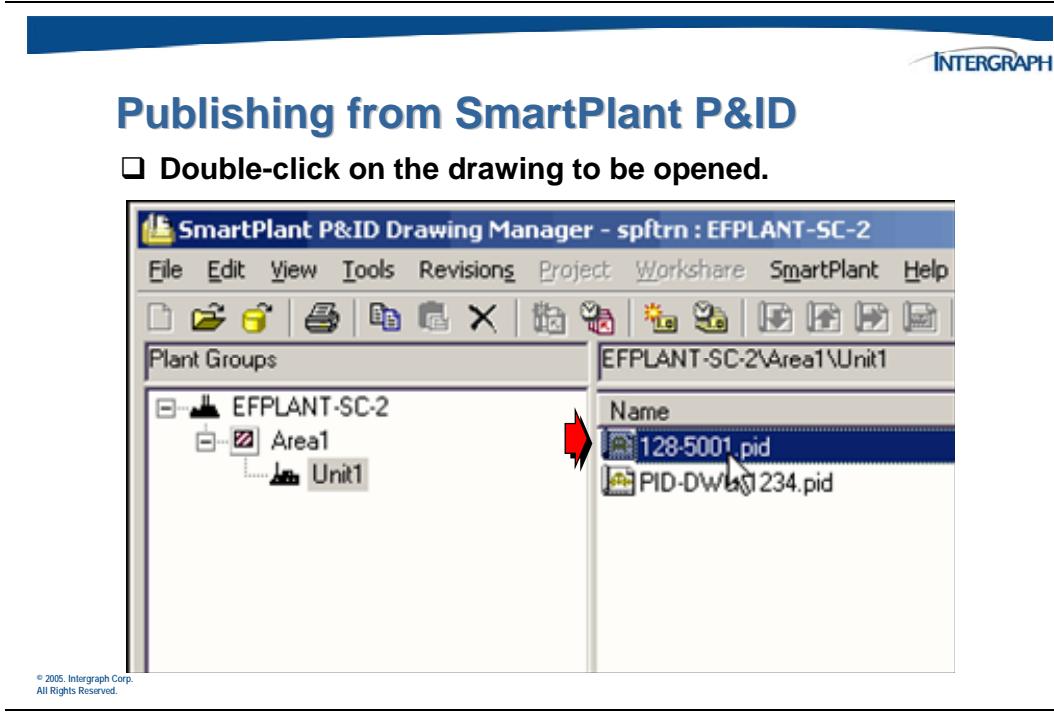


10.6 Publishing a Change from SmartPlant P&ID

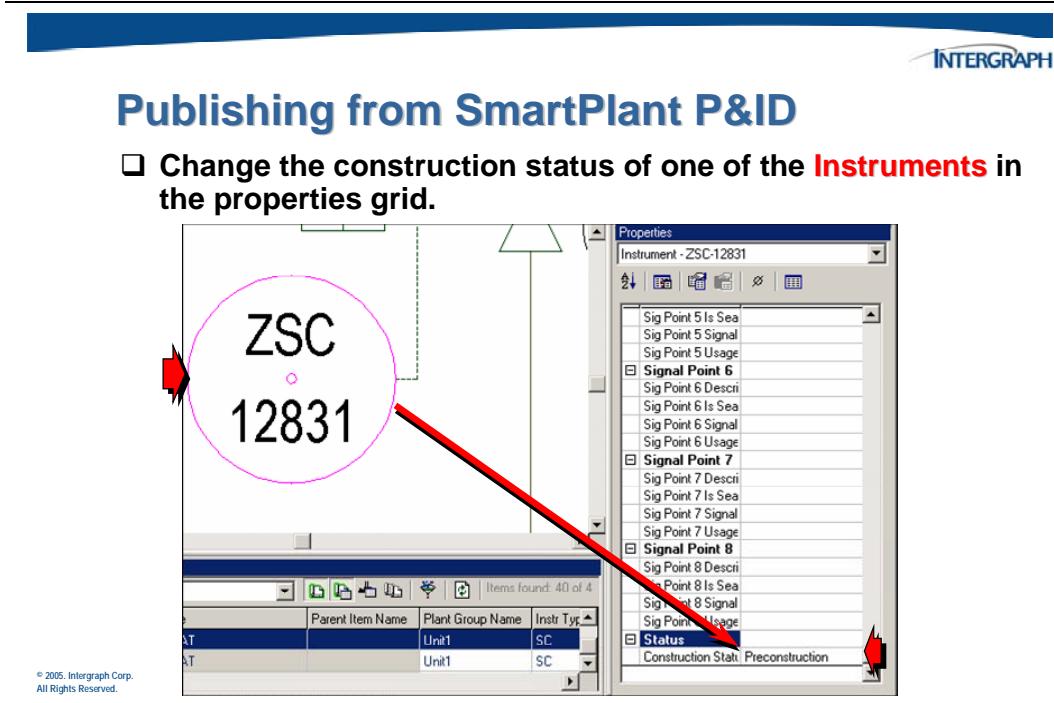
In this section you will see an example of publishing a change to a P&ID instrument using the new Construction Status picklist entry of *Preconstruction*. Then to test the SmartPlant capabilities, the changed instrument will be retrieved into SmartPlant Instrumentation and the Construction Status value reviewed.



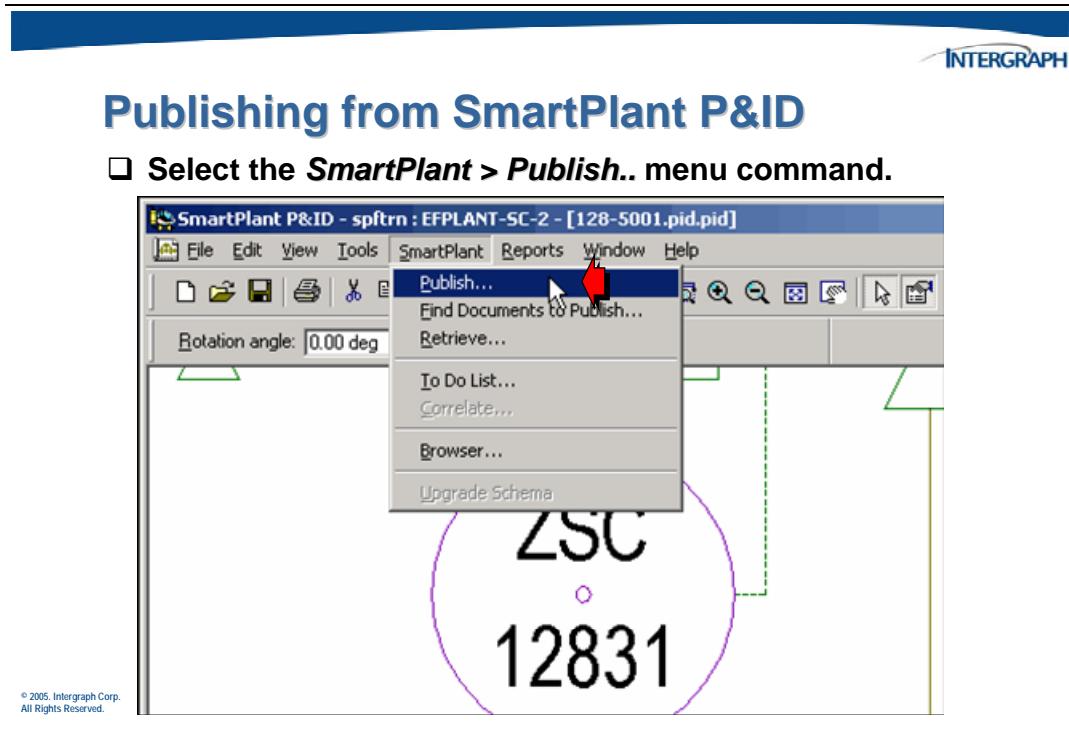
The *SmartPlant P&ID Drawing Manager* window will display. The registered plant to be used for the publish operation is **EFPLANT-SC-2**.



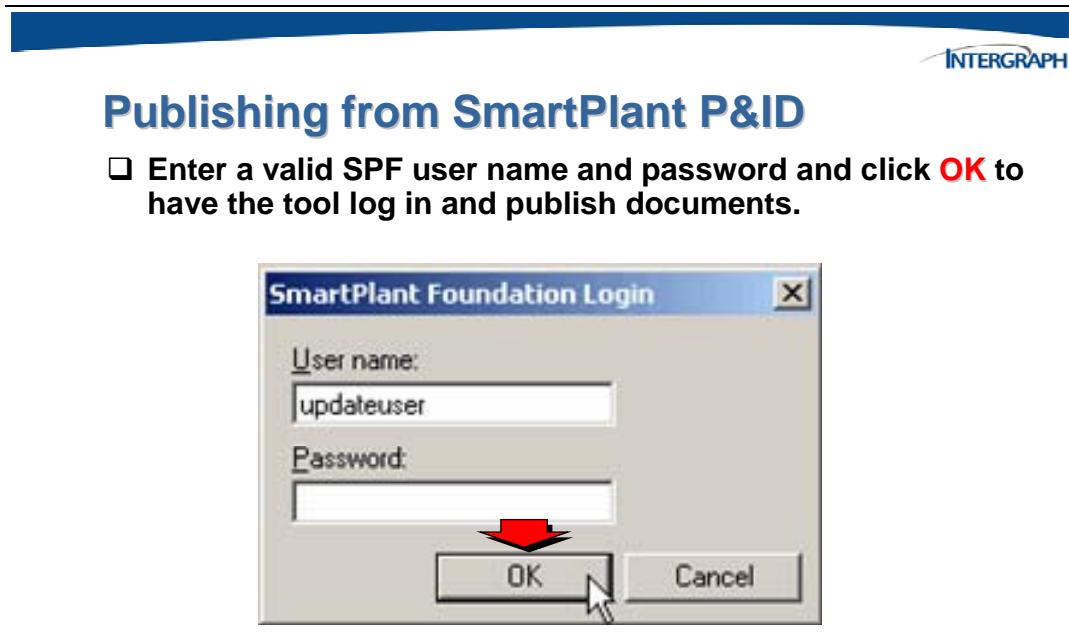
Locate some of the instruments that have been placed in this drawing. This instrument has already been published previously. Update this existing instrument by adding values for the new status.



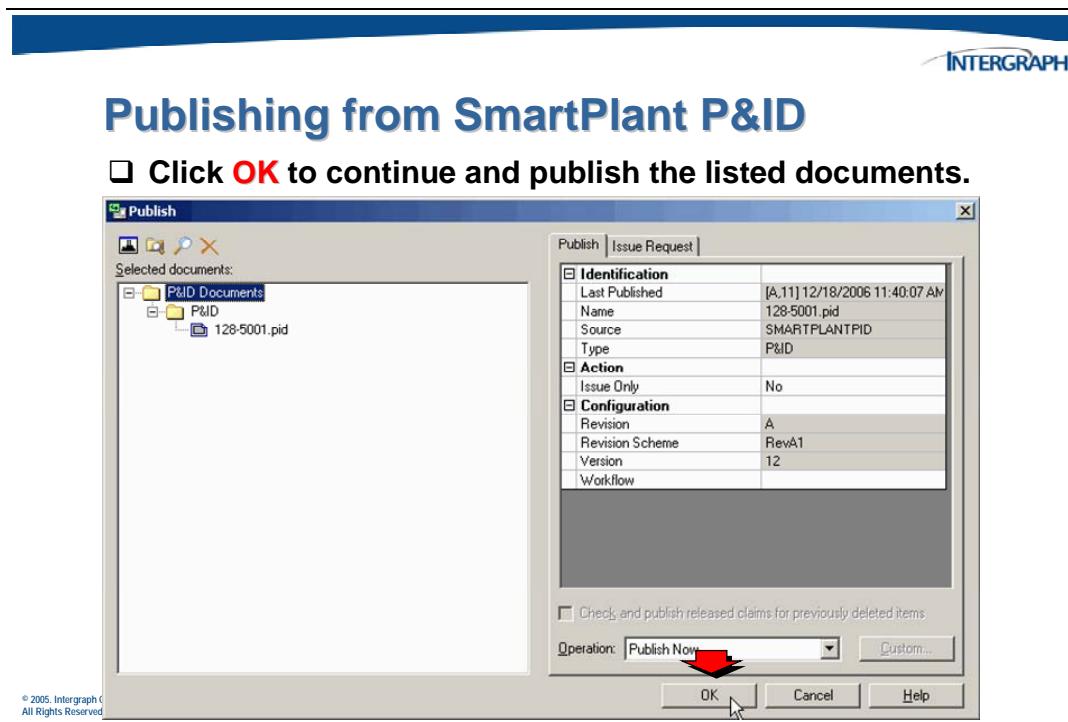
Save the change that has been made before publishing this document to SPF. To perform a publish operation; use the **Publish** command from SmartPlant P&ID.



A *SmartPlant Foundation Login* dialog box will be displayed allowing you to specify a user name to be used for the publish operation.



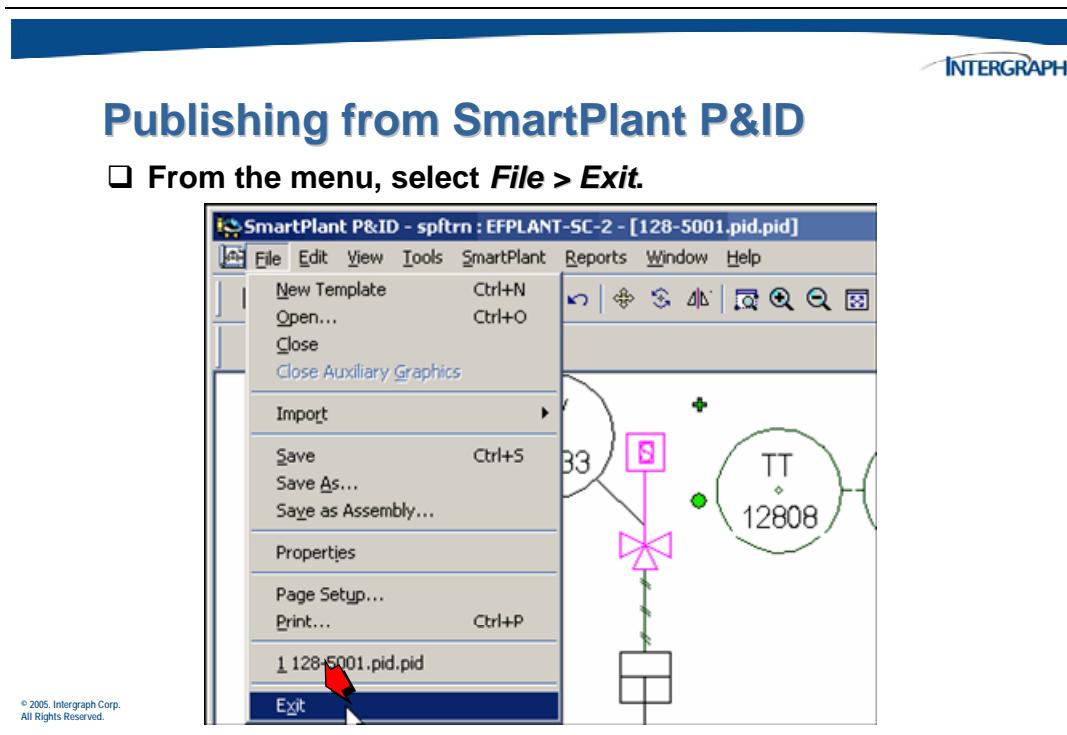
The *Publish* dialog box will display.



A dialog box will display after some time to report success or failure of the publish.



Now that the document has been successfully published, exit from SmartPlant P&ID. In the next section, this document will be retrieved into SmartPlant Instrumentation.

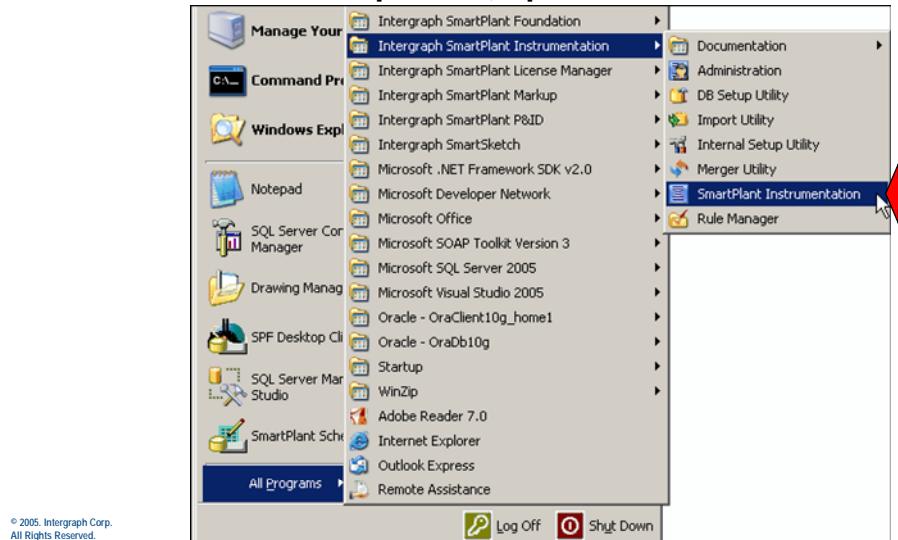


10.7 Retrieving a Change into SmartPlant Instrumentation

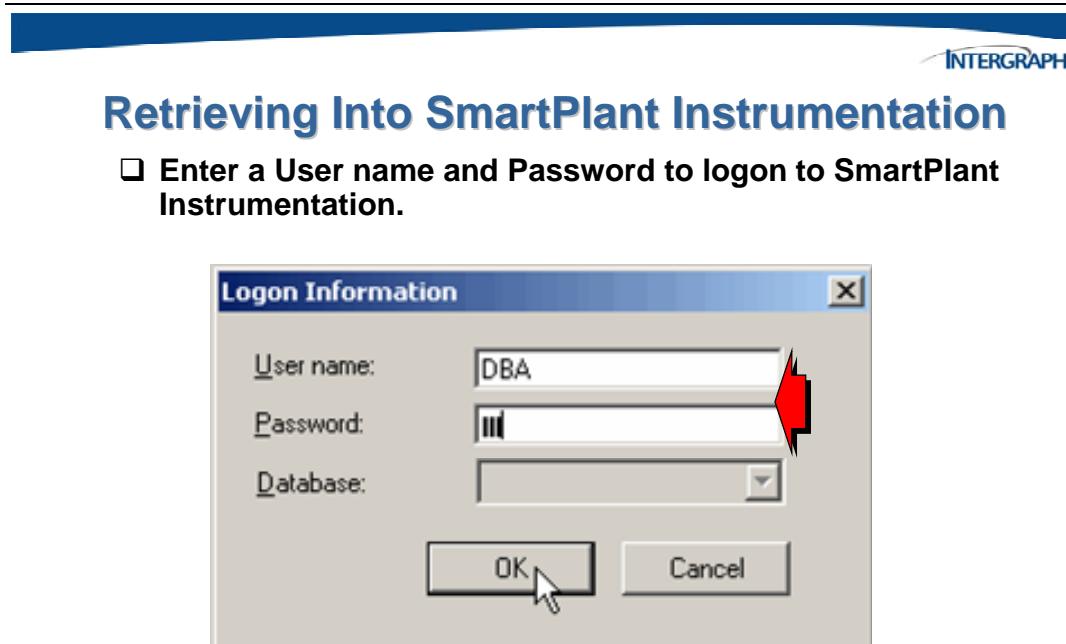
Now that the instrument change has been published into SmartPlant, the document can be retrieved by SmartPlant Instrumentation.

Retrieving Into SmartPlant Instrumentation

- To retrieve the publish, open SmartPlant Instrumentation.

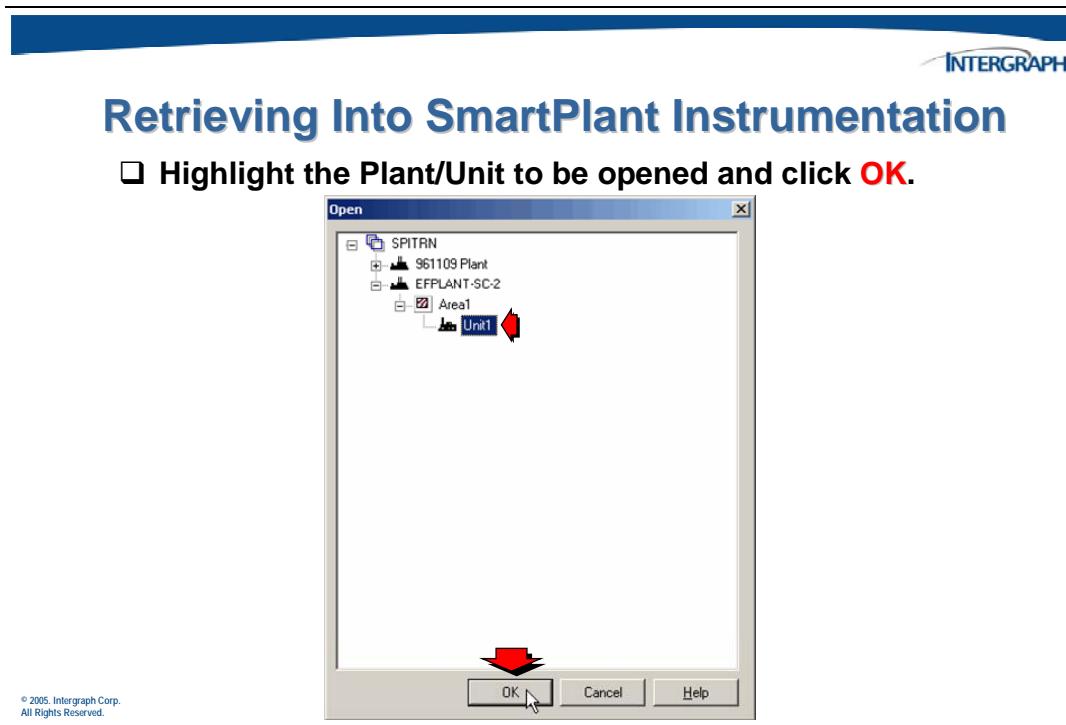


The login dialog will display.



© 2005, Intergraph Corp.
All Rights Reserved.

When the *Open* window displays, choose a plant/unit from the displayed tree.



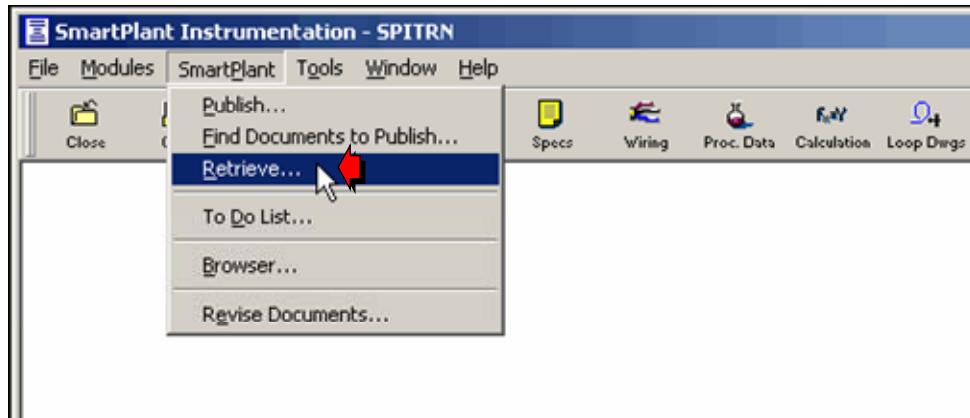
© 2005, Intergraph Corp.
All Rights Reserved.

To perform a retrieve operation; use the **Retrieve** command from SmartPlant Instrumentation.



Retrieving Into SmartPlant Instrumentation

- Select the **SmartPlant > Retrieve...** menu command.



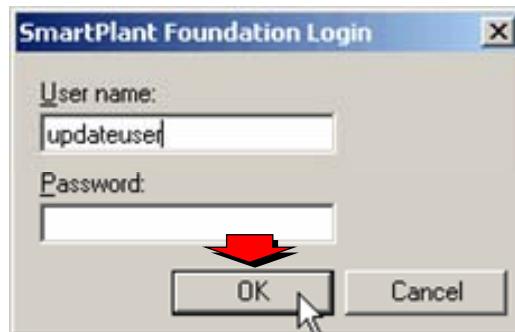
© 2005, Intergraph Corp.
All Rights Reserved.

The SmartPlant Foundation login dialog will display.



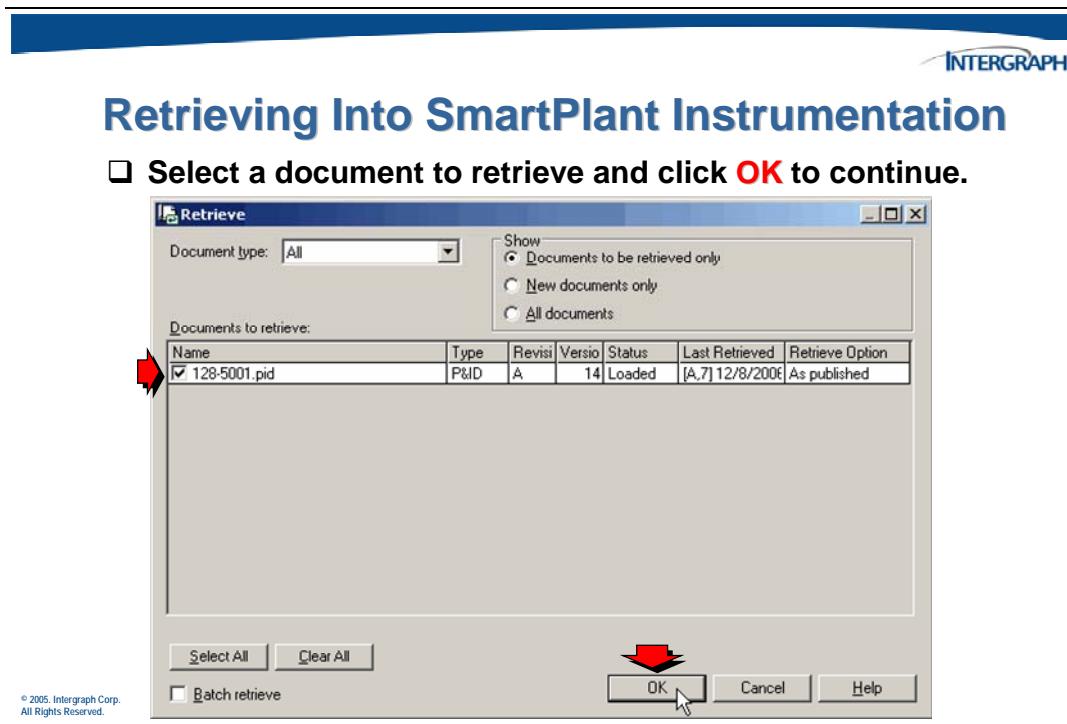
Retrieving Into SmartPlant Instrumentation

- Enter a valid SPF user name and password and click **OK** to have the tool log in and retrieve documents.

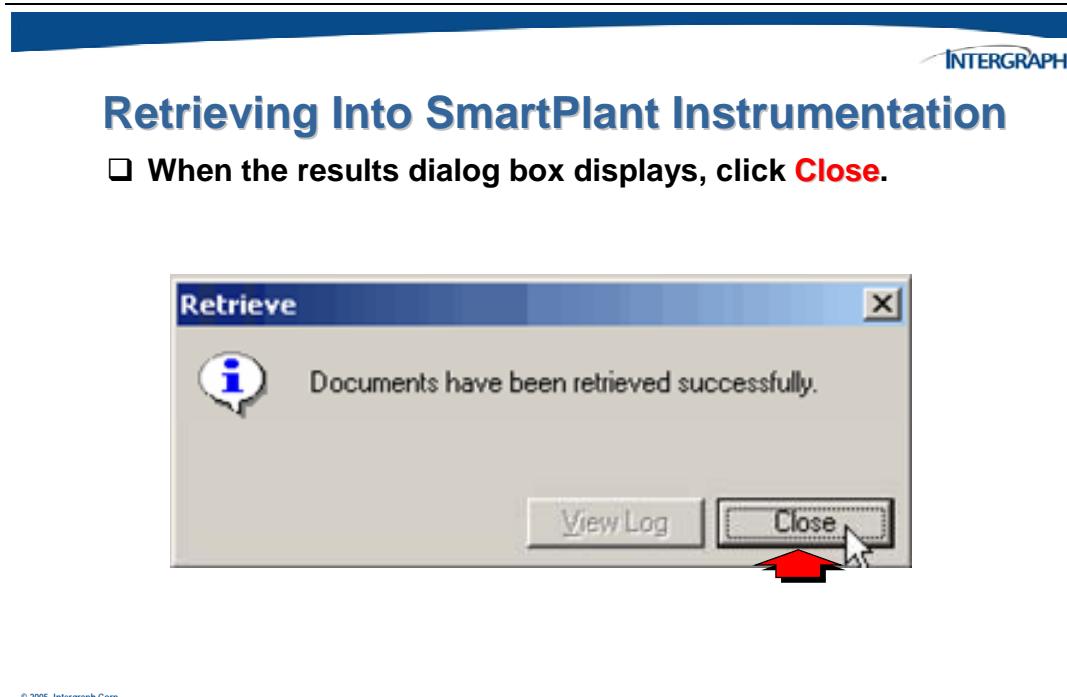


© 2005, Intergraph Corp.
All Rights Reserved.

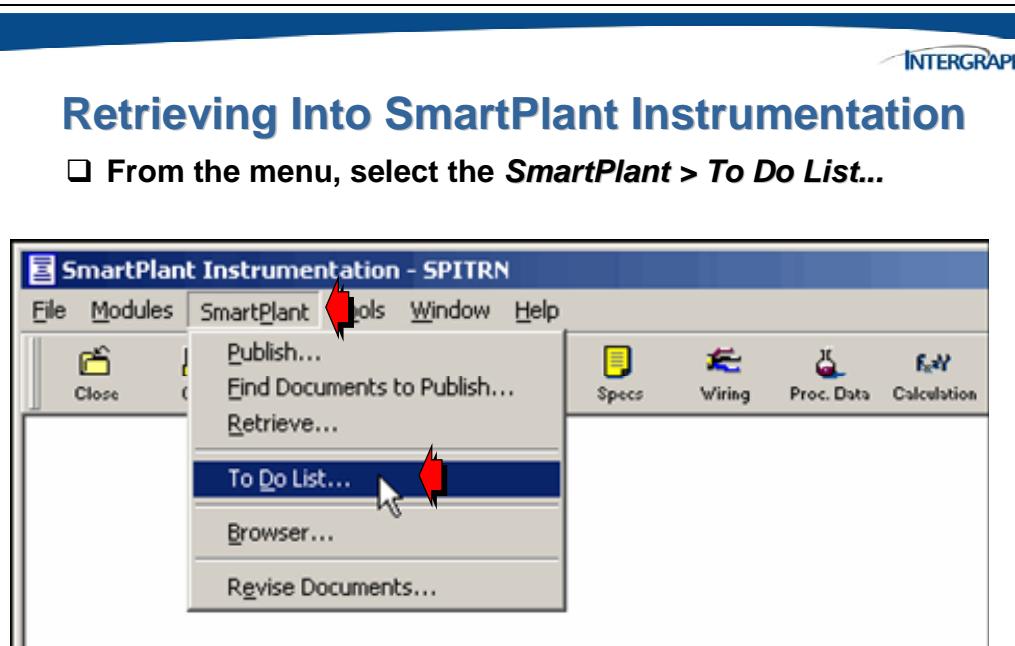
When the retrieve dialog displays, enable the toggle box next to the previously published documents (from SPPID) to be retrieved by SmartPlant Instrumentation.



A dialog box will display after some time to report success or failure of the publish.

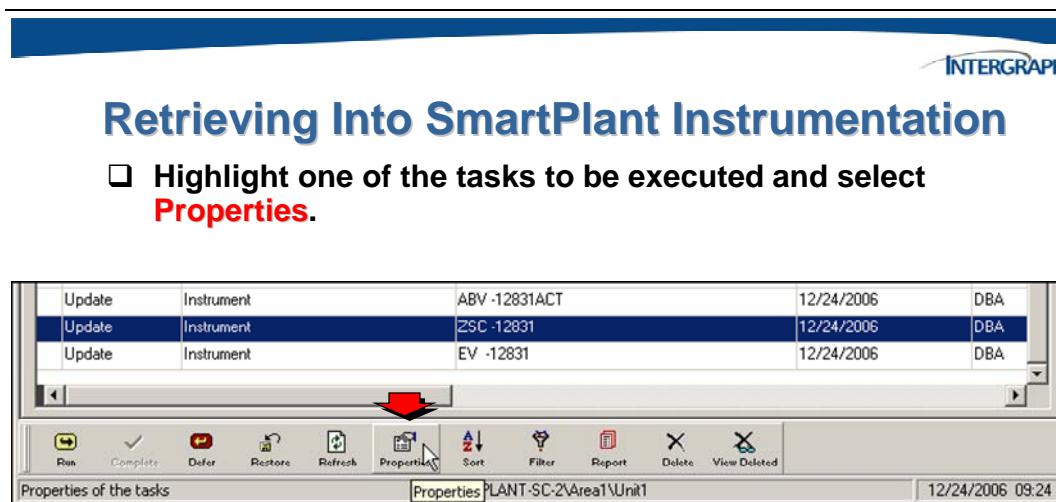


The instrument changes are not automatically applied by SmartPlant Instrumentation.



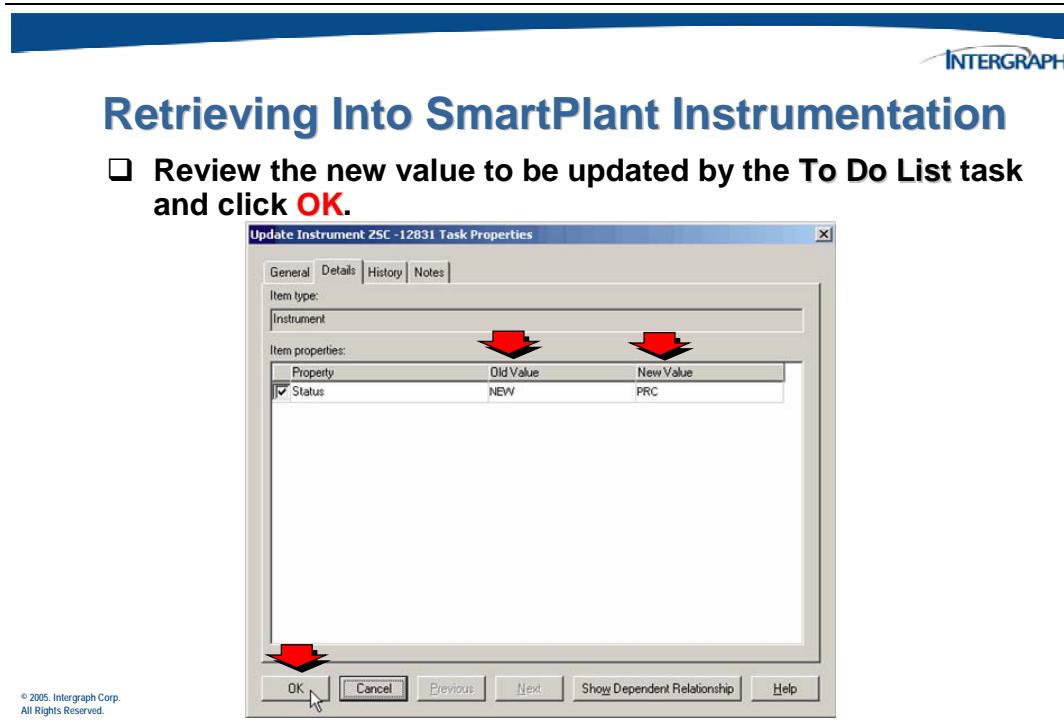
© 2005, Intergraph Corp.
All Rights Reserved.

The engineer will display the **To Do List** which contains all of the commands that can be executed within SmartPlant Instrumentation.

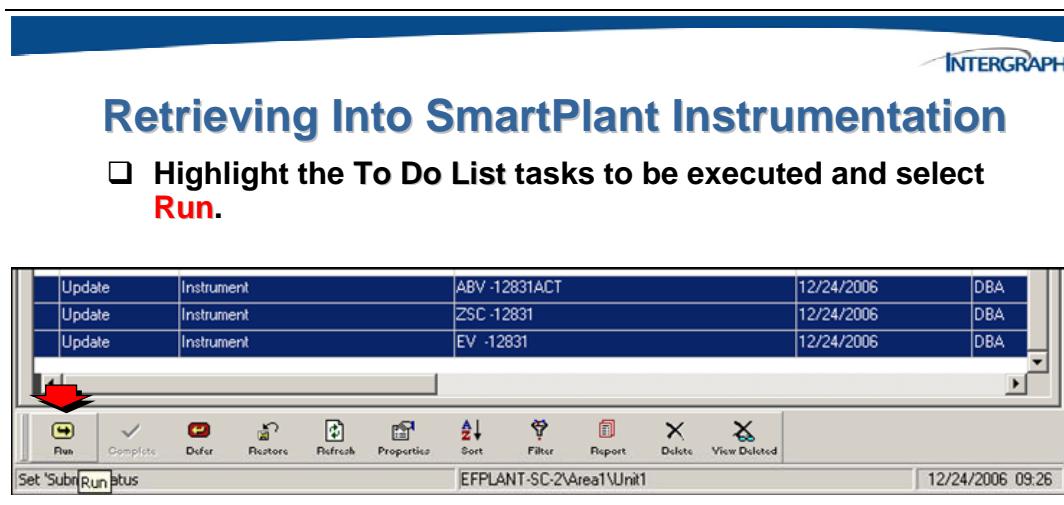


© 2005, Intergraph Corp.
All Rights Reserved.

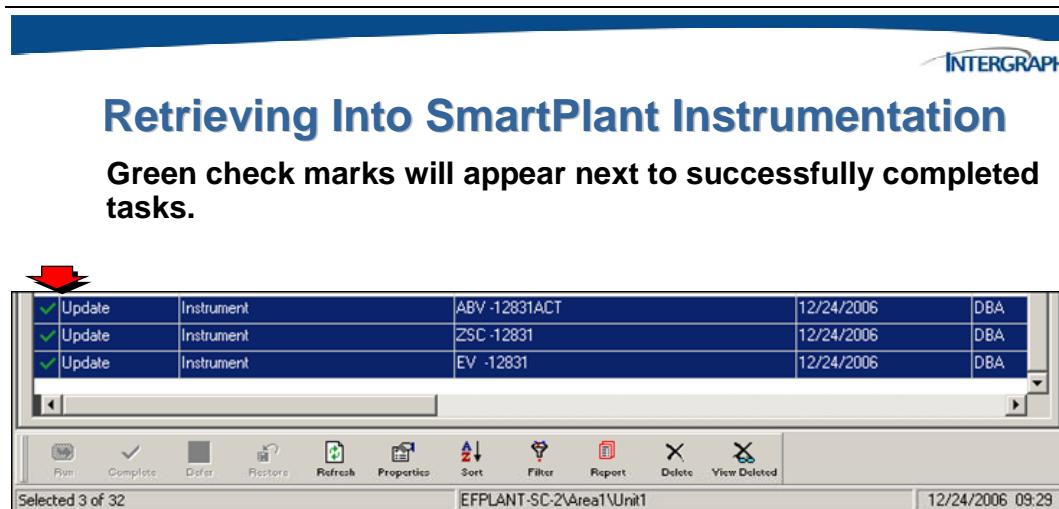
This will allow the engineer to review the new value before it is applied to the existing data.



When the engineer is ready, the retrieved tasks can be executed by using the **Run** command.



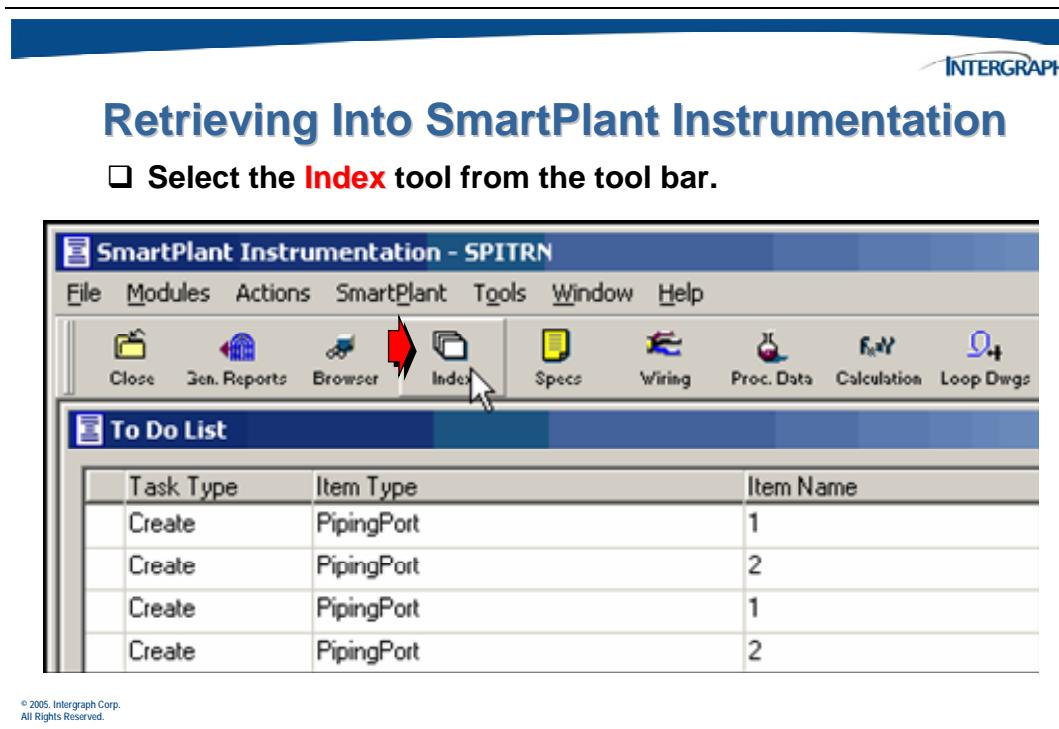
A status icon will be displayed in the left most column as tasks are completed.



The screenshot shows a software window titled "Retrieving Into SmartPlant Instrumentation". Below the title, a message states "Green check marks will appear next to successfully completed tasks." A red arrow points to the first row of a table. The table has columns for Task Type (Update), Item Type (Instrument), Item Name (ABV-12831ACT, ZSC-12831, EV-12831), Date (12/24/2006), and User (DBA). The first three rows have green checkmarks in the Task Type column. The bottom status bar shows "Selected 3 of 32" and the date "12/24/2006 09:29".

© 2005, Intergraph Corp.
All Rights Reserved.

Once the tasks have been executed, use one of the browser windows to view the results.

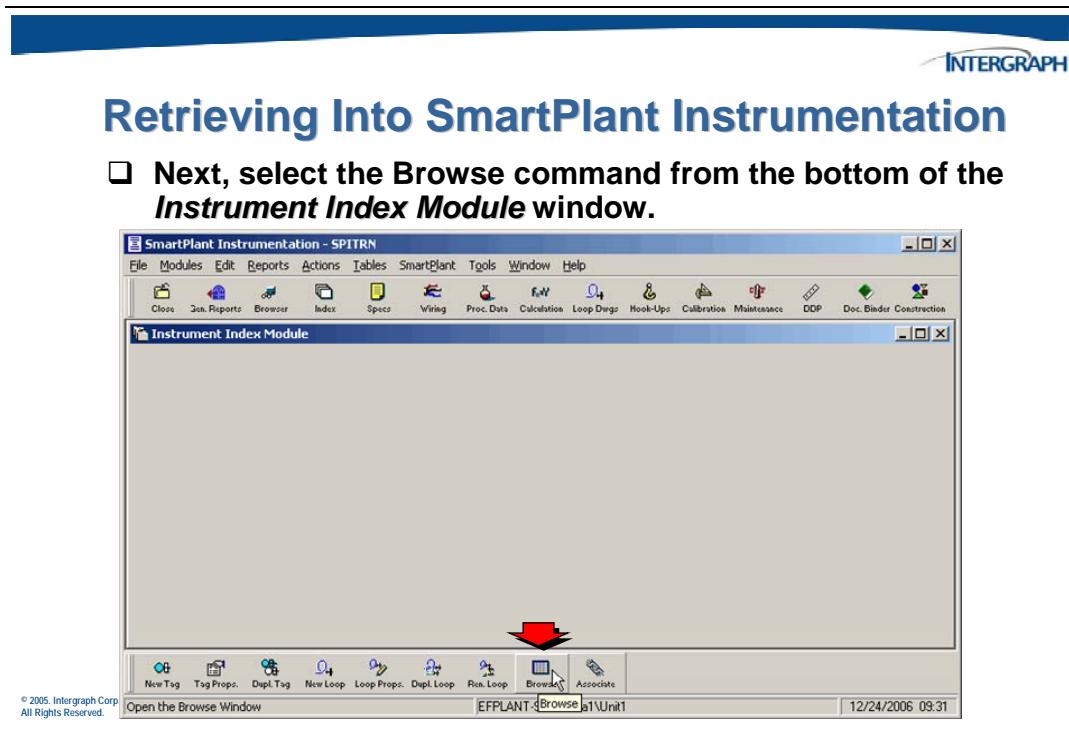


The screenshot shows a software window titled "SmartPlant Instrumentation - SPITRN". The menu bar includes File, Modules, Actions, SmartPlant, Tools, Window, and Help. The toolbar below the menu bar has icons for Close, Gen. Reports, Browser, and Index (which is highlighted with a red arrow). The main area is titled "To Do List" and contains a table:

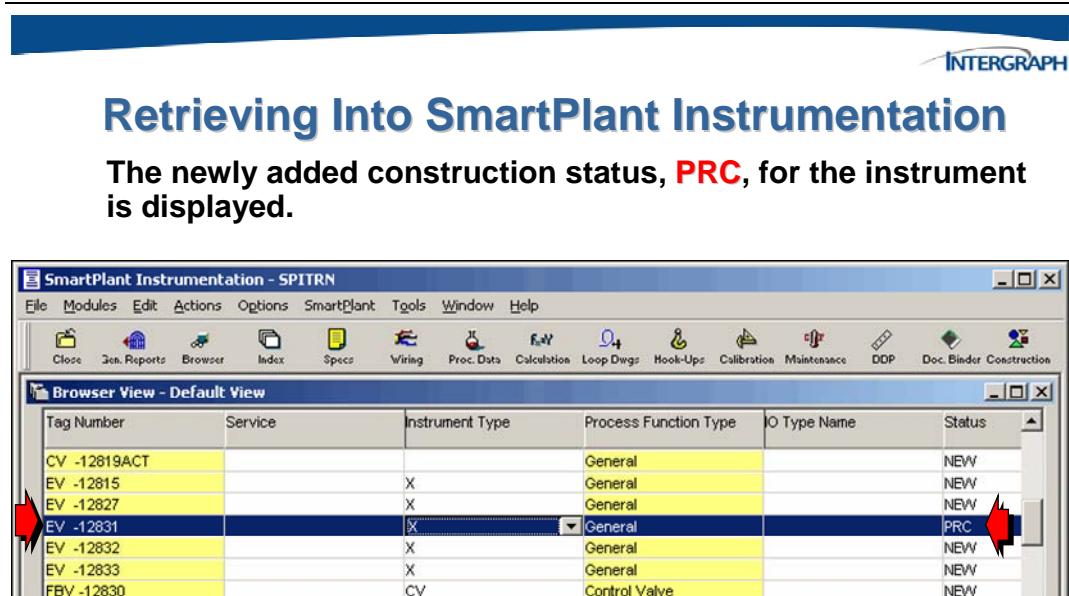
Task Type	Item Type	Item Name
Create	PipingPort	1
Create	PipingPort	2
Create	PipingPort	1
Create	PipingPort	2

The bottom status bar shows "© 2005, Intergraph Corp.
All Rights Reserved."

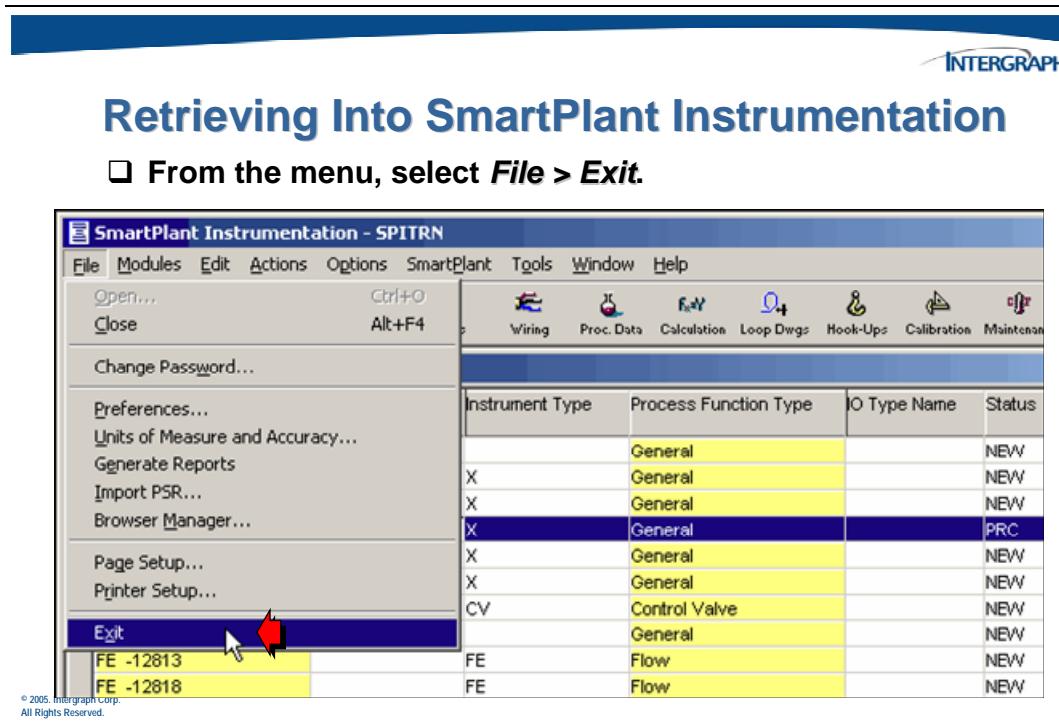
The *Instrument Index Module* window will display.



A *Browser View* window with instrument information will be displayed.



Now that the document has been successfully retrieved and the selected tasks executed, exit from SmartPlant Instrumentation.

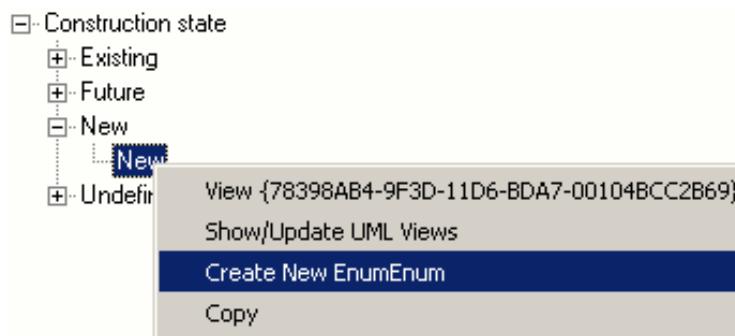


10.8 Activity 1 – Extending an Existing Enumerated List

The objective of this activity is to use the Schema Editor to extend the SmartPlant Schema and two application tool schemas. Once this has been done, it will be possible to publish a document from the P&ID authoring tool and retrieve the document in the SP Instrumentation tool and view the results of the retrieve.

1. Open SmartPlant P&ID Data Dictionary Manager and add a new entry, Preconstruction, to the **Construction Status** *select list*. Be sure and save your changes.
2. Start the *Schema Editor* by selecting *Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor*.
3. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant...**
4. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with
 - Click **Next**
 - Select the tool map schema to be loaded, *SMARTPLANTPID/SmartPlant P&ID Tool Schema*
 - Enable the **Load map schema** toggle
 - Enable the **Connect to application schema** toggle
 - Click **Finish**
5. When the *Synchronize* dialog displays, confirm that the new select list values have been imported by the meta data adapter and click **OK**.

6. Add new entries to an existing Enumerated List in the SmartPlant schema to correspond to new entries that have been added to a tool map schema during the synchronization.
 - Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant P&ID Tool Schema entries are displayed.
 - Expand the **Map Enumerated Lists** and the **ConstructionStatus** objects.
 - Right-click on the ConstructionStatus entry in the tree and choose **Edit SP_93** from the dynamic menu.
 - Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog. Then right-click on the **New** existing enum **child** entry in the SmartPlant control and choose **Create New EnumEnum** from the dynamic menu.



- In the *New Enumerated Entry* dialog box, enter the following information:
 - Name/short description – **Preconstruction**
 - Long description – **Preconstruction**
 - Click **OK** to add the new entry
7. Highlight the *Preconstruction* MapEnum in the **Application** upper tree control and the *New* enumerated child list in the **SmartPlant** upper tree control in order to perform the schema mapping.
 - Make sure that **Preconstruction** is highlighted in the middle control in the application section and that **Preconstruction** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPPID tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *Preconstruction* maps to *Preconstruction* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Enumerated List Definition* dialog box.

8. Save the changes to the all schema files.
 - From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **Yes**.
9. Use Windows Explorer to verify that all the files were uploaded to **C:\Program Files\Common\Intergraph\EFSchema\03.08** with the latest time and date.
10. Open SmartPlant Instrumentation (use the User name / Password of DBA) and add a new entry to the **Instrument Statuses** table. Refer to the steps outlined in section 10.3 of this chapter. Run a report of instrument statuses to determine the unique number associated with your new entry.
11. View the results of the instrument statuses report with Microsoft Excel. Write the values for columns A, B and C in this space **A_____**, **B_____**,
C_____ for the new instrument status.
12. Start the *Schema Editor* by selecting **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor**.
13. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant...**
14. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with
 - Click **Next**
 - Select the tool map schema to be loaded, *INtools/INtools Tool Schema*
 - Enable the **Load map schema** toggle
 - Enable the **Connect to application schema** toggle

Click **Finish**

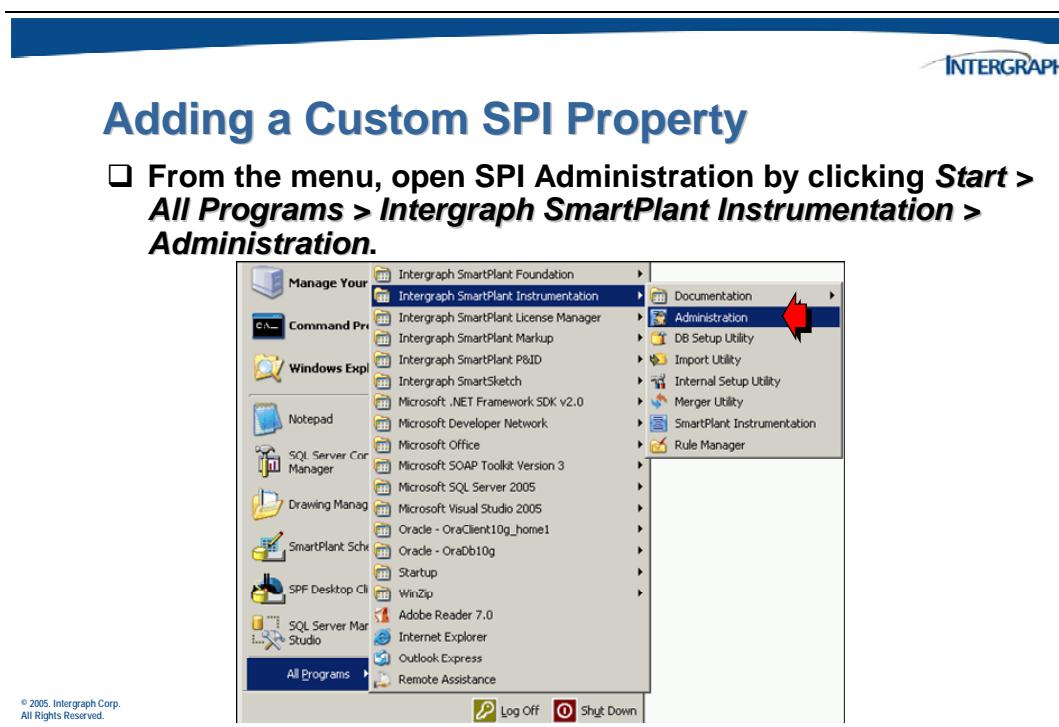
15. When the *Synchronize* dialog displays, confirm that the new select list values have been imported by the meta data adapter and click **OK**.
16. Map the new entry to an existing Enumerated List in the SmartPlant schema to correspond to the new entry that has been added to a tool map schema during the synchronization.
 - Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant Instrumentation Tool Schema entries are displayed.
 - Expand the **Map Enumerated Lists** and the **ConstructionStatuses** objects.
 - Right-click on the **ConstructionStatuses** entry in the tree and choose **Edit INTL_ConstructionStatuses** from the dynamic menu.
 - Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog.
17. Highlight the *Preconstruction* entry in the middle control in the application section and *Preconstruction* in the SmartPlant section in order to perform the schema mapping.
 - Click the **Map** button to define the mapping between the SPI tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *Preconstruction* maps to *Preconstruction* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Select the **Retrieve** tab at the top of the *Edit Map Enumerated List Definition* dialog.
 - Again, highlight the *Preconstruction* entry in the middle control and *Preconstruction* in the SmartPlant section in order to perform the retrieve mapping.
 - Click **OK** to close the *Edit Map Enumerated List Definition* dialog box.
12. Save the changes to the all schema files.
 - From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **Yes**.

19. Open Windows Explorer and clean up the temporary cache folder, **C:\Documents and Settings\spfuser\Local Settings\Temp\EFSchemaCache\03.08\pimdemo1_spf38asp**. Delete all the files in this folder.
 - Open schema editor and click **File Configurations > Open Configuration....**
 - Browse to **C:\Program Files\Common\Intergraph\EFSchema\03.08** in the *Open Configuration File* dialog box.
 - Select **EFSchema.cfg** file and click **Open**.
 - Make sure the *Active schema file* is **ToolExten.xml**.
 - On the *Schema Editor* menu, click **Tools > Schema > Generate Component Schemas**.
20. Load the changed SmartPlant Schema into the SPF administration database so that the publish will pick up the new enumerated list values.well, an
 - Use the **EFSchema.cfg** file to perform the schema load.
21. Using the example in section 10.6, make a change to some instruments using your new construction status entry value. Save your changes in P&ID and then publish the change to SPF.
 - Use the schema editor to open the published XML file to verify the new construction values are being written correctly.
22. Using the example in section 10.7, retrieve the P&ID instrument into SP Instrumentation.
 - Select the **SmartPlant > Retrieve...** command to retrieve the changes/additions. Use only 128-5001 and NOT 128-5001.pid.
 - Use the **To Do List...** command to view the changes/additions.
 - Review the *Properties* of one of the instrument objects that has a status of **Update** to see the *Old* value and the *New* value once this task is executed.
 - Select and **Run** the *Update* tasks to apply the new retrieved values.
 - Use the Instrument Index Module to review the results of the retrieve.
23. When you are finished with this activity, you may take a short break until the other students have finished.

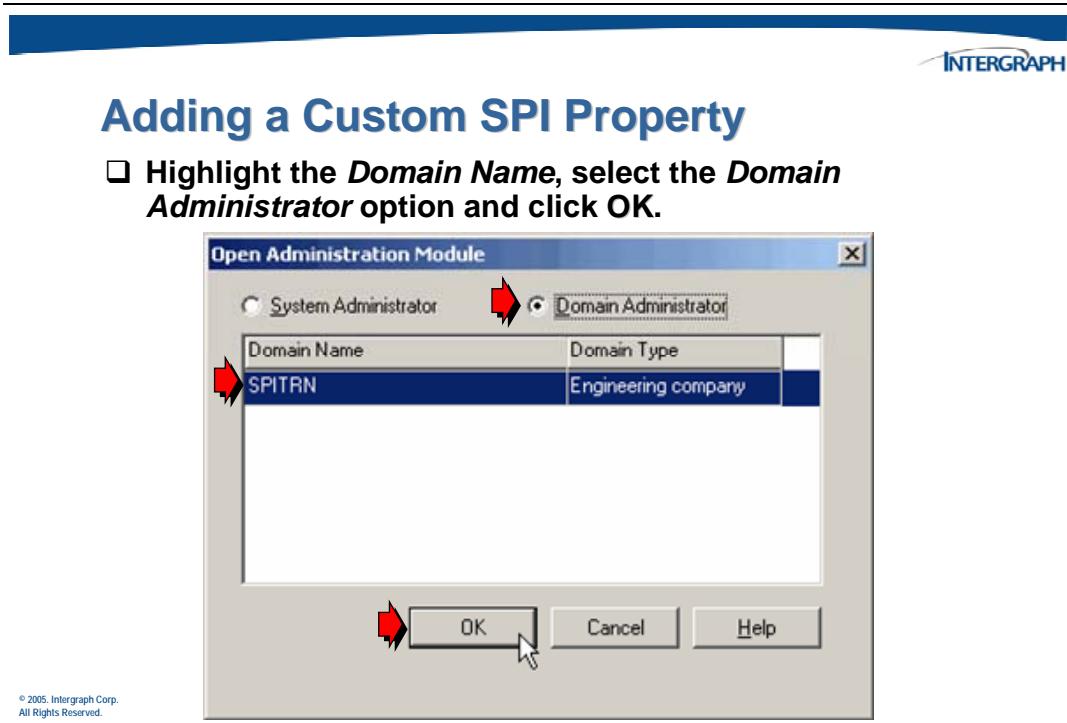
10.9 Adding a Custom Property to SmartPlant Instrumentation

In this section, the process of adding a new custom property to SmartPlant Instrumentation (SPI) and then mapping that property to the SmartPlant schema will be discussed. The type of property that will be added is a simple or string property. The ability to add new enumerated properties is not possible within SPI.

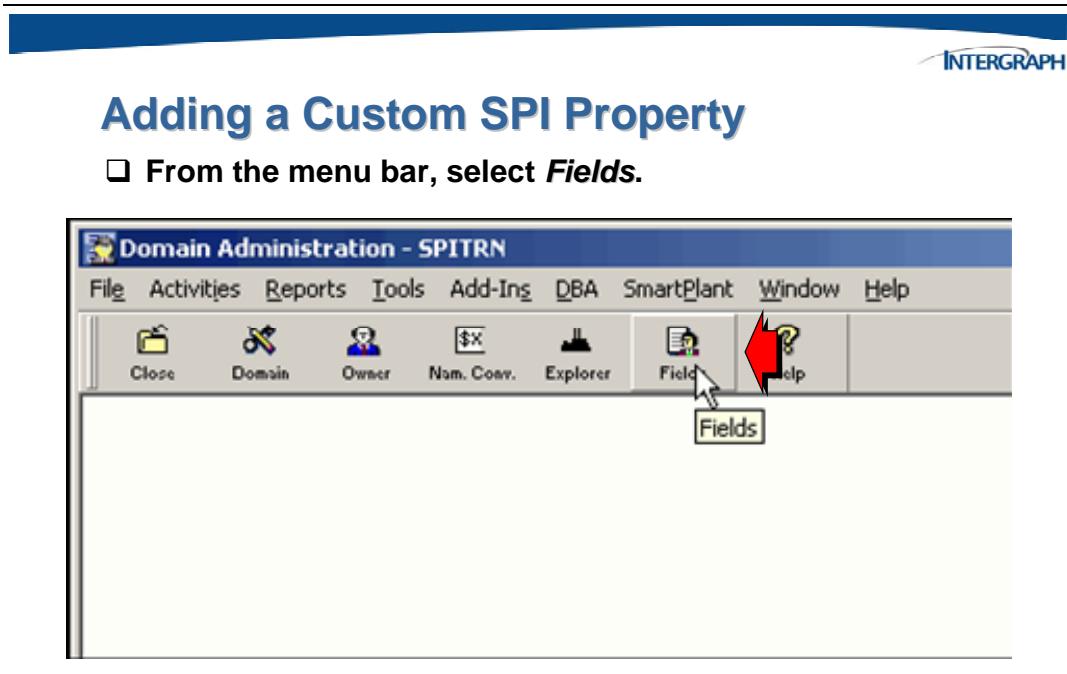
The first step is to use the SPI Administration module to add the new property, called a UDF (user defined field).



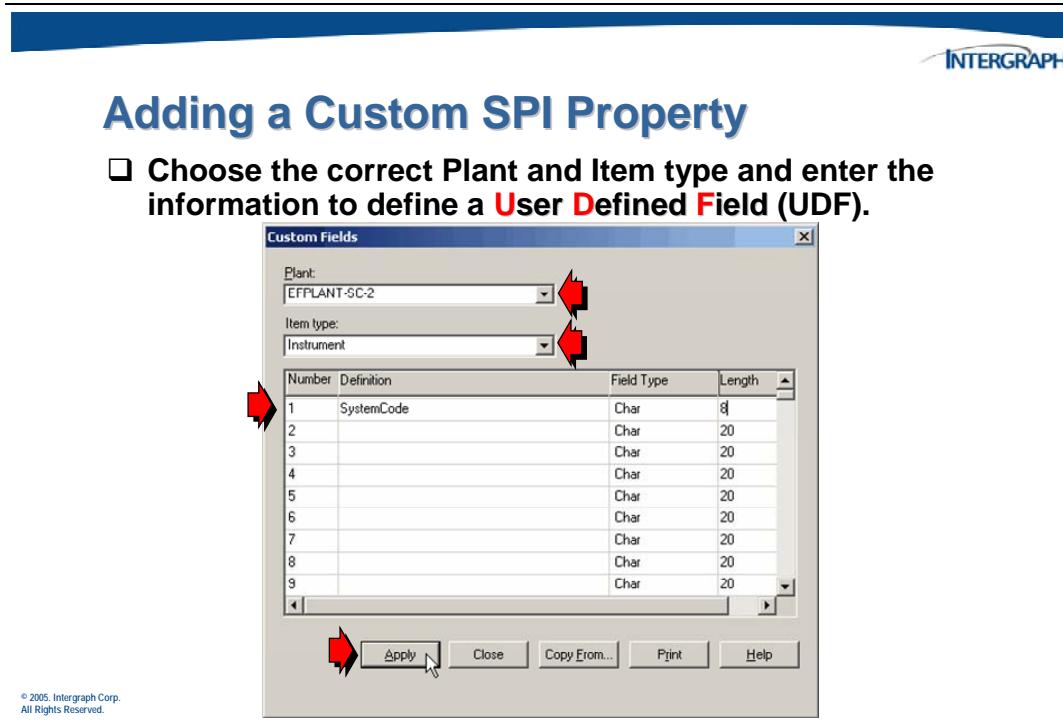
The *Open Administration Module* dialog will be displayed.



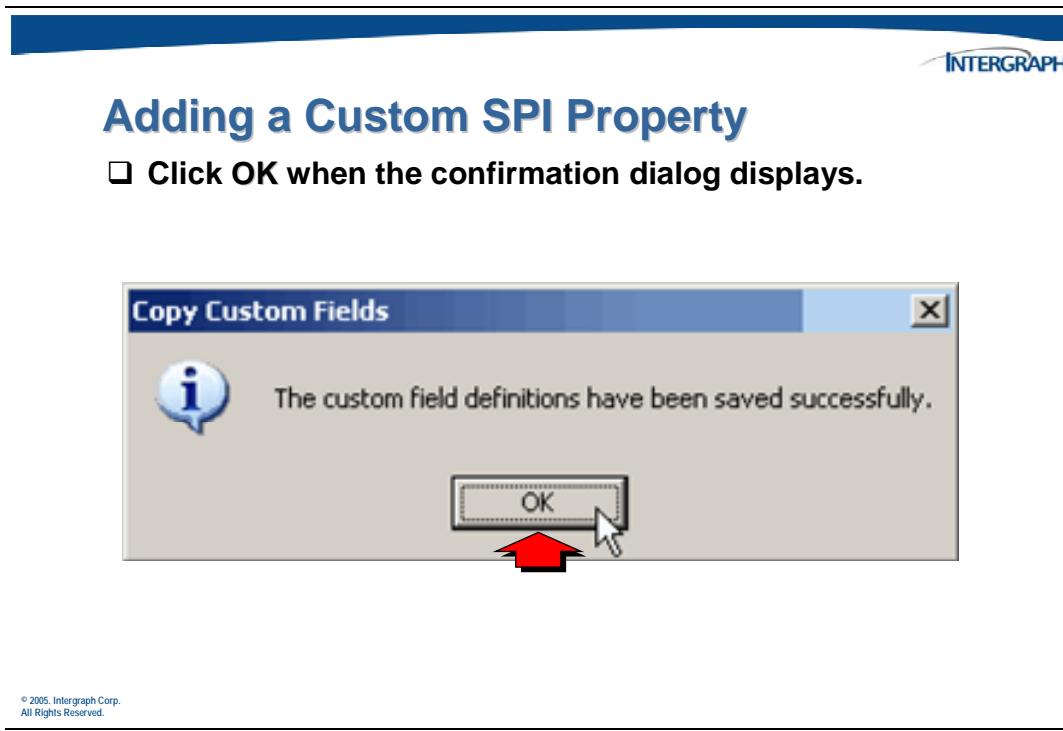
Use the **Fields** button to display the *Custom Fields* dialog where new UDF's can be added.



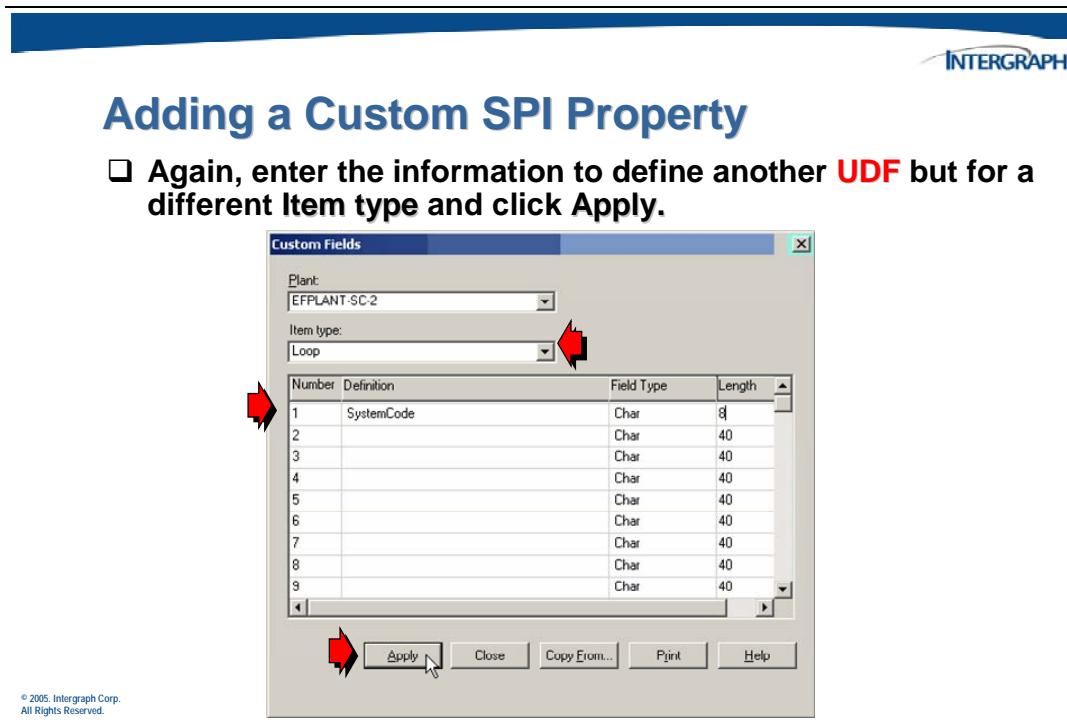
In the following example, the property *SystemCode* that was added to SPPID will be added as an SPI UDF for the **Instrument** class.



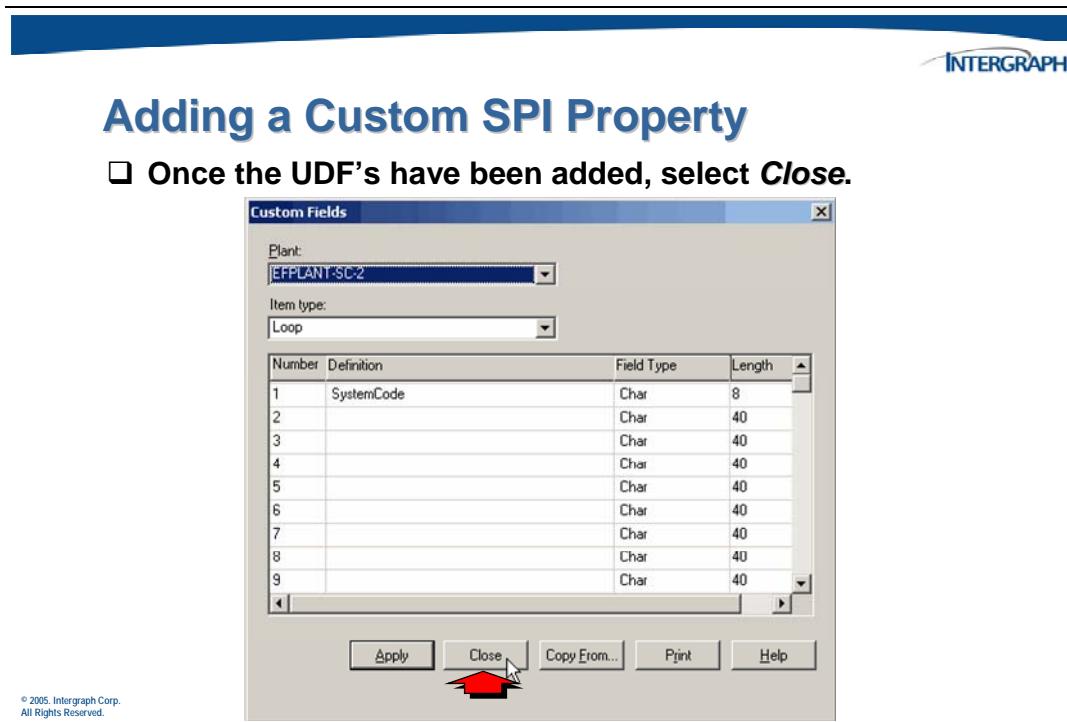
A confirmation dialog will display.



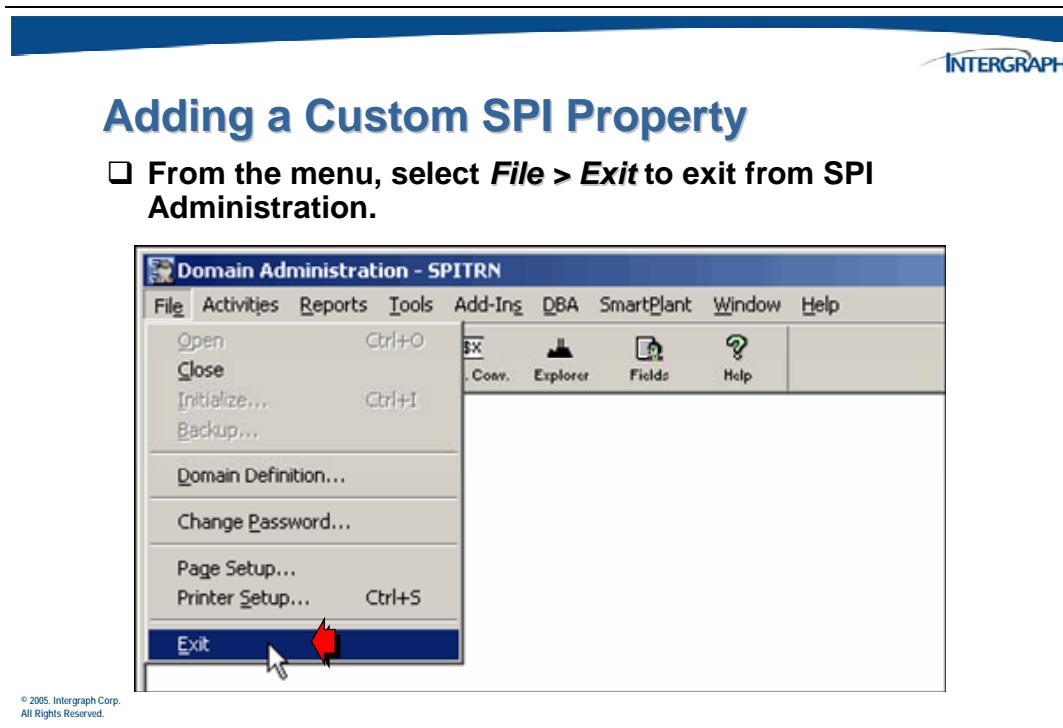
Repeat the process to add the *SystemCode* UDF to the Instrument **Loop** class.



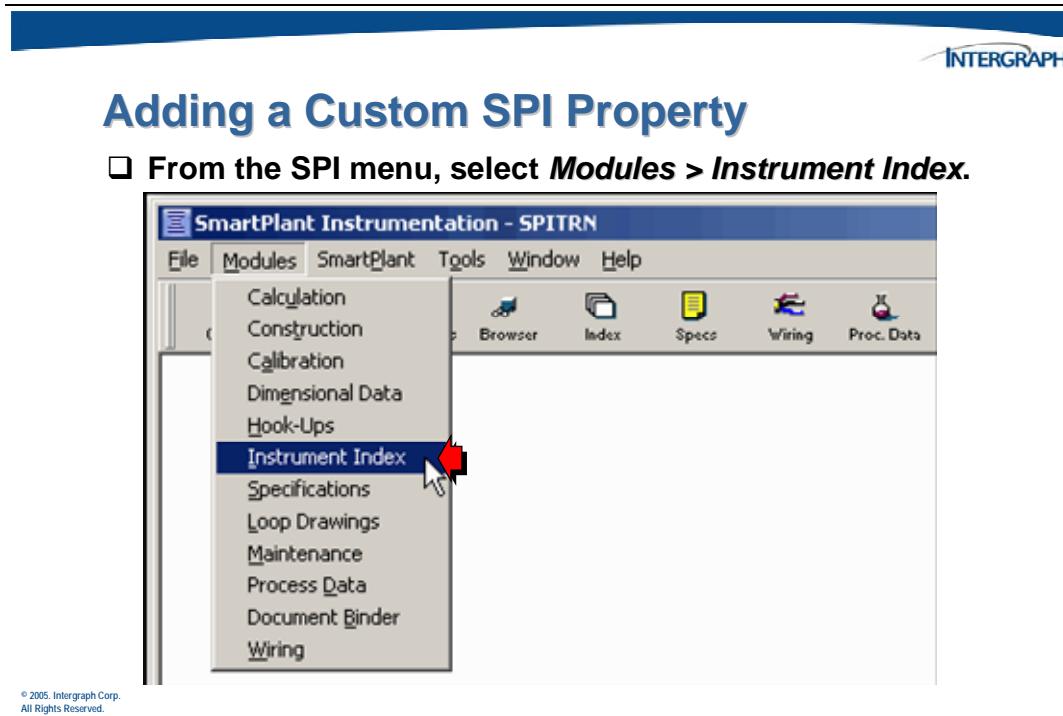
A confirmation dialog will once again display.



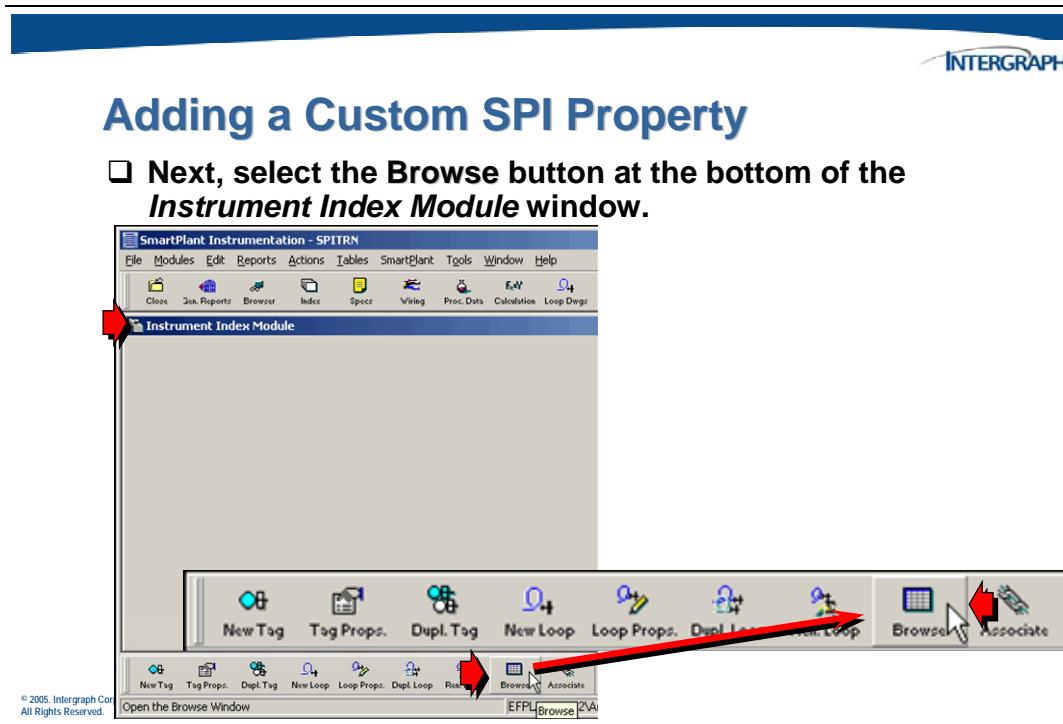
Now that the *SystemCode* UDF has been added, **exit** from the Administration module.



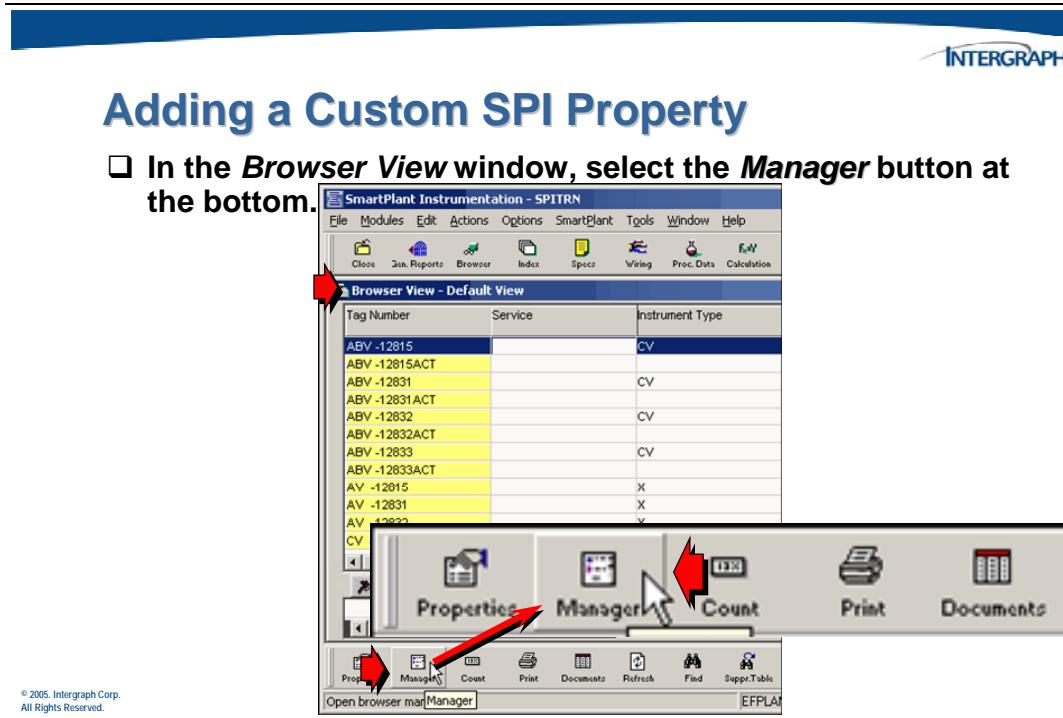
Next, start SmartPlant Instrumentation and configure the new field to be display in the *Instrument Index* browser.



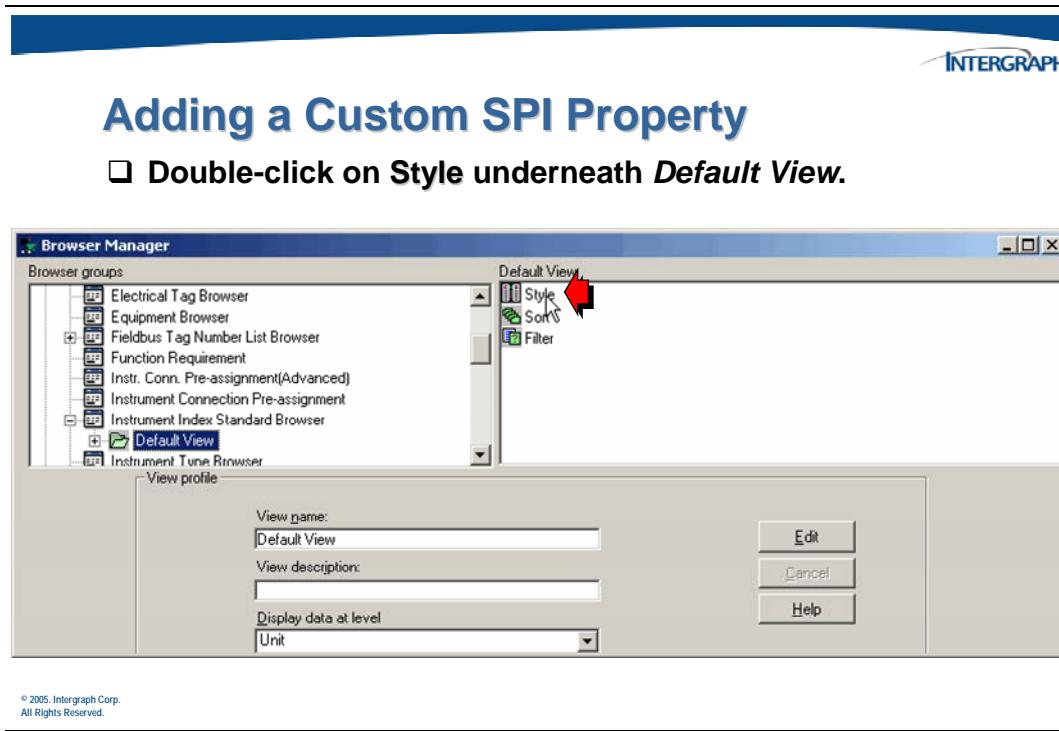
Display the *Instrument Index* browser in order to add the new UDF.



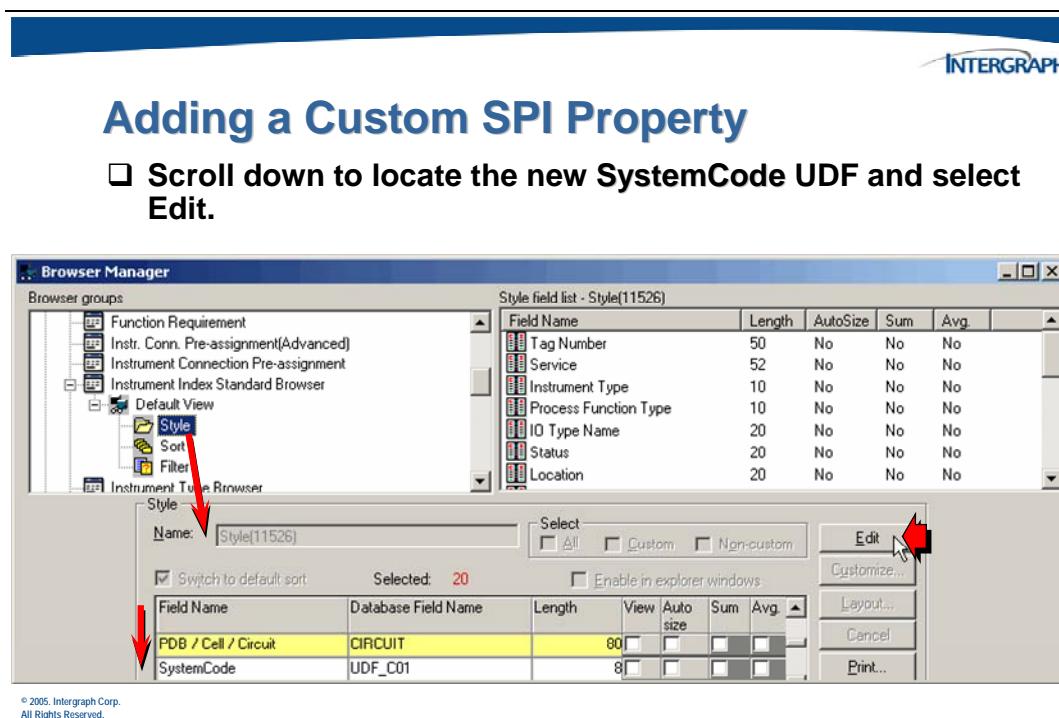
When the browser view displays, use the **Manager** button to change the default configuration.



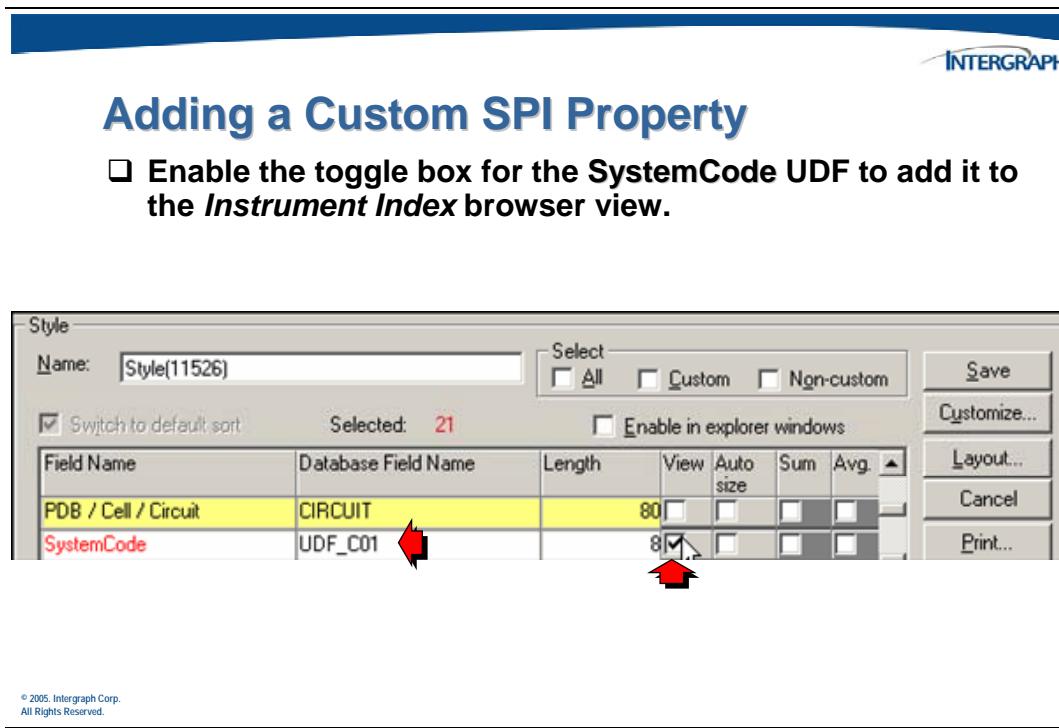
The *Browser Manager* window will display with the **Default View** for Instrument Index.



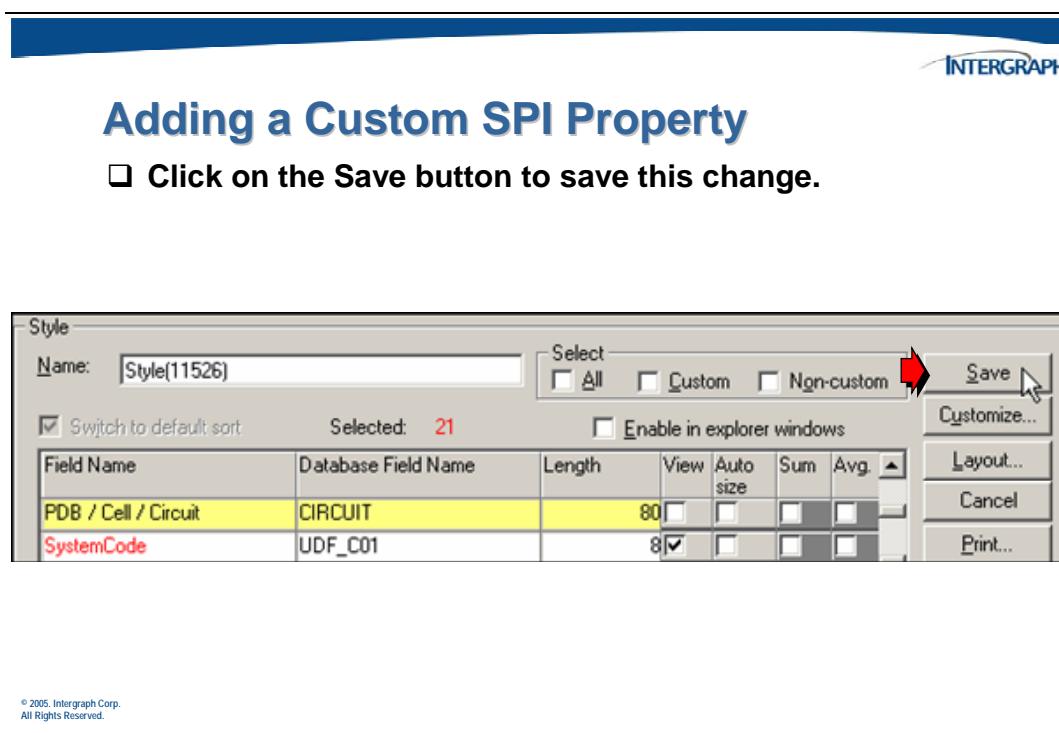
The available fields for this view will display in the bottom portion of the window.



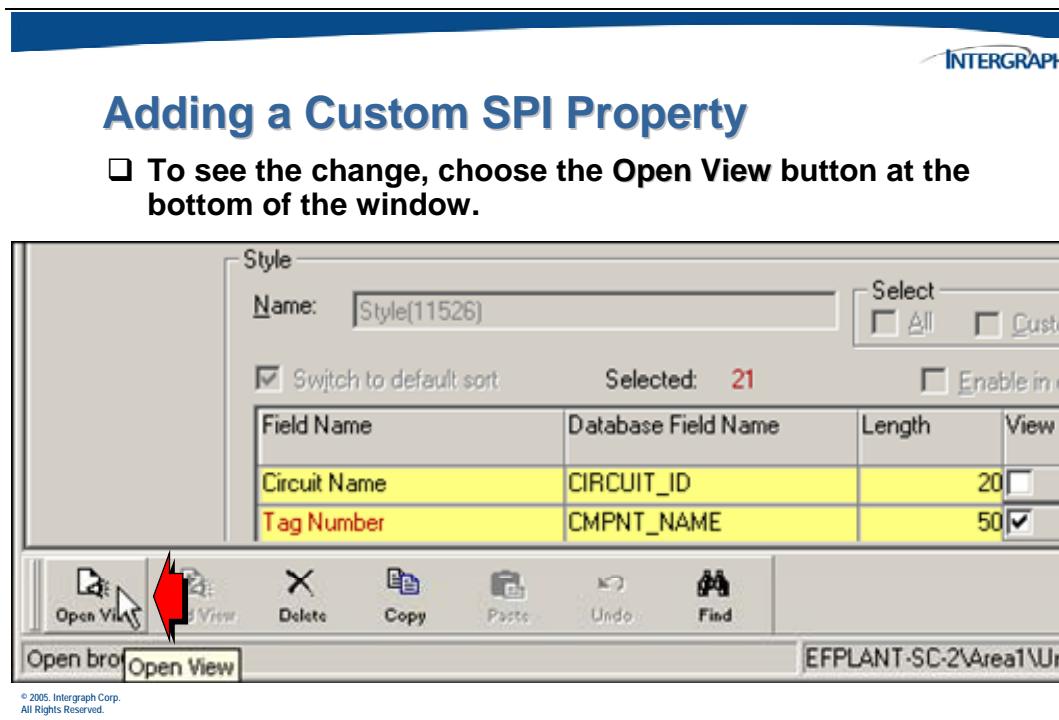
Locate the new **SystemCode** UDF and add it to the default Instrument Index browser view.



Once the UDF has been added to the view, save the change.

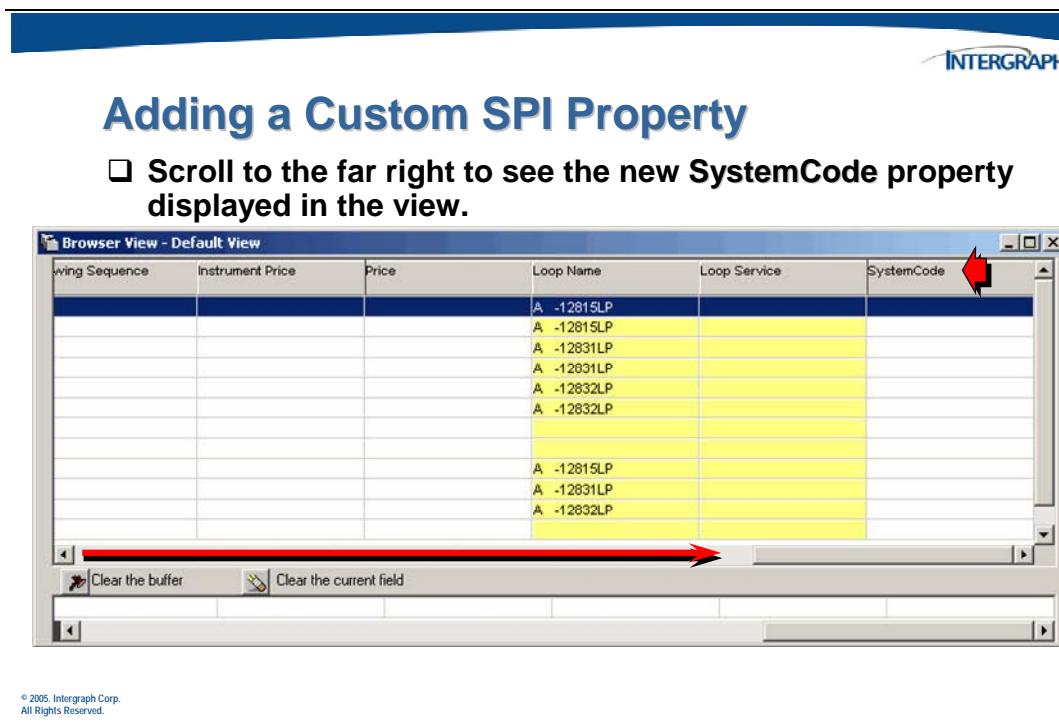


Once the change has been made and save, display the browser view and verify that the new UDF has been configured correctly.



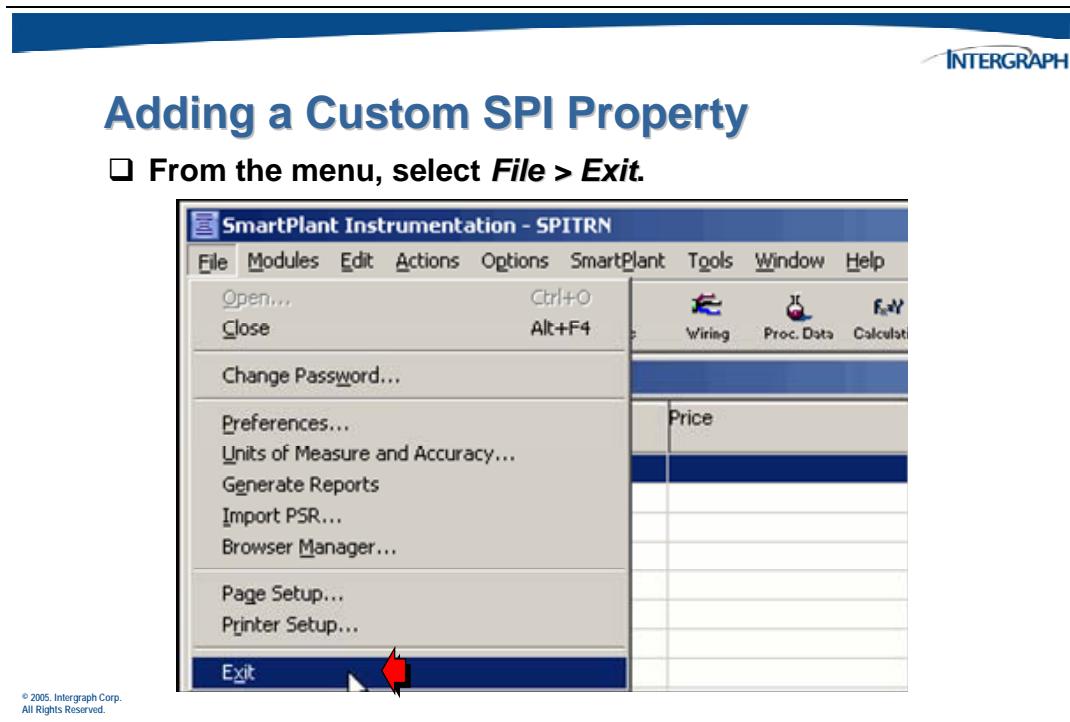
The screenshot shows the 'Style' configuration dialog box. In the 'Name' field, 'Style(11526)' is entered. Under the 'Selected' section, '21' is highlighted. A red arrow points to the 'Open View' button at the bottom left of the toolbar. The 'View' tab is selected. The table lists two properties: 'Circuit Name' (Database Field Name: CIRCUIT_ID, Length: 20) and 'Tag Number' (Database Field Name: CMPNT_NAME, Length: 50). The 'Tag Number' row has a checked checkbox in the 'Enable in e...' column. The toolbar includes icons for Open View, View, Delete, Copy, Paste, Undo, and Find.

The *Instrument Index Browser View* will display including the new property.



The screenshot shows the 'Browser View - Default View'. The table has six columns: Wing Sequence, Instrument Price, Price, Loop Name, Loop Service, and SystemCode. A red arrow points to the 'SystemCode' column header. Another red arrow points to the right edge of the table, indicating the scroll bar. The table contains several rows of data, mostly yellowed. At the bottom, there are buttons for 'Clear the buffer' and 'Clear the current field'.

Exit from SPI and perform the SmartPlant mapping.



10.9.1 Mapping a Custom Property in SPI

Before defining mapping for the new UDF in the tool map schema, the new SystemCode property must be added. Remember, this is done automatically when the tool map schema and the SmartPlant schema are synchronized. This needs to be done in “connected” mode using the schema editor.

Map the Property in the Tool Schema

- ❑ Select the tool map schema to be loaded and click Finish.



A list of available tool map schemas will be displayed. Select the tool map schema for SmartPlant Instrumentation.

The differences between the two map schemas are displayed in the **Synchronize** form.

Class	Parent Name	Name	Property	SPI Database
SPMapPropertyDef	lInstrumentUDF	SystemCode	Name	Set value to "UDF_C01"
SPMapPropertyDef	lLoopUDF	SystemCode	Name	Set value to "LOOP_UDF_C01"

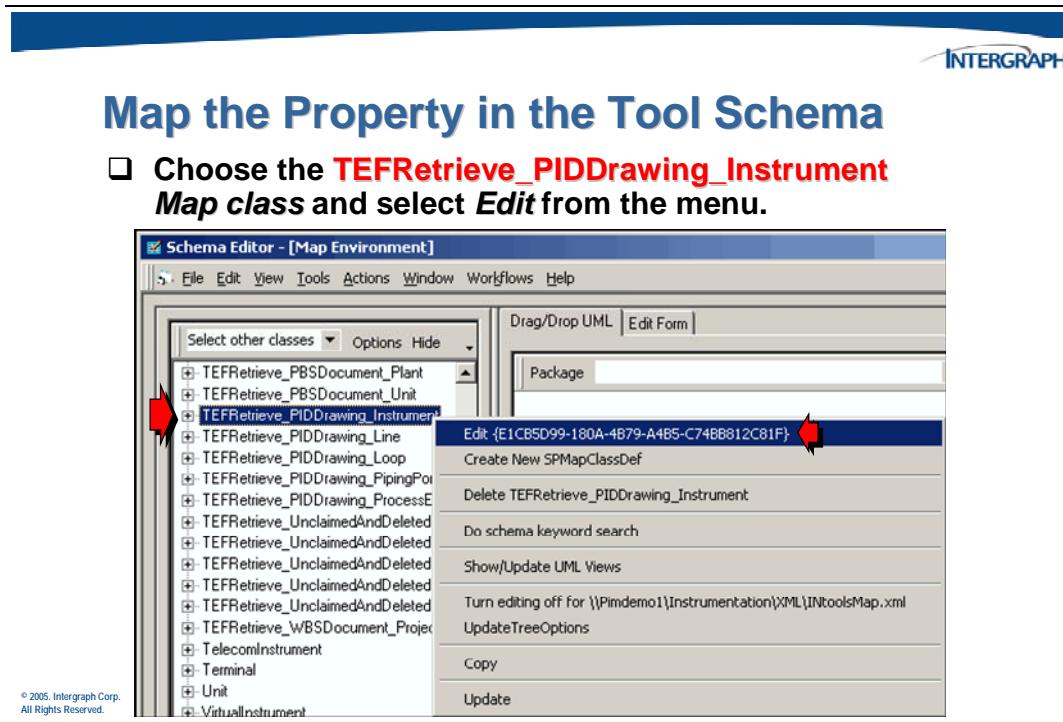
© 2005, Intergraph Corp.
All Rights Reserved.

The property “SystemCode” is automatically added to the tool map schema. Because the property has already been added to the extension schema (ToolExten.xml) all that remains to be done is the mapping.

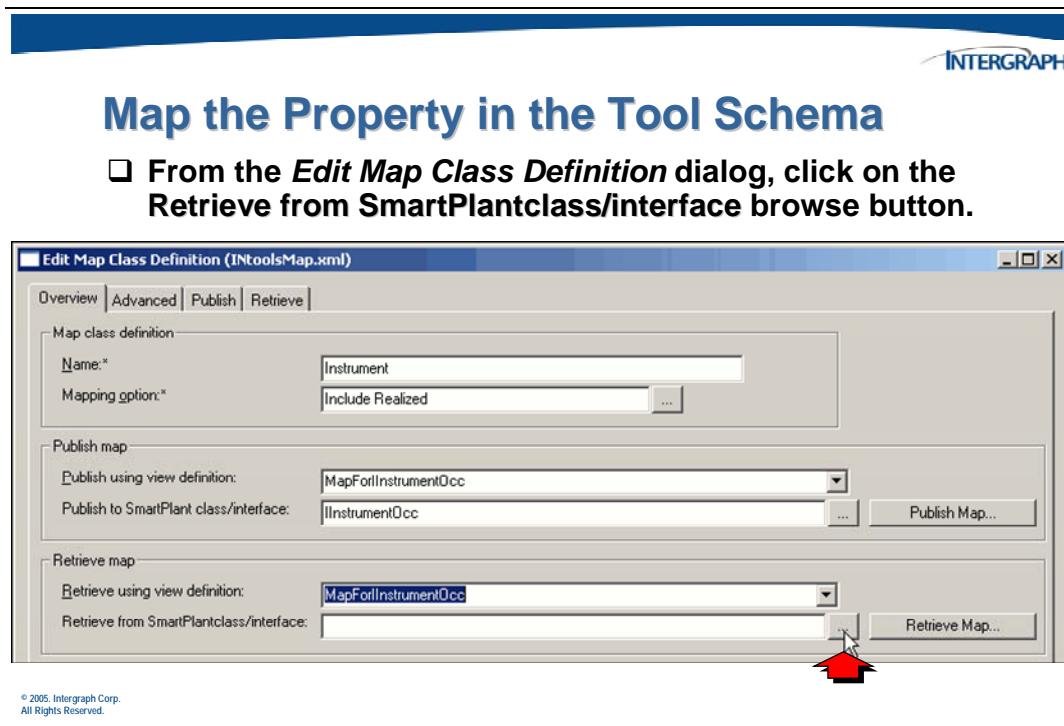
© 2005, Intergraph Corp.
All Rights Reserved.

10.9.2 Retrieve Mapping for a Custom Property

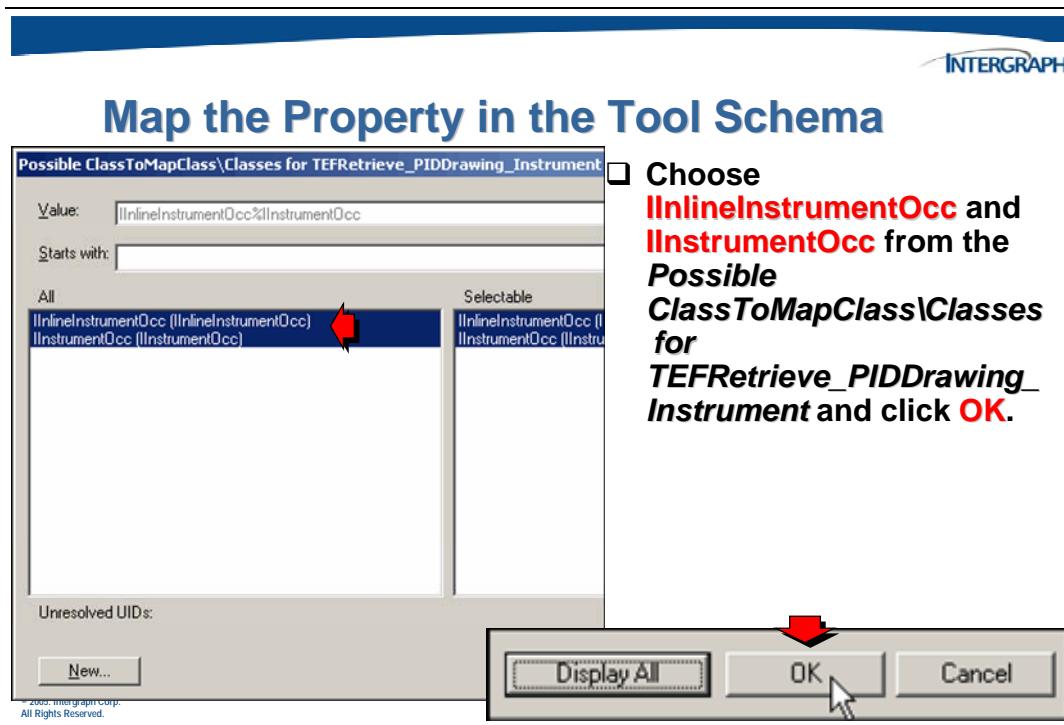
The SPI tool adapter uses special **TEFPublish** and **TEFRetrieve** map classes.



The *Edit Map Enumerated List Definition* dialog will display.



Add the necessary interfaces that expose the new property in order to configure retrieval mapping.

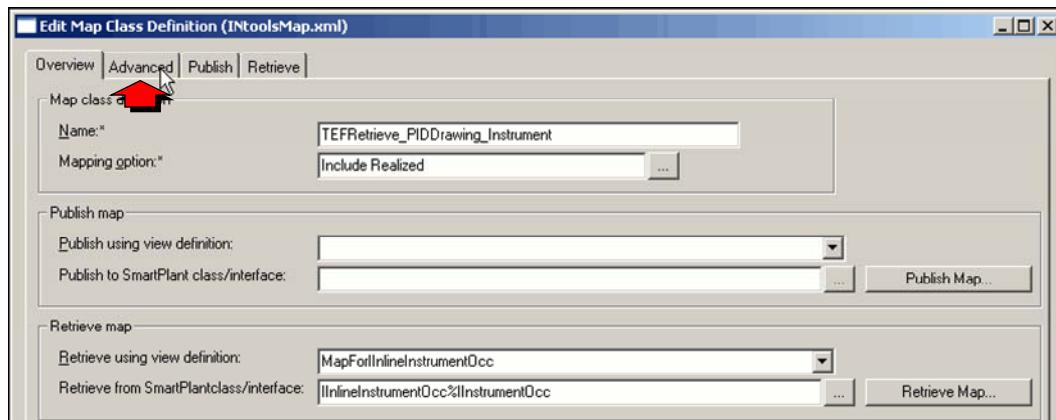


Next, expose the UDF containing the custom property.



Map the Property in the Tool Schema

- From the *Edit Map Class Definition* dialog, click on the Advanced tab.



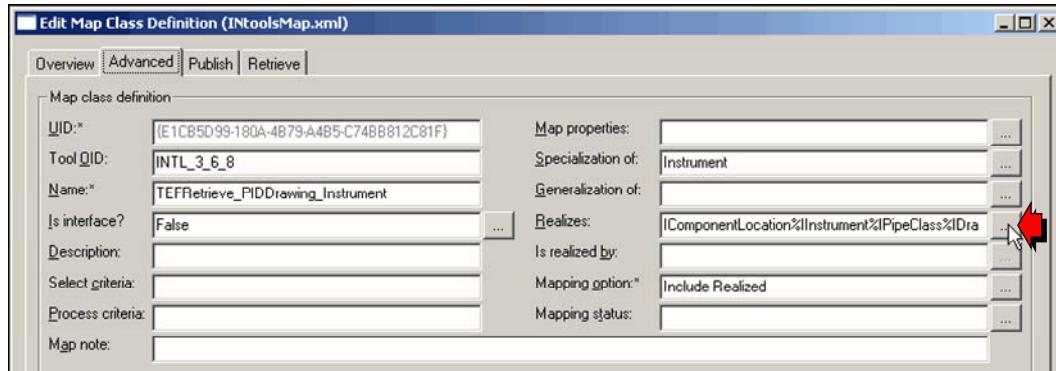
© 2005. Intergraph Corp.
All Rights Reserved.

The map class will need to realize the IInstrumentUDF map class, which exposes the custom property.



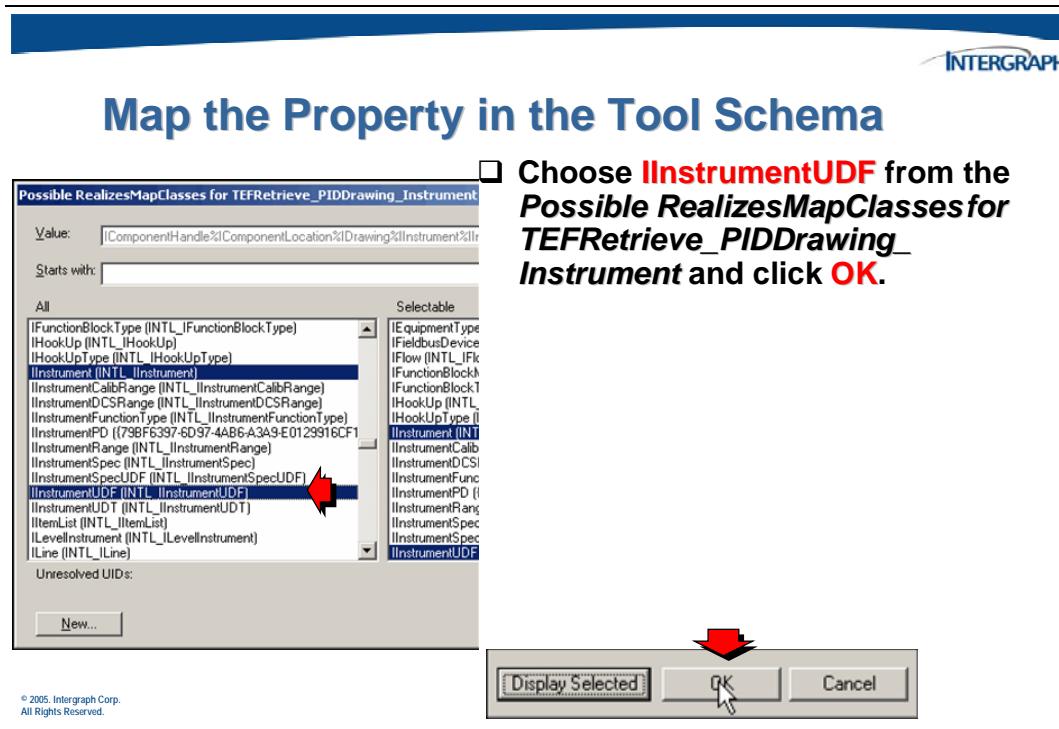
Map the Property in the Tool Schema

- Click the browse (...) button to the right of the *Realizes:* field.

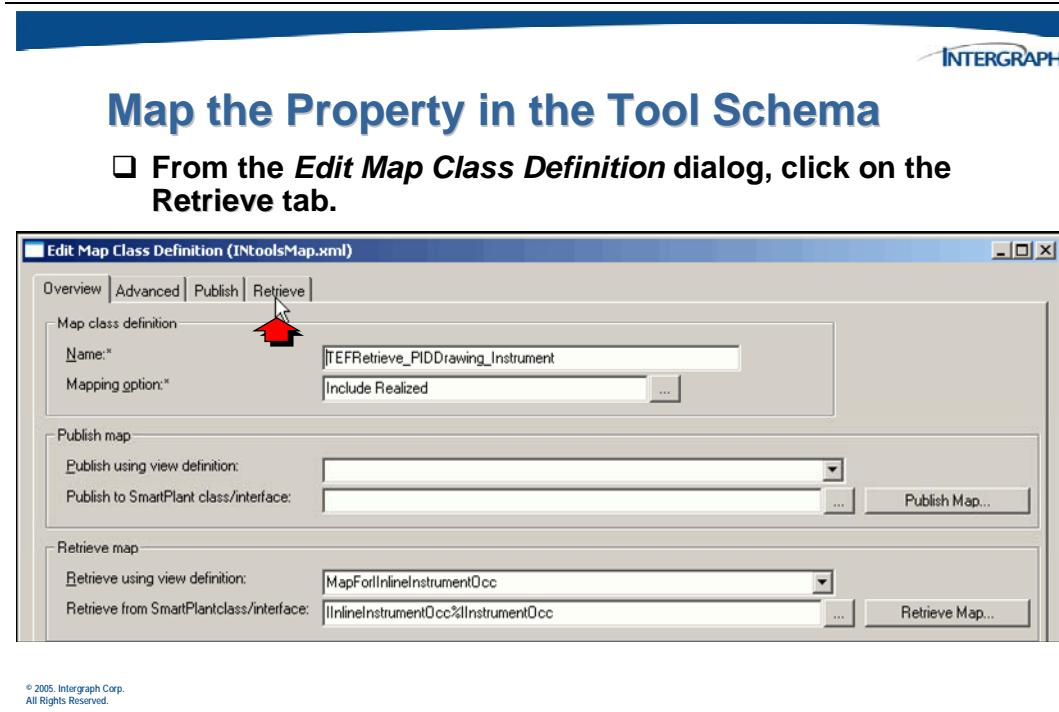


© 2005. Intergraph Corp.
All Rights Reserved.

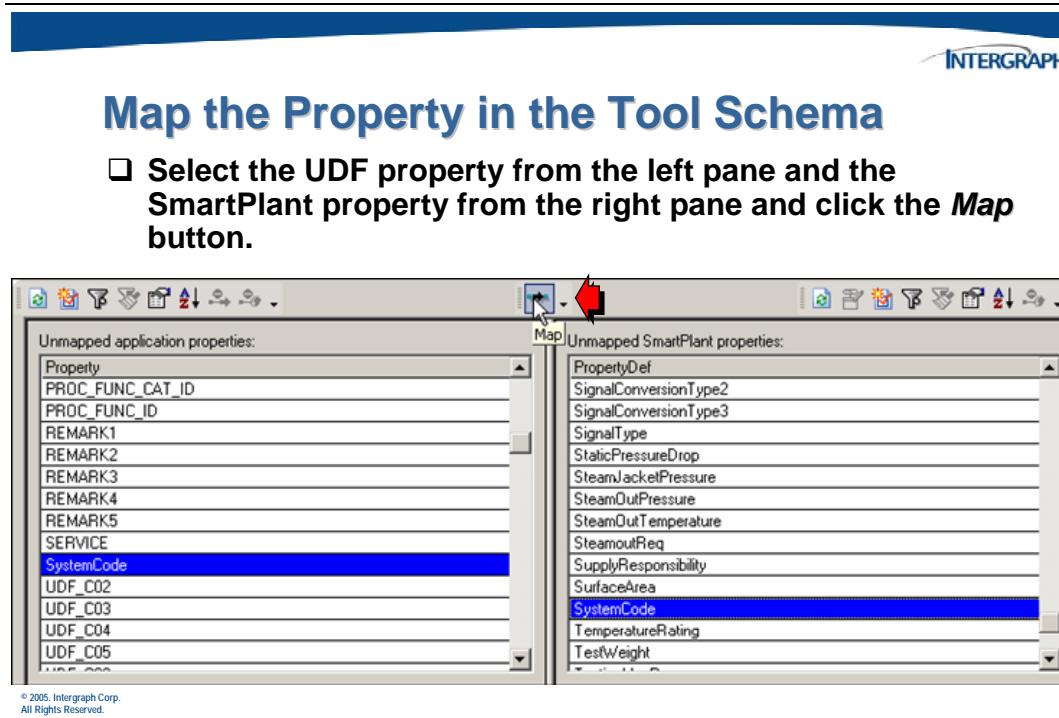
A list of possible realizes map classes will be displayed.



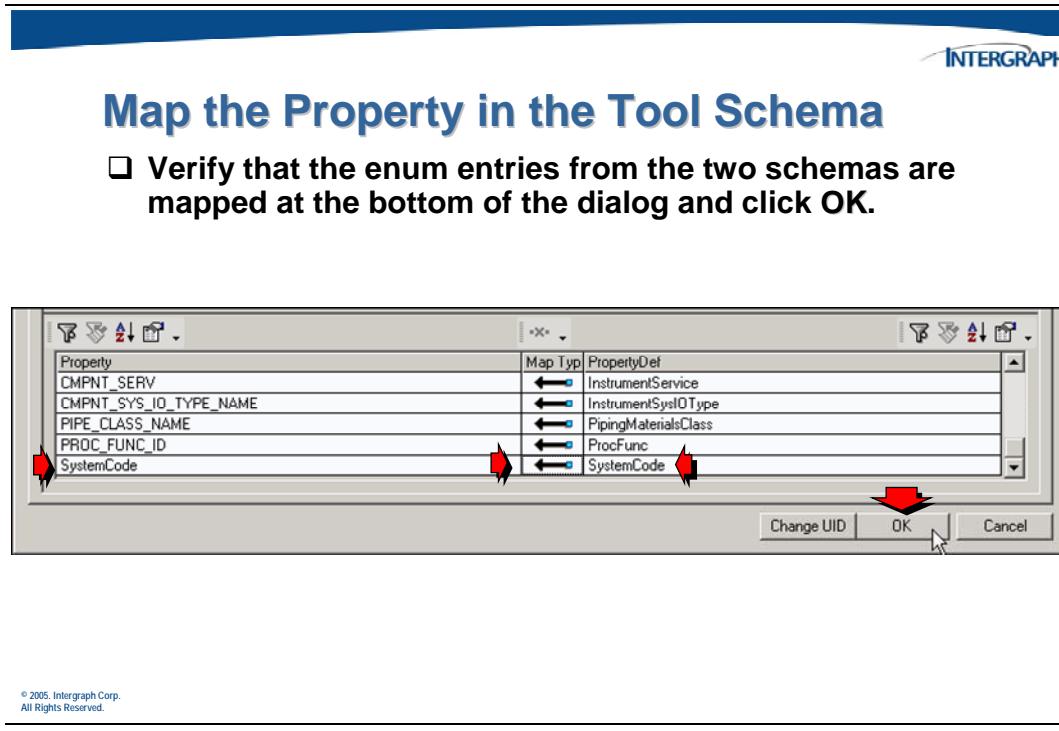
Once the property has been exposed by adding the required interface, perform the mapping.



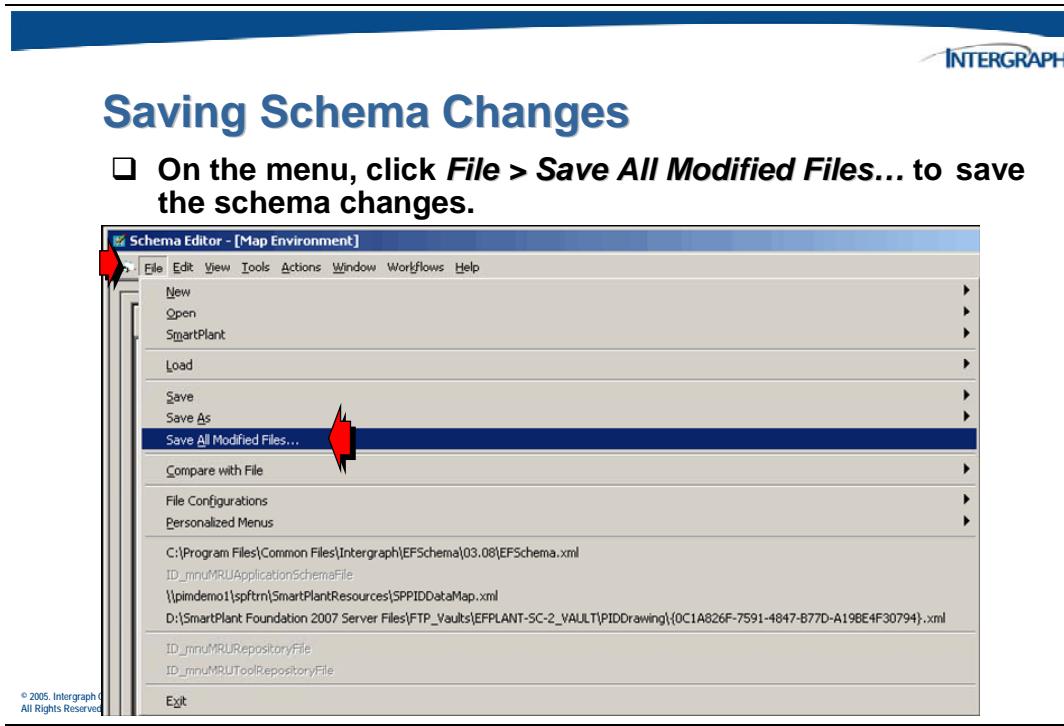
The **Map** button is used to map an unmapped tool property to an unmapped SmartPlant property.



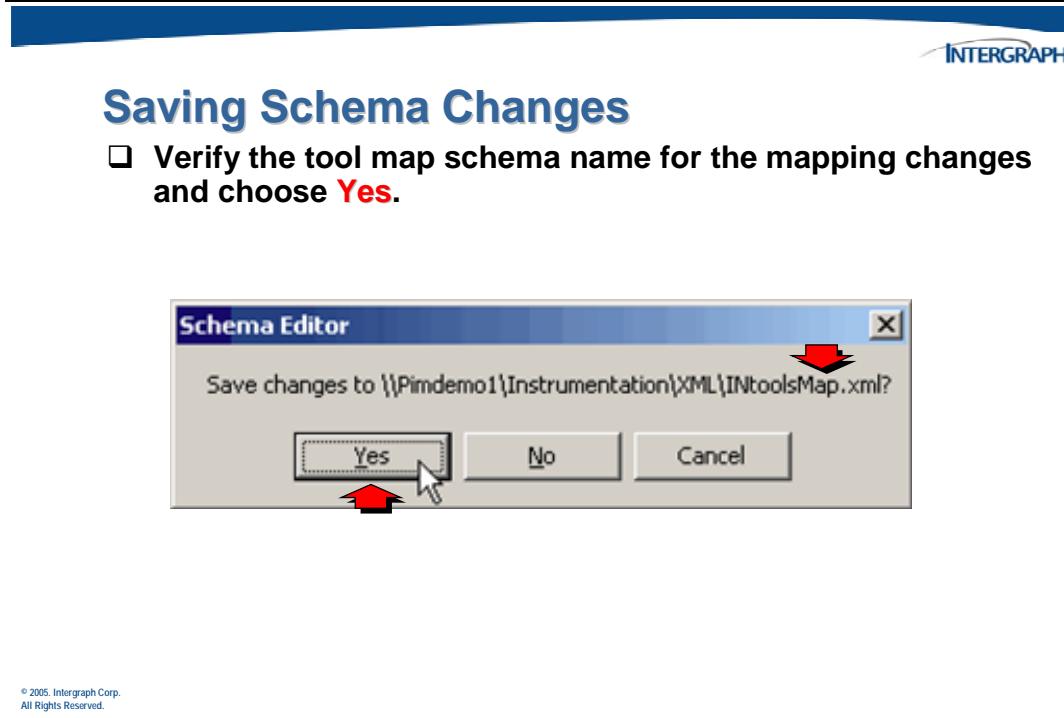
This completes the mapping for the new custom property for **Retrieve** operations.



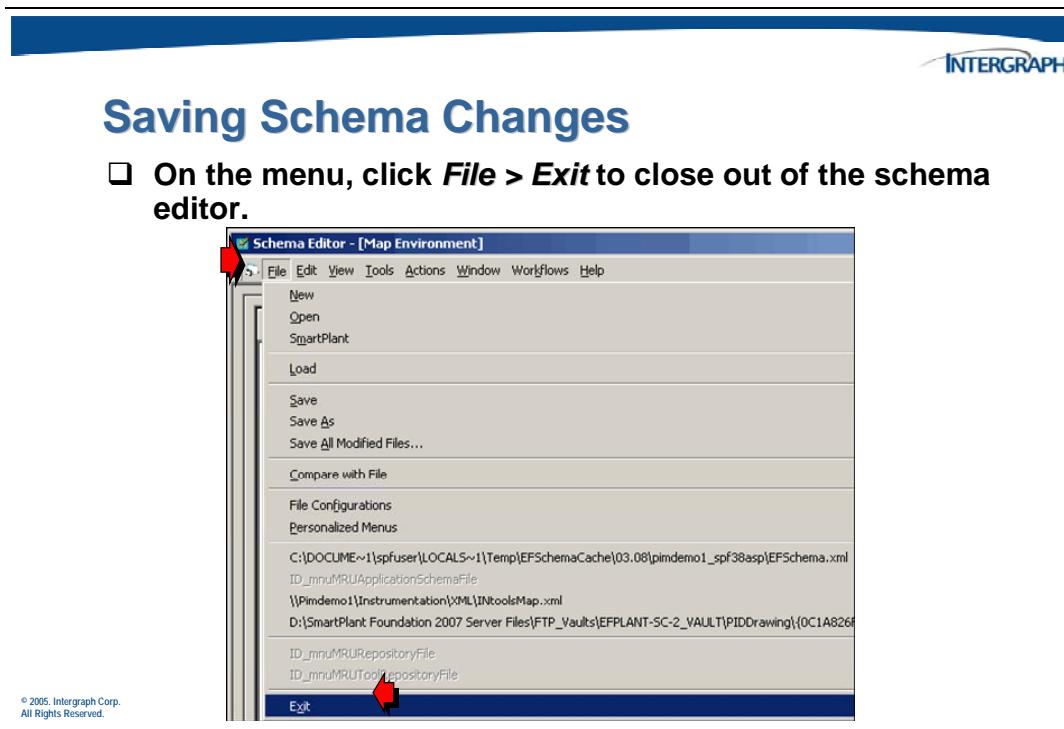
All of the mapping information is stored in memory will need to be saved to the respective xml files.



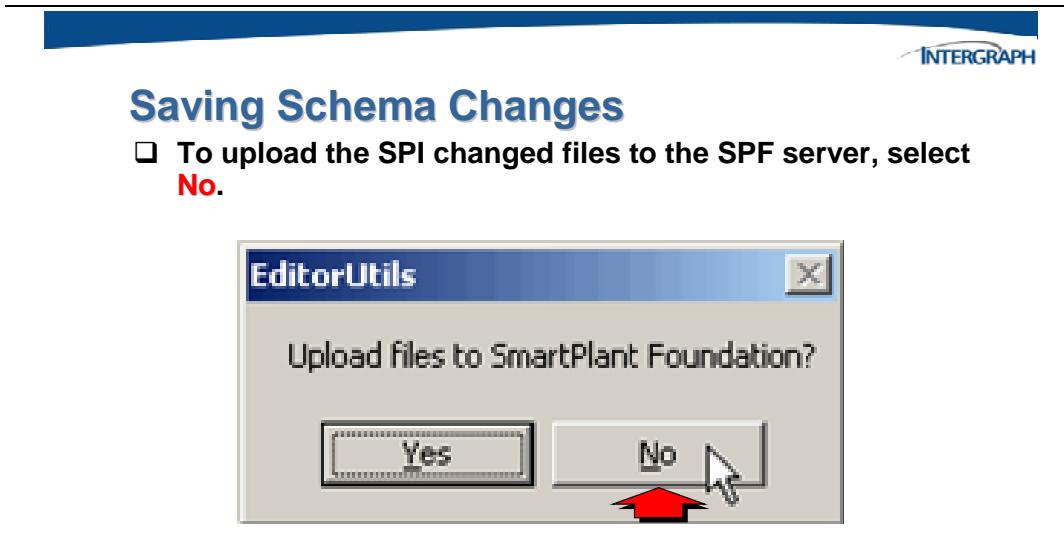
You will be prompted to save the changes to the tool map schema.



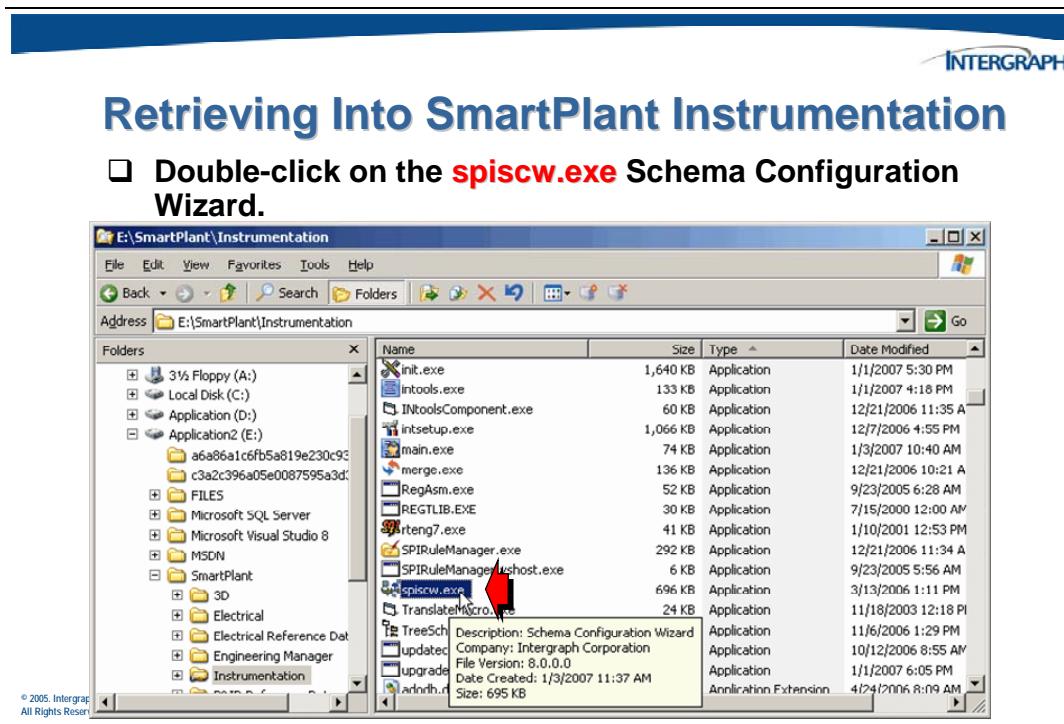
Once all of the schema files have been saved exit the Schema Editor.



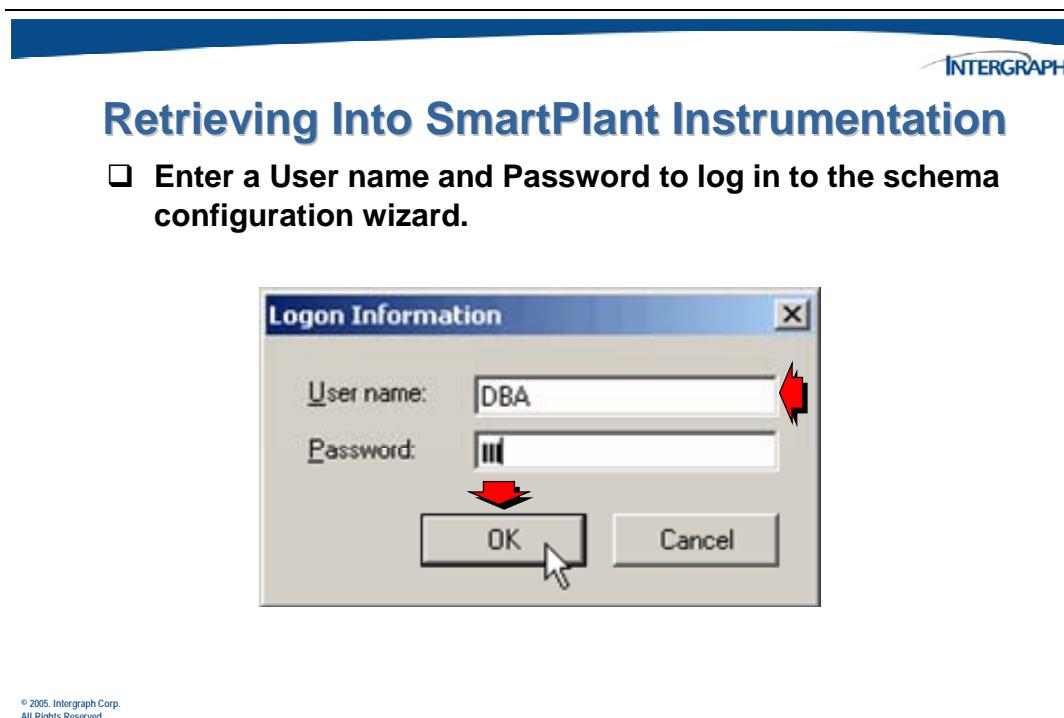
Since no changes were made to the SmartPlant schema, there is no need to perform an upload operation. This is because the SystemCode property already existed in the schema extension from the work done earlier for SPPID.



In order to perform a retrieve, run the SPI Schema Configuration Wizard, which is located in the **\SmartPlant\Instrumentation** product folder.



A login dialog will display.



In order to perform a retrieve, the SPI Configuration Wizard allows you to realize a new interface but within the SPI database.



Retrieving Into SmartPlant Instrumentation

- When the Welcome screen displays, click Next.

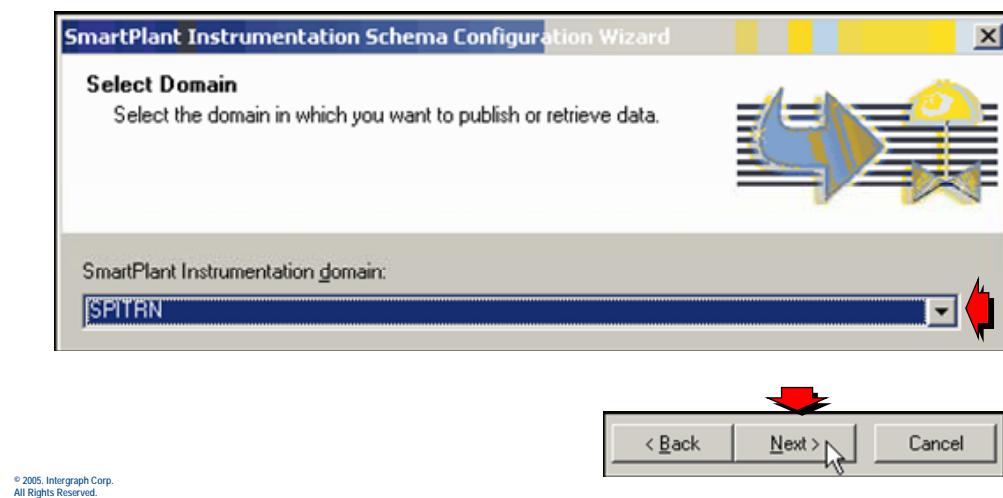


First, select the domain within SPI where data will be retrieved.

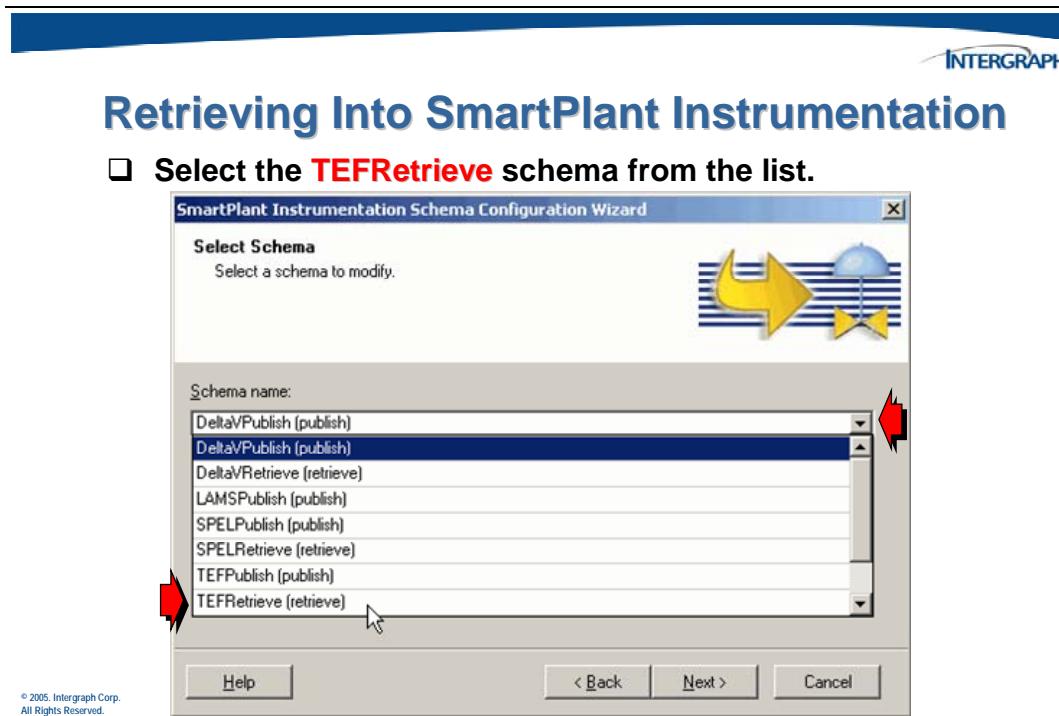
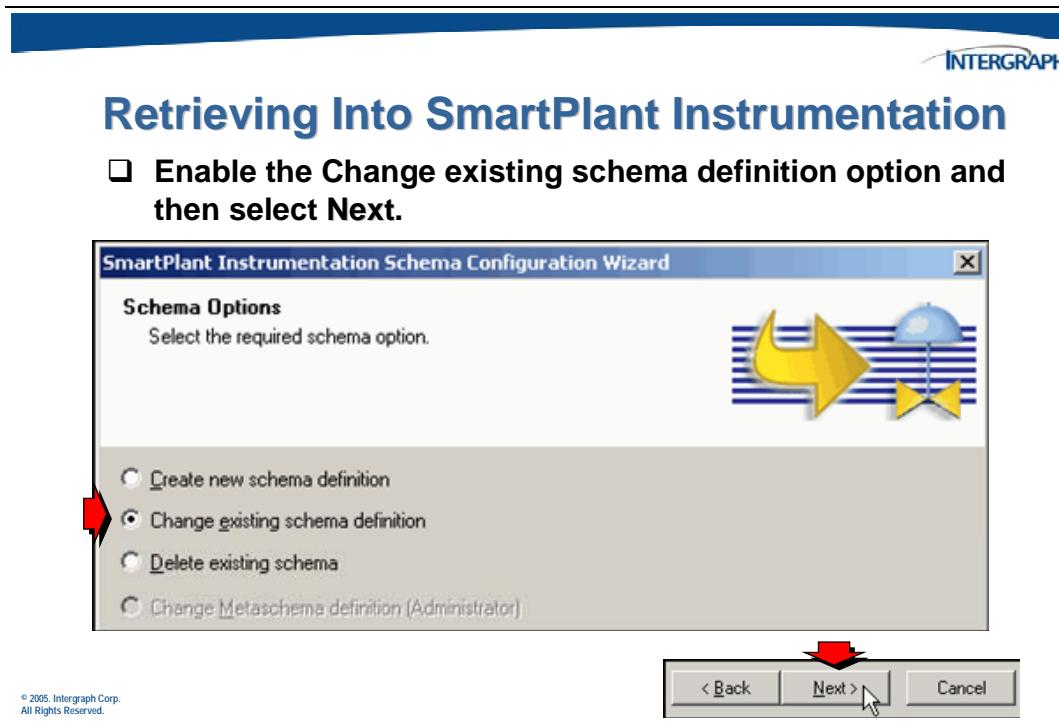


Retrieving Into SmartPlant Instrumentation

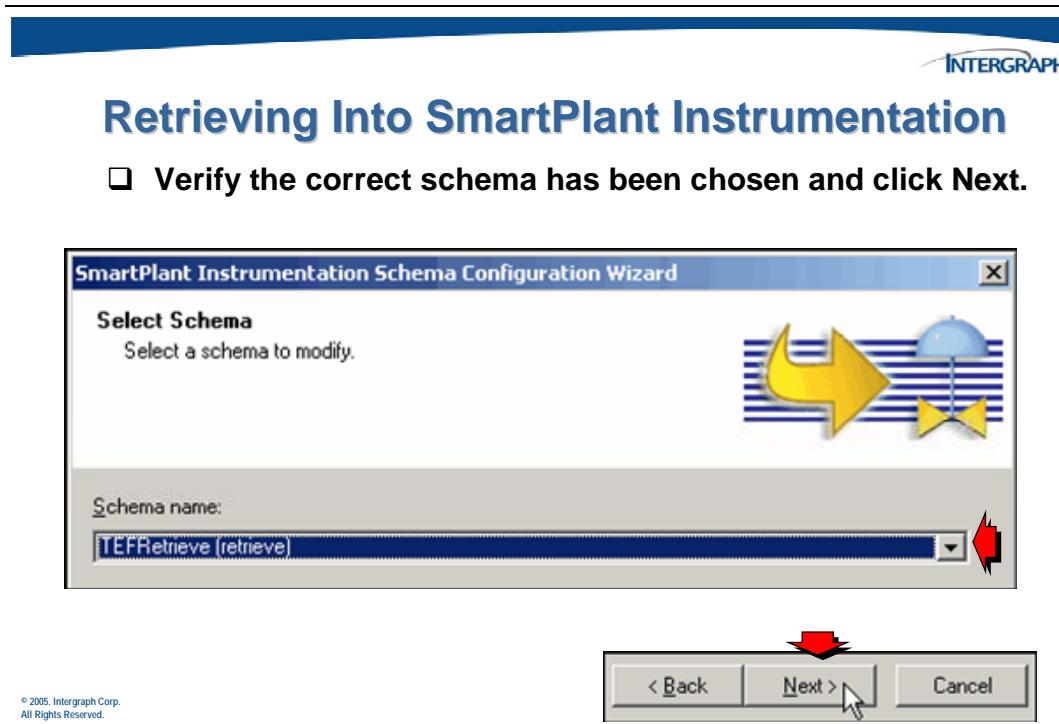
- Select the correct domain and click Next.



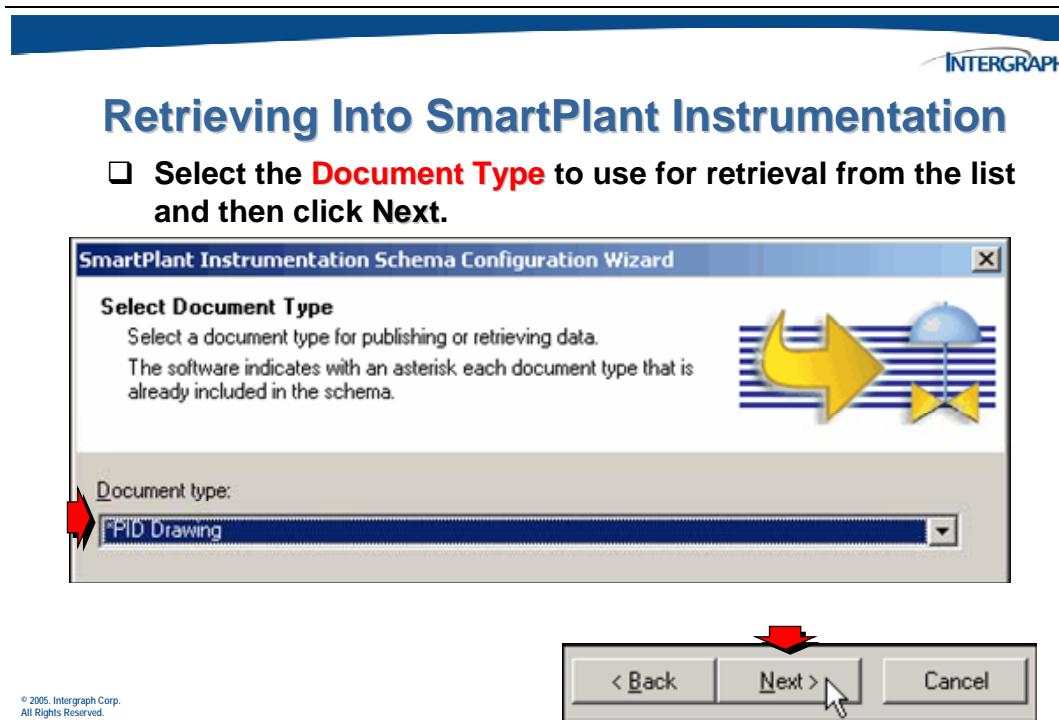
The *Schema Options* screen will allow for modifications to an existing schema in the SPI database.



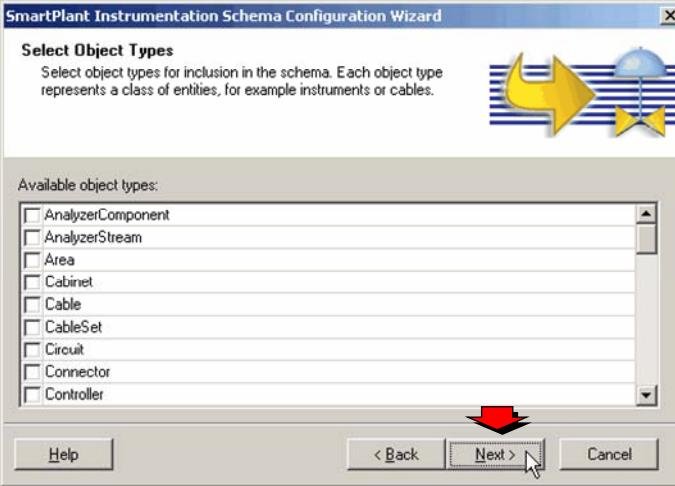
Specify which schema within the SPI database will be changed.



Next, select the kind of information that is to be retrieved.

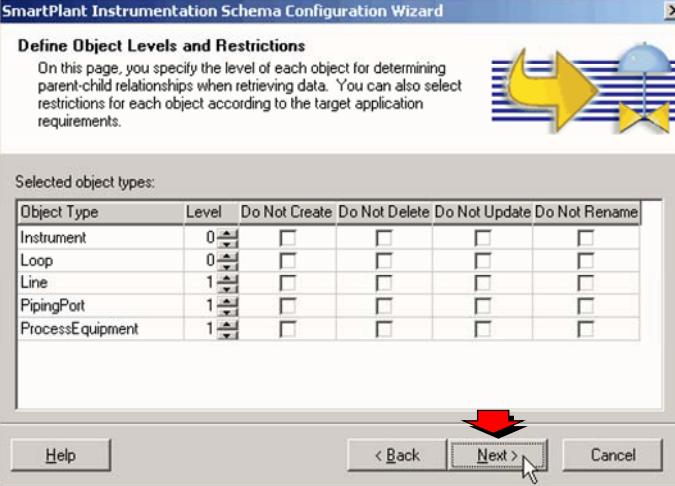


Since the necessary objects to be retrieved are already in the tool map schema file, no additional types (classes) need to be selected.



The screenshot shows the "Select Object Types" step of the SmartPlant Instrumentation Schema Configuration Wizard. The window title is "SmartPlant Instrumentation Schema Configuration Wizard". The sub-title "Select Object Types" is displayed, along with a note: "Select object types for inclusion in the schema. Each object type represents a class of entities, for example instruments or cables." Below this is a decorative graphic of three yellow arrows pointing right. A scrollable list titled "Available object types:" contains a long list of checkboxes, including AnalyzerComponent, AnalyzerStream, Area, Cabinet, Cable, CableSet, Circuit, Connector, and Controller. At the bottom of the window are buttons for "Help", "< Back", "Next >" (which has a red arrow pointing to it), and "Cancel". A copyright notice at the bottom left reads: "© 2005, Intergraph Corp. All Rights Reserved."

The object levels are used for new relationships if new types have been added to the schema. Retrieve uses the existing relationships so take the defaults.



The screenshot shows the "Define Object Levels and Restrictions" step of the SmartPlant Instrumentation Schema Configuration Wizard. The window title is "SmartPlant Instrumentation Schema Configuration Wizard". The sub-title "Define Object Levels and Restrictions" is displayed, with a note: "On this page, you specify the level of each object for determining parent-child relationships when retrieving data. You can also select restrictions for each object according to the target application requirements." Below this is a decorative graphic of three yellow arrows pointing right. A scrollable list titled "Selected object types:" shows a table of object types and their levels, along with checkboxes for various restrictions. The table data is as follows:

Object Type	Level	Do Not Create	Do Not Delete	Do Not Update	Do Not Rename
Instrument	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Loop	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Line	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PipingPort	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ProcessEquipment	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

At the bottom of the window are buttons for "Help", "< Back", "Next >" (which has a red arrow pointing to it), and "Cancel". A copyright notice at the bottom left reads: "© 2005, Intergraph Corp. All Rights Reserved."

The filter conditions are used for new object types. Again, retrieve uses the existing so take the defaults.



Retrieving Into SmartPlant Instrumentation

- No changes are required for any of the filter conditions; just select Next.

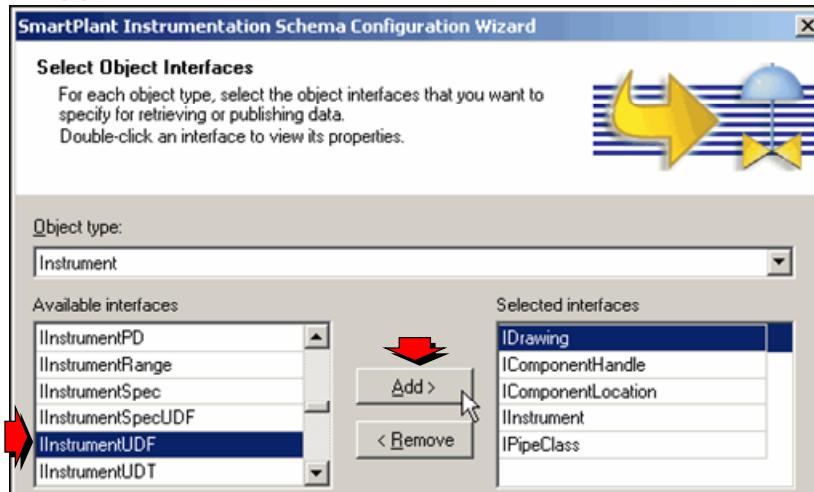


Add the **IInstrumentUDF** interface to the list of existing realized interfaces.



Retrieving Into SmartPlant Instrumentation

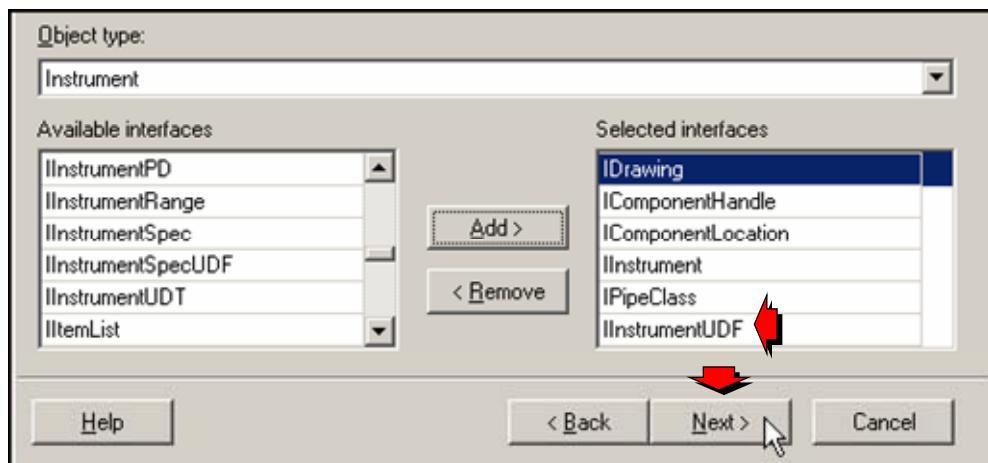
- Select the **IInstrument** interface from the available list and Add> it to the selected interfaces list.





Retrieving Into SmartPlant Instrumentation

- Verify the interface has been added and click Next.



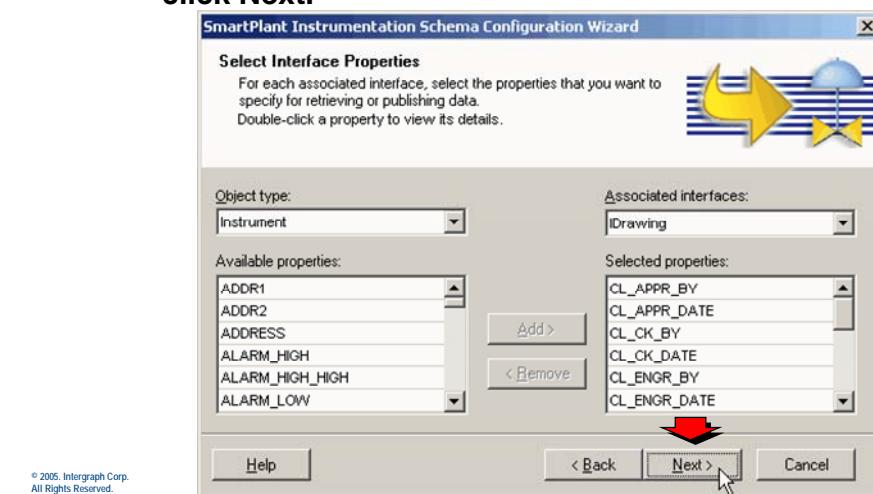
© 2005. Intergraph Corp.
All Rights Reserved.

The list of properties on the right are exposed by the interfaces that have been realized. The properties in the list on the left are the properties exposed by the unselected interfaces.



Retrieving Into SmartPlant Instrumentation

- No additional properties are required to be selected; just click Next.



If you choose **IIInstrumentUDF** in the *Associated interfaces* list, the exposed properties for this interface would display in the *Selected properties* list.



Retrieving Into SmartPlant Instrumentation

- ❑ Once all of the needed schema definition changes have been configured, click **Finish**.

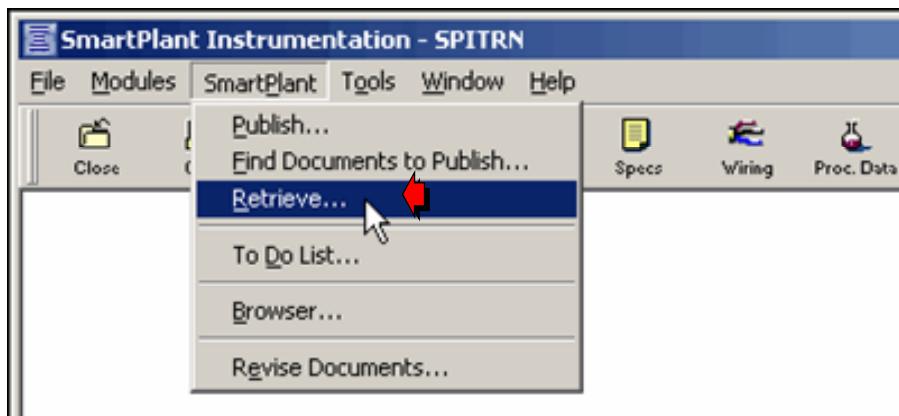


10.9.3 Testing Custom Property Retrieval

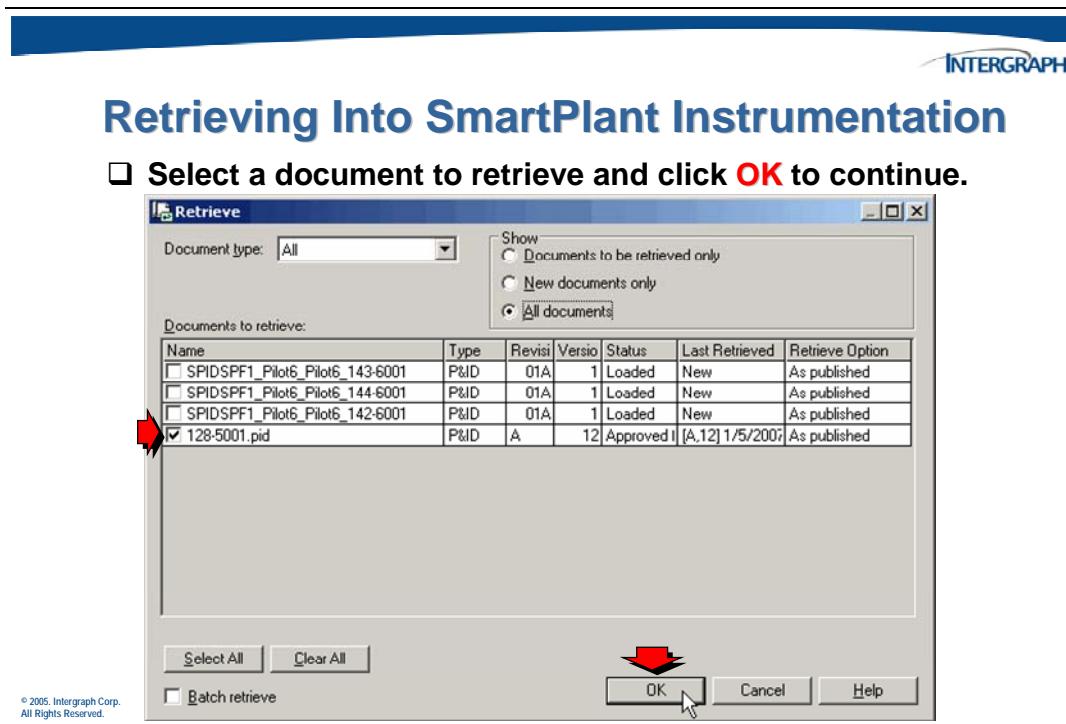
To perform a retrieve operation, use the **Retrieve** command from SmartPlant Instrumentation.

Retrieving Into SmartPlant Instrumentation

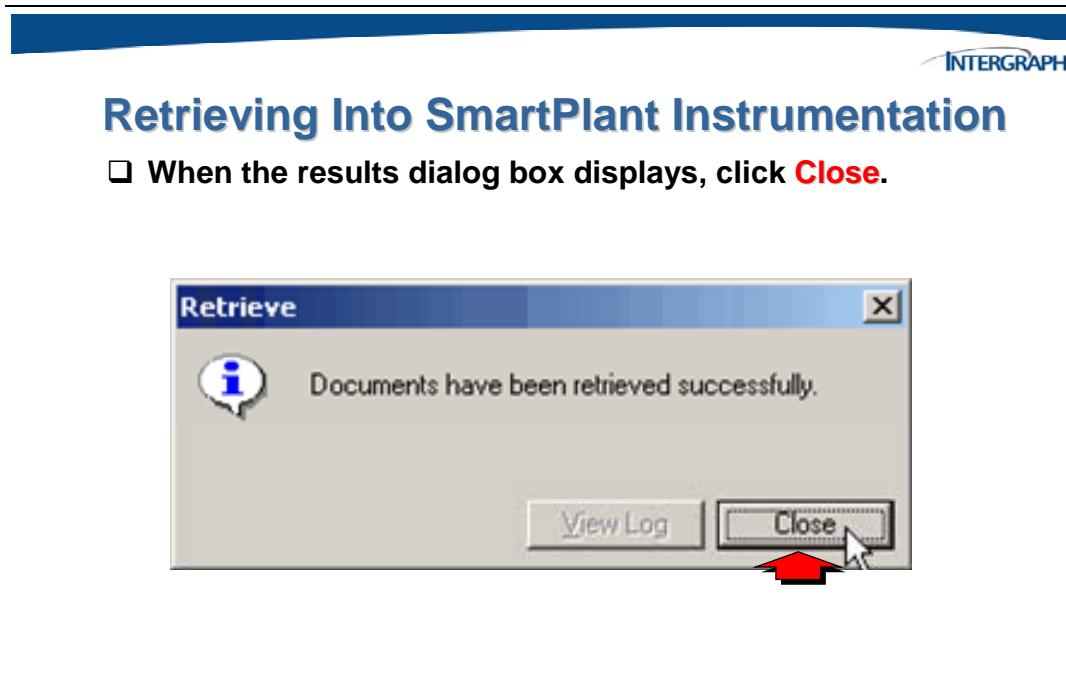
- Select the **SmartPlant > Retrieve...** menu command.



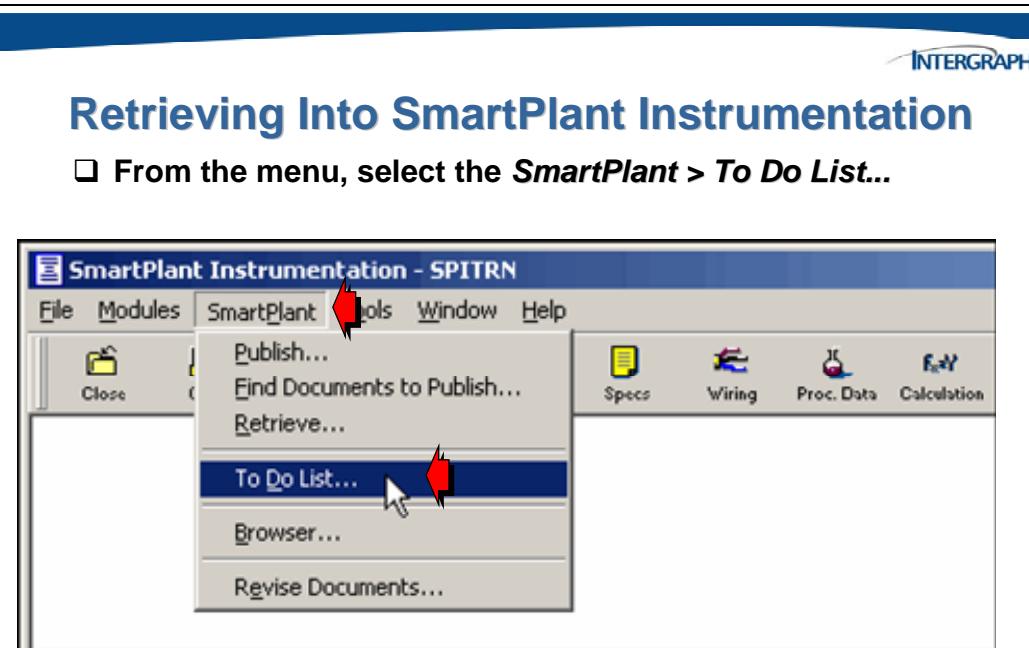
When the retrieve dialog displays, enable the toggle box next to the previously published documents (from SPPID) to be retrieved by SPI.



A dialog box will display after some time to report success or failure of the publish.

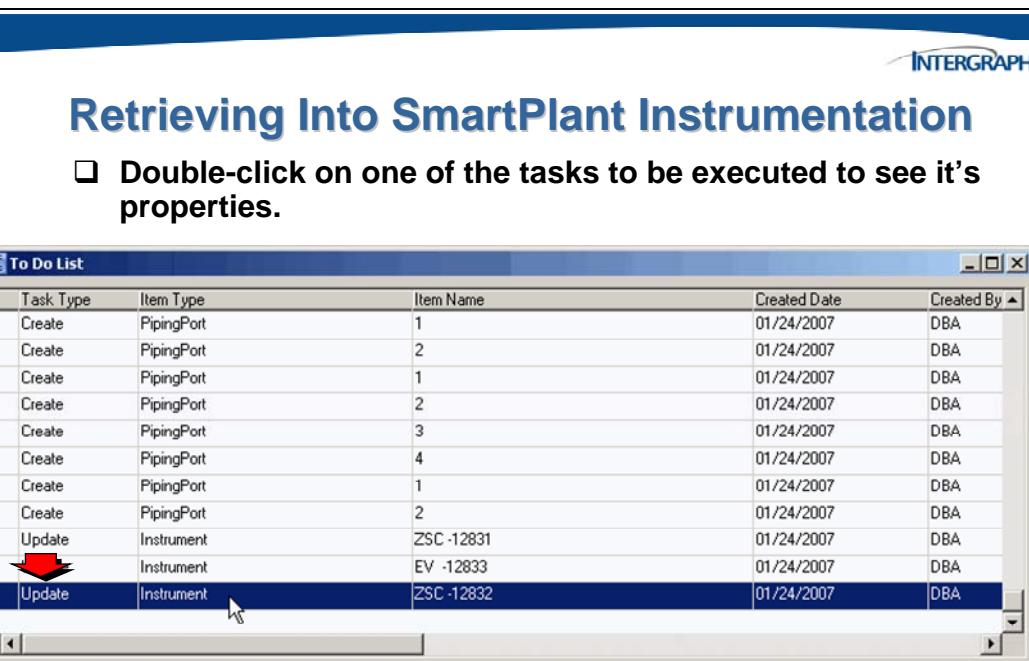


To see the changes to be made to SPI, open the *To Do List*.

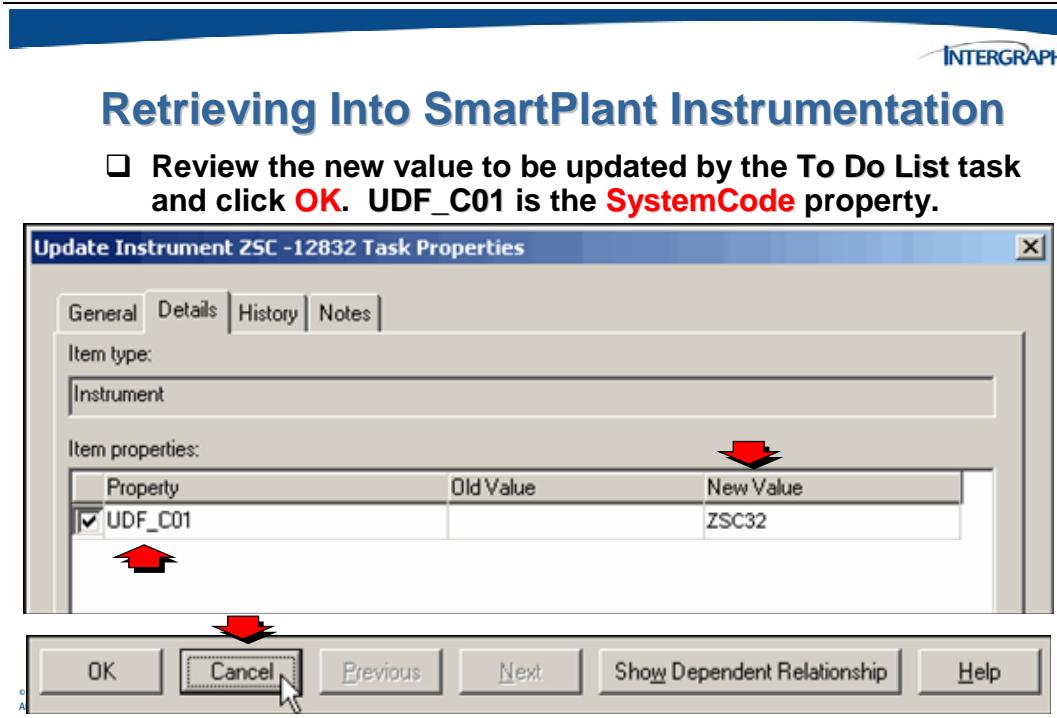


© 2005, Intergraph Corp.
All Rights Reserved.

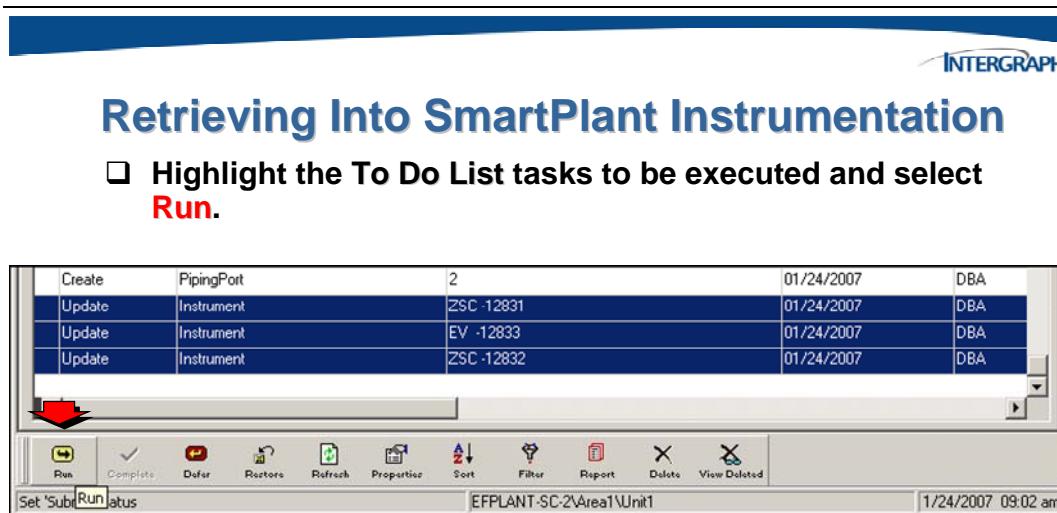
This will display the **To Do List** which contains all of the commands that can be executed within SPI.



© 2005, Intergraph Corp.
All Rights Reserved.



This will allow the engineer to review the new value before it is applied to the existing data.



Once the tasks have been executed, use the browser window to view the results.

Sequence	Instrument Price	Price	Loop Name	Loop Service	SystemCode
					C31 ZSC32

Clear the buffer | Clear the current field

© 2005, Intergraph Corp.
All Rights Reserved.

Now that the document has been successfully retrieved and the selected tasks executed, exit from SmartPlant Instrumentation.

Instrument Type	Process Function Type	IO Type Name	Status
FE	General		NEW
X	General		NEW
X	General		NEW
X	General	PRC	NEW
X	General		NEW
X	General		NEW
CV	Control Valve		NEW
	General		NEW
FE -12813	FE	Flow	NEW
FE -12818	FE	Flow	NEW

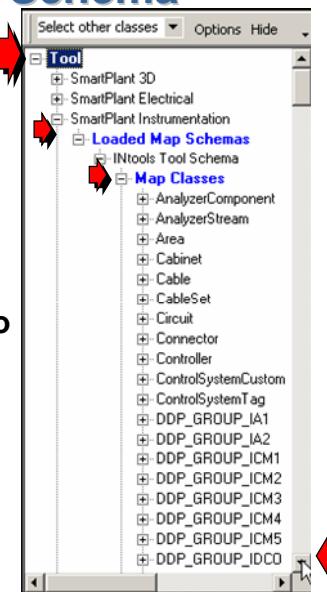
© 2005, Intergraph Corp.
All Rights Reserved.

10.9.4 Publish Mapping for a Custom Property

Now that the retrieve mapping has been completed and tested, the ability to publish a change from SPI will need to be configured and mapped. This needs to be done in “connected” mode using the schema editor.

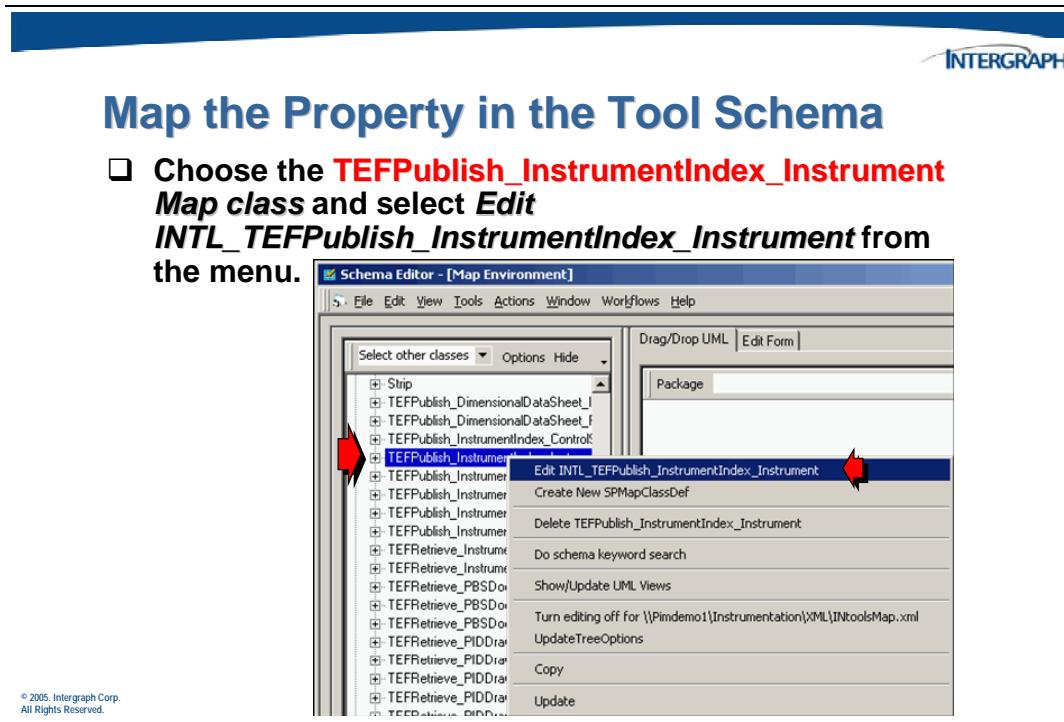
Map the Property in the Tool Schema

- Expand the SmartPlant Instrumentation Tool map schema hierarchy and Map Classes.
- Scroll down the list of Map Classes to locate the possible *TEFPublish* classes.

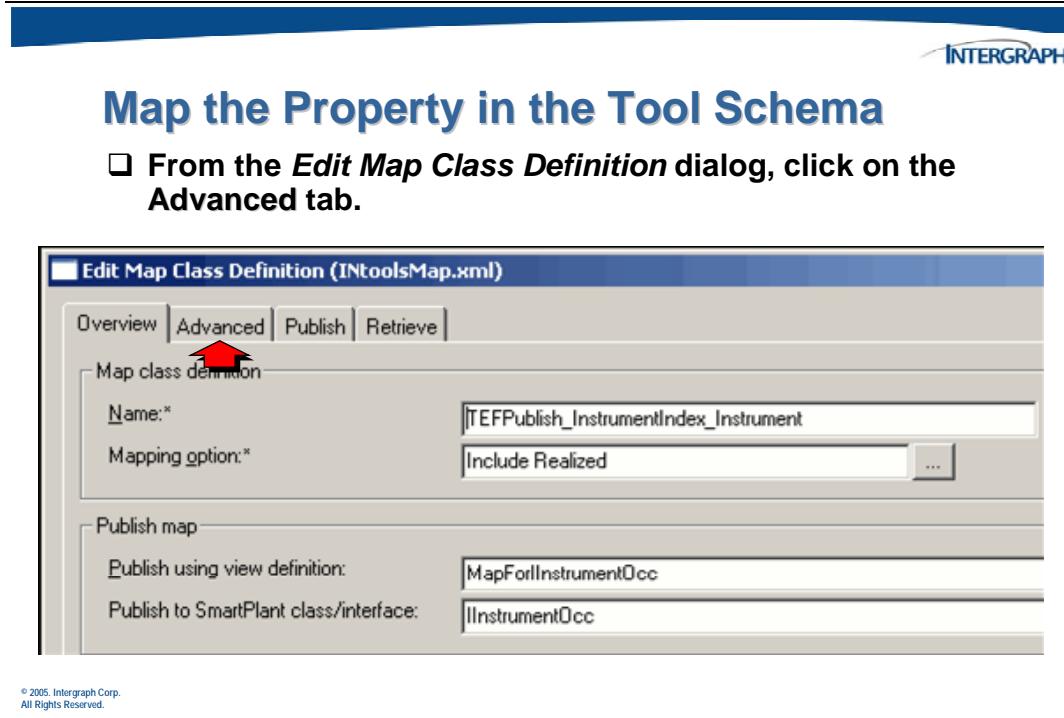


© 2005, Intergraph Corp.
All Rights Reserved.

Remember, the SPI tool adapter uses special **TEFPublish** map class.



The *Edit Map Enumerated List Definition* dialog will display.

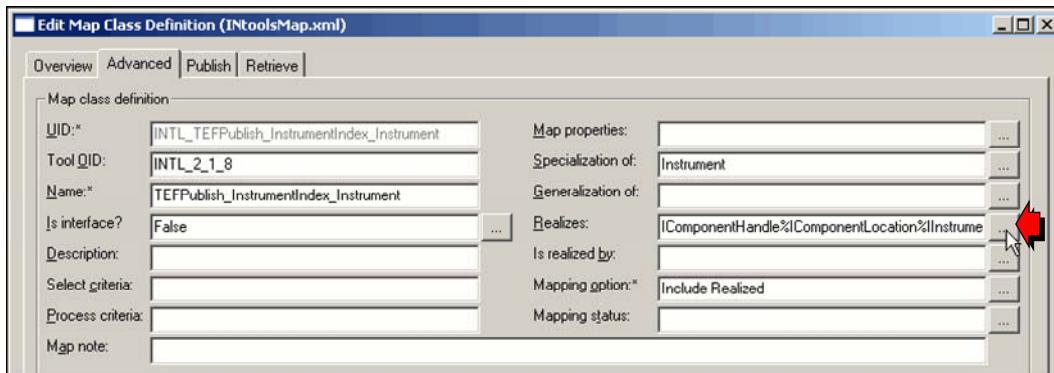


The map class will need to realize the **IInstrumentUDF** map class, which exposes the custom property.



Map the Property in the Tool Schema

- Click the browse (...) button to the right of the *Realizes:* field.



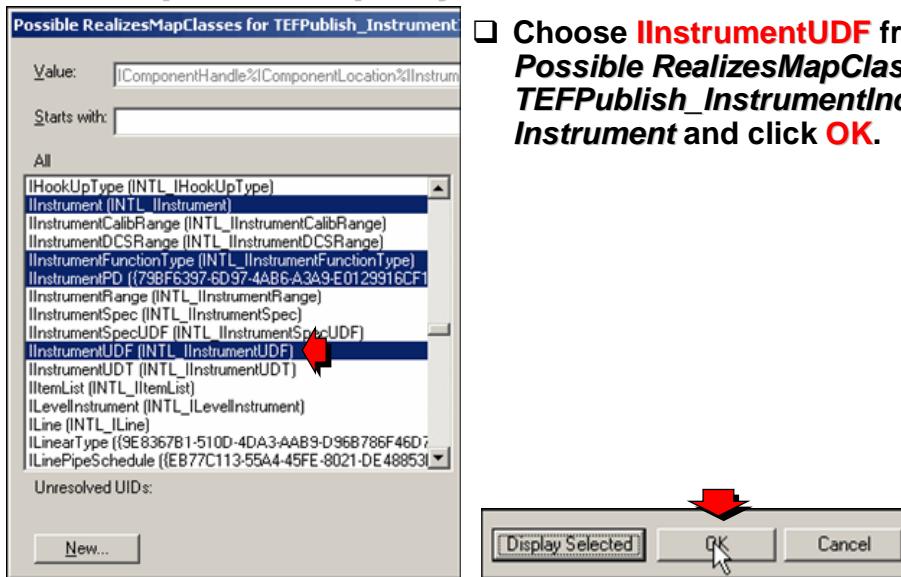
© 2005, Intergraph Corp.
All Rights Reserved.

A list of possible realizes map classes will be displayed.



Map the Property in the Tool Schema

- Choose **IInstrumentUDF** from the **Possible RealizesMapClasses for TEFPublish_InstrumentIndex_Instrument** and click **OK**.

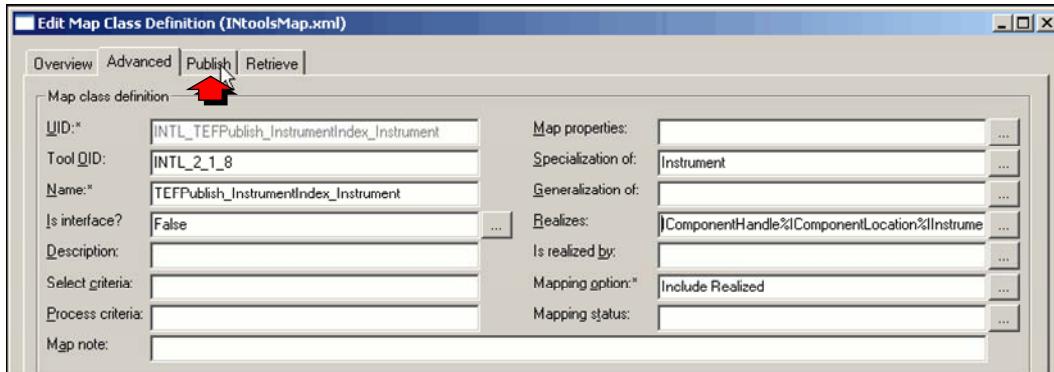


Once the property has been exposed by adding the required map class, perform the mapping.



Map the Property in the Tool Schema

- From the *Edit Map Class Definition* dialog, click on the Publish tab.



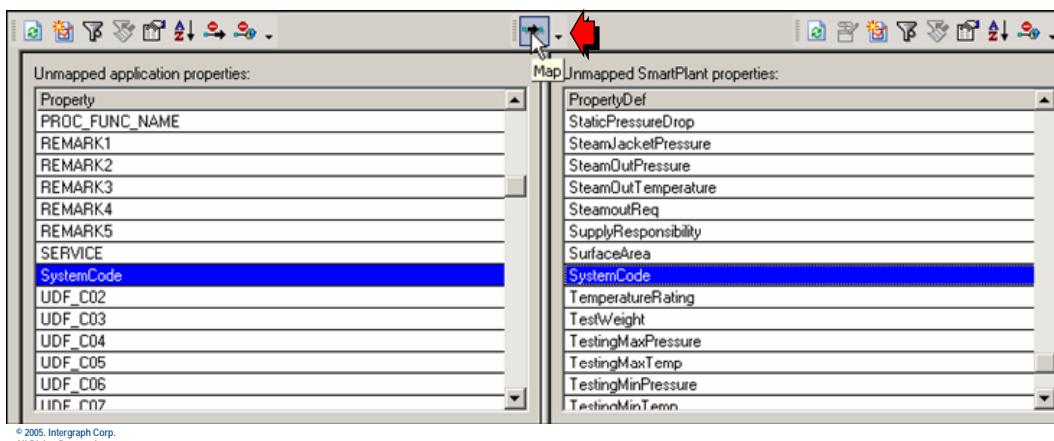
© 2005, Intergraph Corp.
All Rights Reserved.

The **Map** button is used to map an unmapped tool property to an unmapped SmartPlant property.



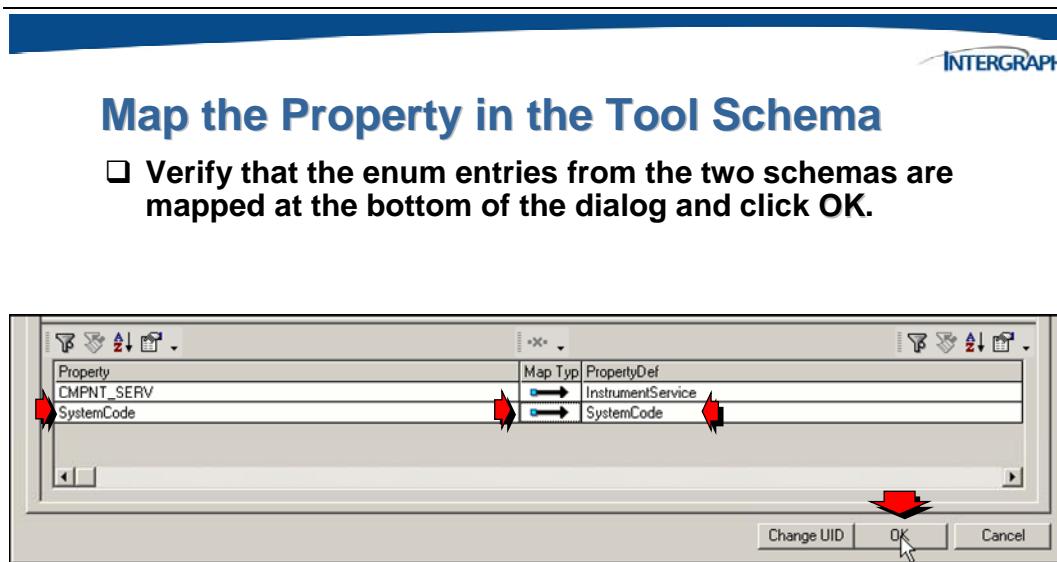
Map the Property in the Tool Schema

- Select the UDF property from the left pane and the SmartPlant property from the right pane and click the **Map** button.



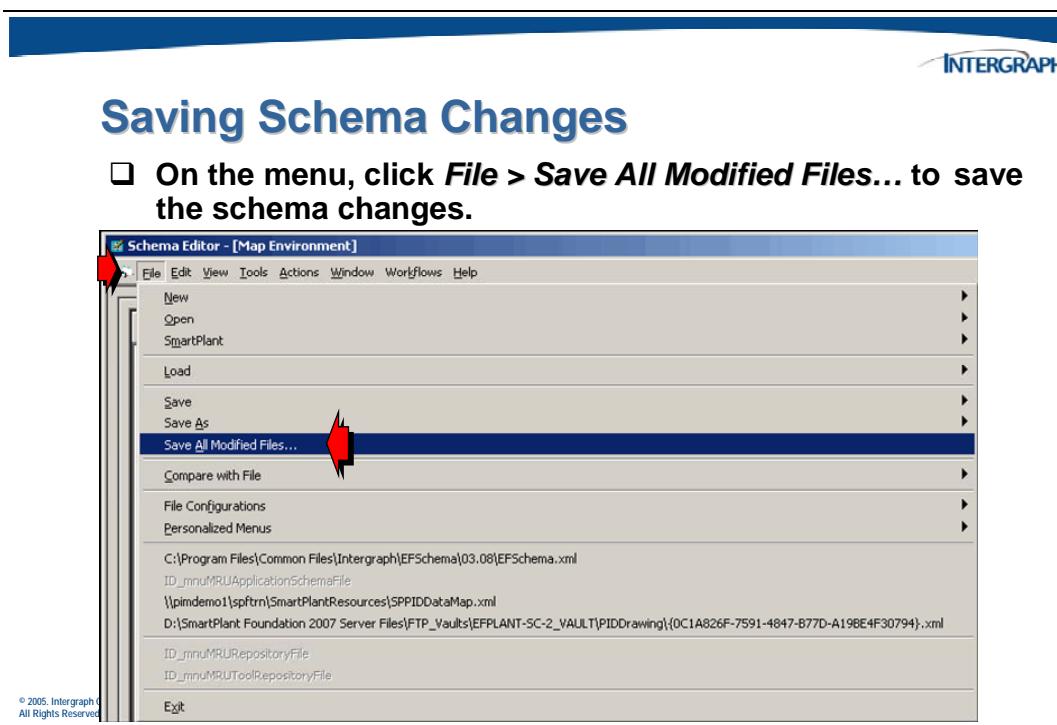
© 2005, Intergraph Corp.
All Rights Reserved.

This completes the mapping for the new custom property for **Publish** operations.

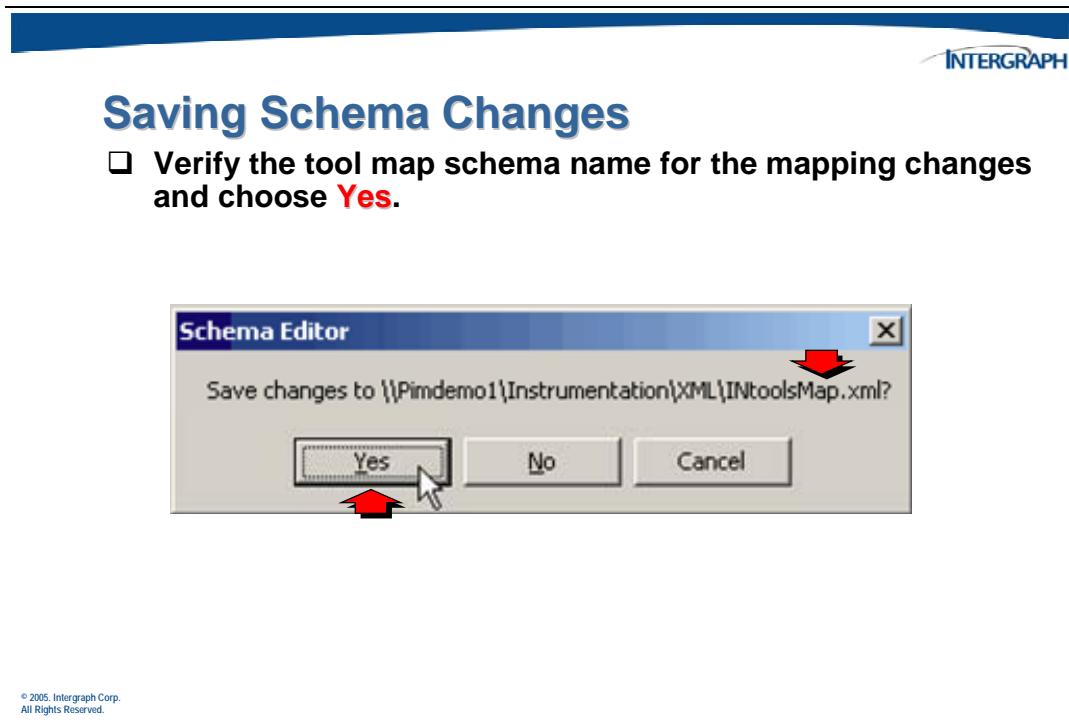


© 2005. Intergraph Corp.
All Rights Reserved.

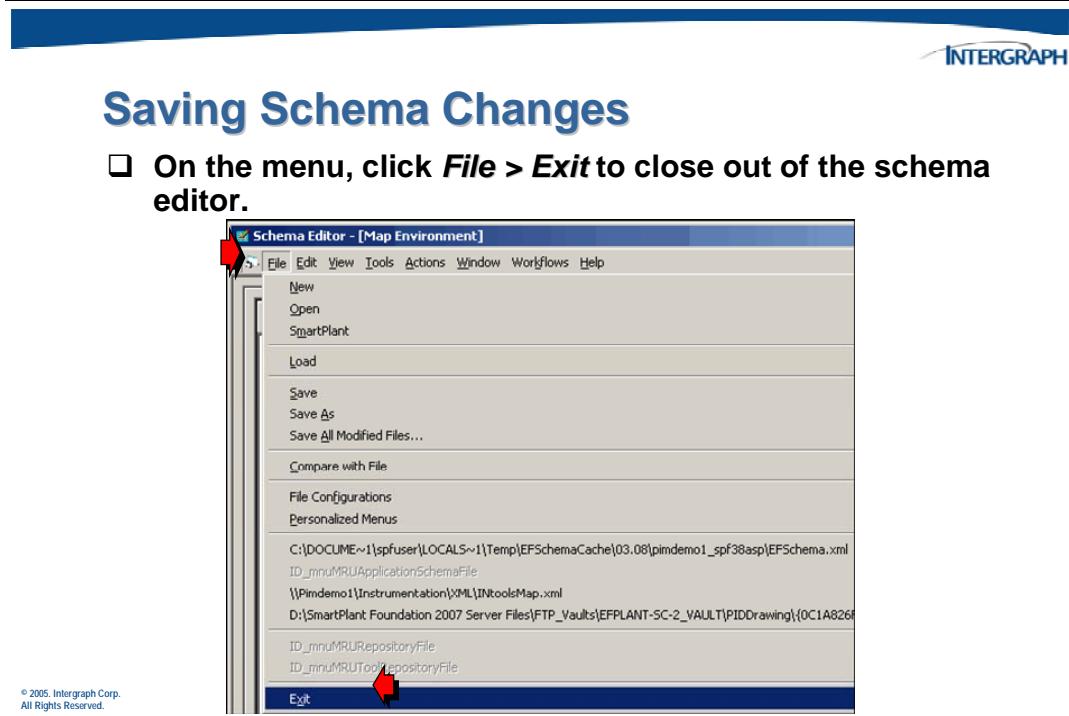
All of the mapping information is stored in memory will need to be saved to the respective xml files.



You will be prompted to save the changes to the tool map schema.



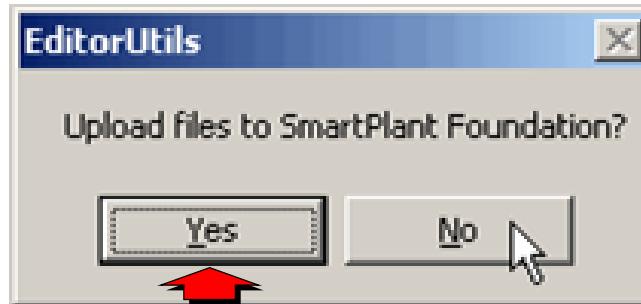
Once all of the schema files have been saved exit the Schema Editor.



Again, since no changes were made to the SmartPlant schema, there is no need to perform an upload operation.

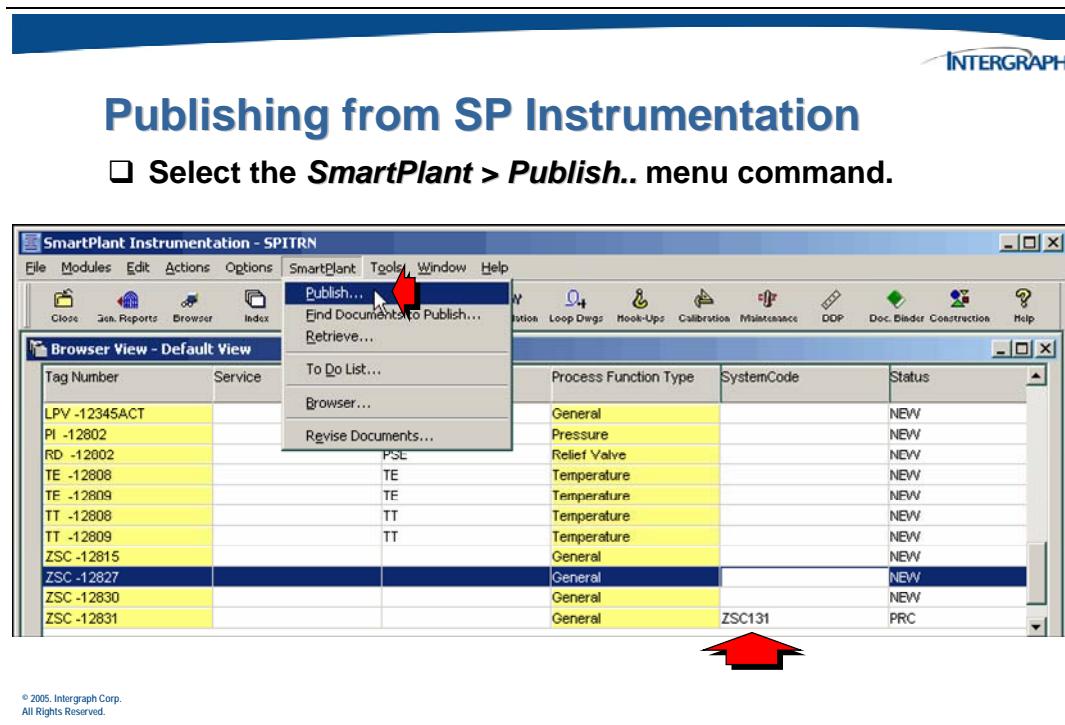
Saving Schema Changes

- To upload the SPI changed files to the SPF server, select **No**.

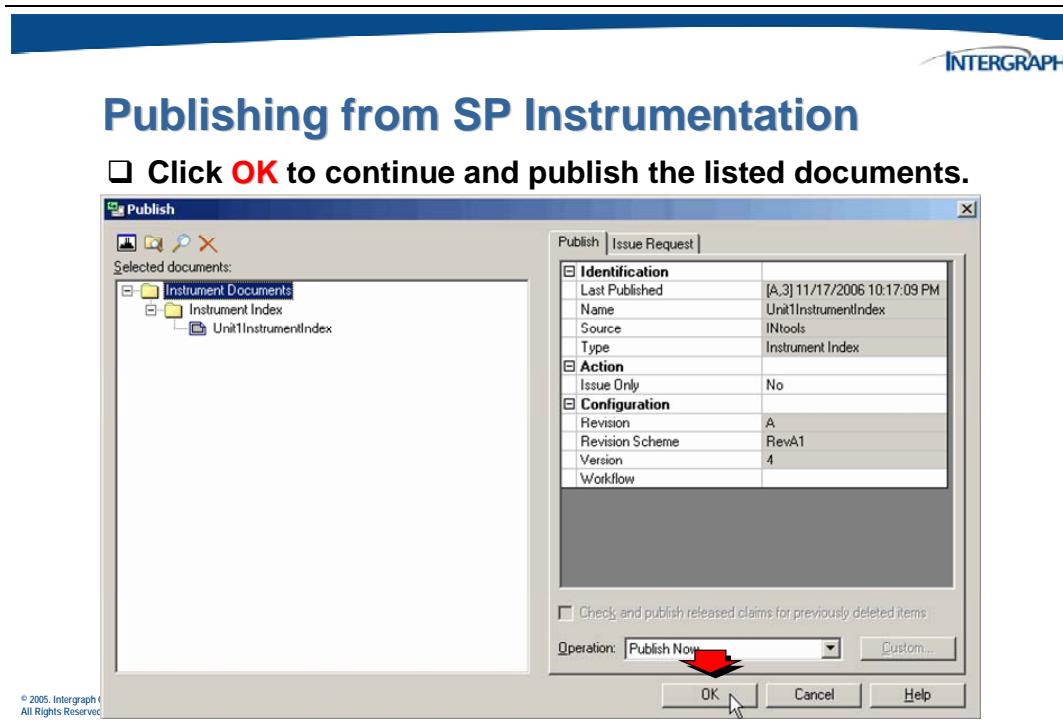


10.9.5 Testing Custom Property Publishing

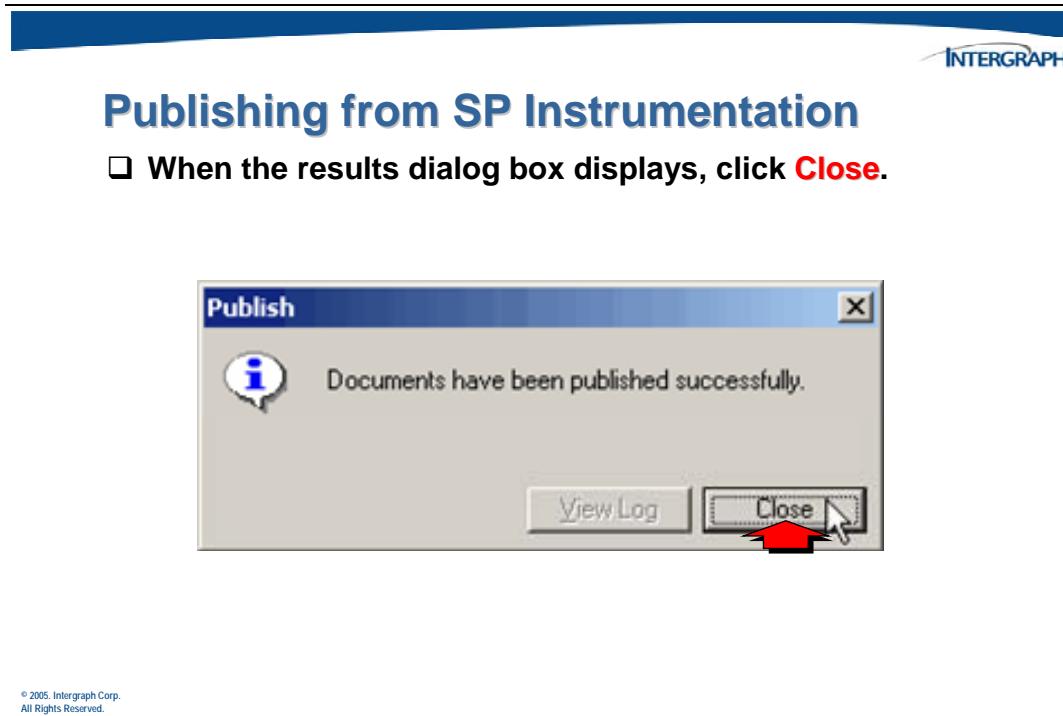
To perform a publish operation; use the **Publish** command from SmartPlant Instrumentation.



The *Publish* dialog box will display.



A dialog box will display after some time to report success or failure of the publish.

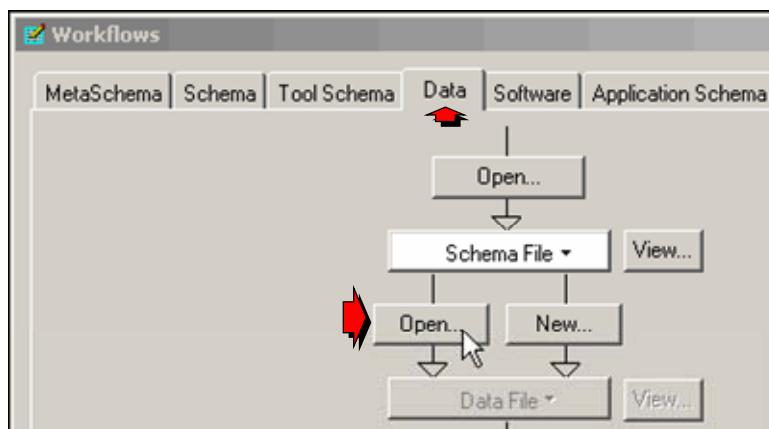


Use the schema editor to review the contents of the published XML files. Once the schema editor has been started, use the **EFSchema.cfg** file to open the schemas. You will also need to use the open data file command to open the published xml.



Publishing from SP Instrumentation

- In the **Workflows** dialog box, click the **Open** button to open a published **XML** data file.



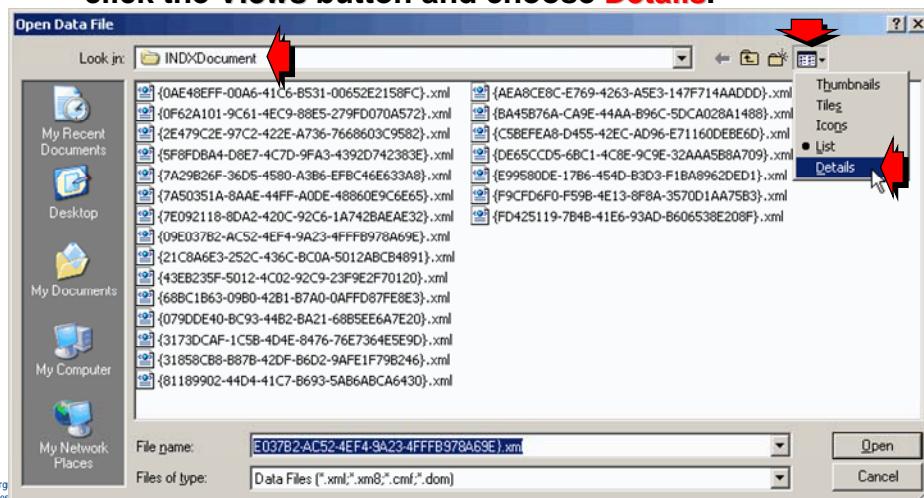
© 2005, Intergraph Corp.
All Rights Reserved.

Open the **INDXDocument** folder to see a list of available XML files. You can use the Views button on the tool bar to sort the folder contents by date.



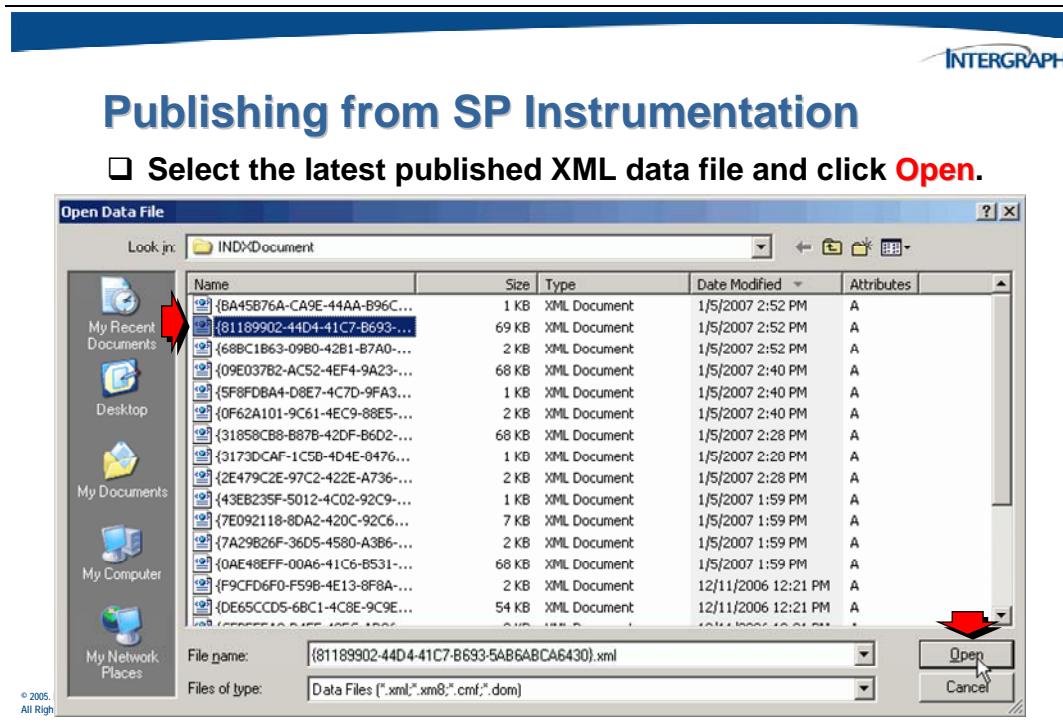
Publishing from SP Instrumentation

- In the **Open Data File** dialog box, select the vault folder, click the Views button and choose **Details**.

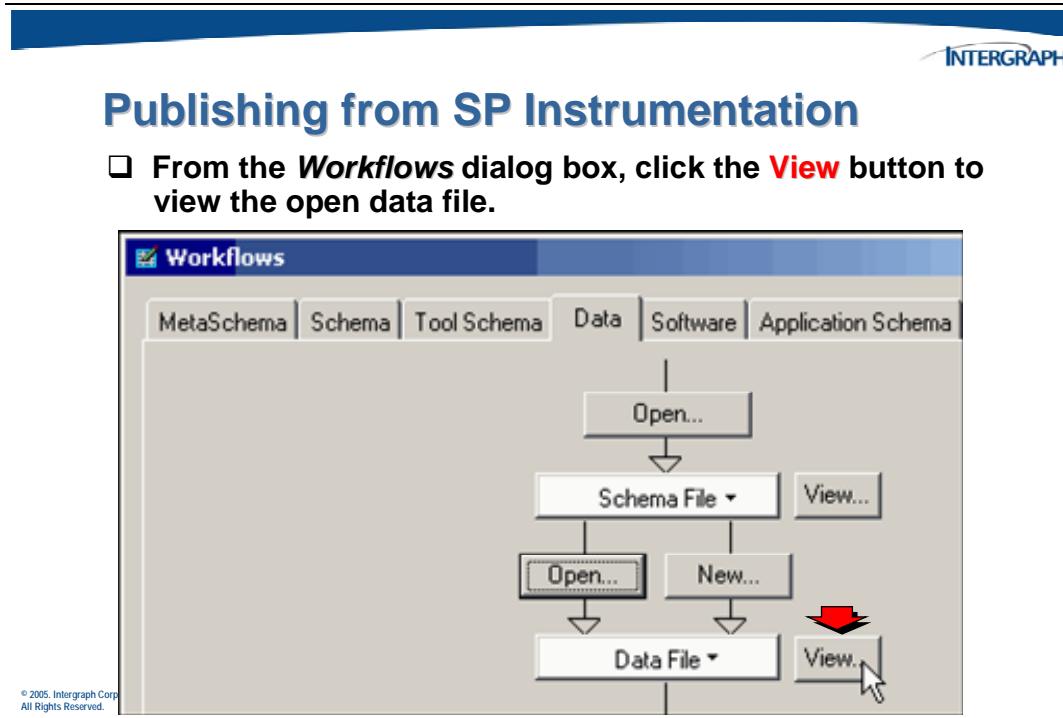


© 2005, Intergraph Corp.
All Rights Reserved.

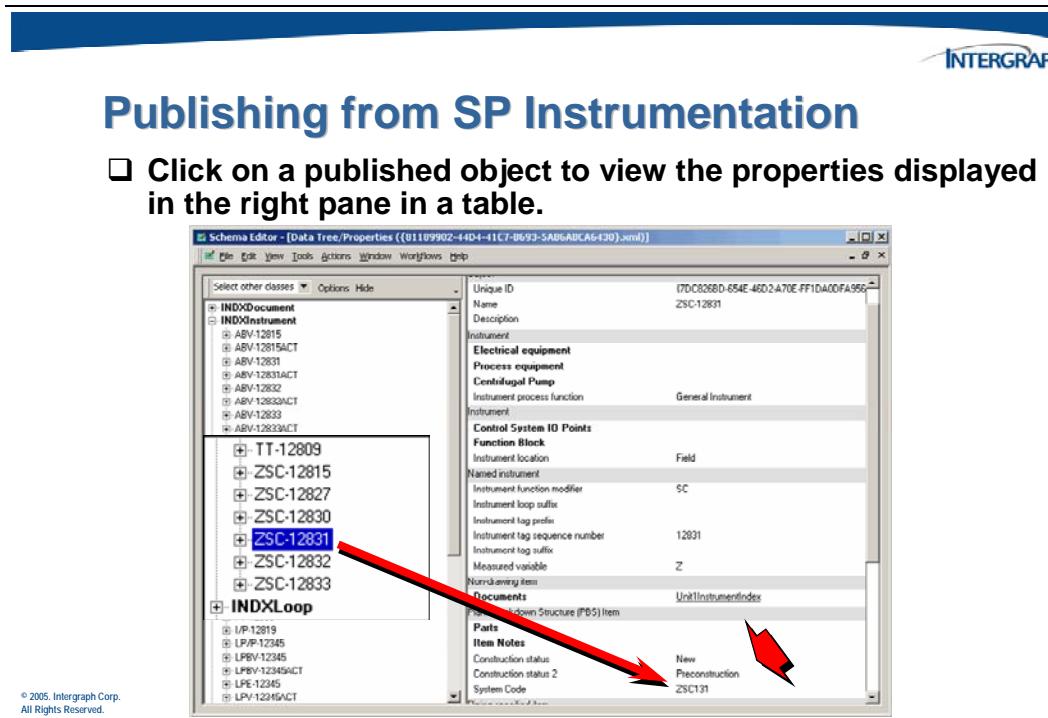
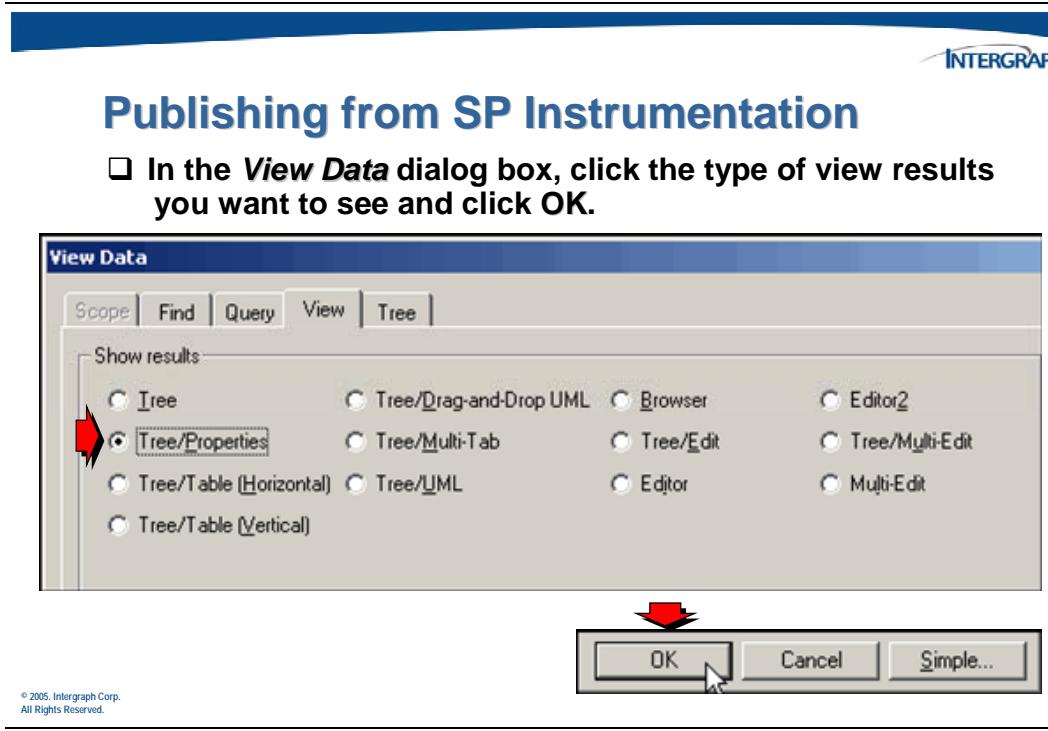
This vault folder, **INDXDocument**, is the server location where the published XML file is stored. The published file will be one of the latest ones in the folder.



After the published XML file has been opened, use the schema editor to view it.



The schema *Tree/Properties* view allows you to traverse relationships in the tree and see information for items selected in the tree view in a table format.



10.10 Activity 2 – Adding a Custom Property to SmartPlant Instrumentation

The goal of this activity is to give you the opportunity to create a custom property in the SmartPlant Instrumentation application meta data and the SmartPlant Instrumentation tool map schema. You will then perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant Instrumentation > Administration** to start the *SmartPlant Instrumentation Administration* module.
3. Select the Domain Administrator.
 - Click **SPITRN** in the *Open Administration Module* dialog.
 - Click **OK** to close the dialog.
4. Use Domain Administration to add the new custom SPI property.
 - From the *Domain Administration* window, select the **Fields** tool.
 - In the Custom Fields dialog, select the **Plant – EFPLANT-SC-2** and the **Item Type - Instrument**.
 - Add the User Defined Field (UDF) **SystemCode** in the *Definition* field and a *Length* of **8**.
 - Click **Apply**.
 - Select the **Item Type - Loop**.
 - Add the User Defined Field (UDF) **SystemCode** in the *Definition* field and a *Length* of **8**.
 - Click **Apply**.
 - Click **Close** to close the *Domain Administration* window.
 - File > Exit** from *Domain Administration*.

5. Start SPI and configure the new custom property in the *Instrument Index* browser.
 - From the *Instrument Index* module, use the **Browse** button to display a *Browser View* window.
 - Use the **Manager** button to add the new property to the *Style* of the **Default View**. Refer to section 10.9 for details on how to do this.
 - Open the *Instrument Index Browser View* window to confirm that the new property will display in the view.
 - Exit** from SPI.
6. Click **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
7. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant...**
8. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
 - If prompted for a SmartPlant Foundation login, use **adminuser** with no password, otherwise continue.
 - Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with
 - Click **Next**
 - Select the tool map schema to be loaded, *INtools/INtools Tool Schema*
 - Enable the **Load map schema** toggle
 - Enable the **Connect to application schema** toggle
 - Click **Finish**
9. When the *Synchronize* dialog displays, confirm that the new property values have been imported by the meta data adapter and click **OK**.
10. Configure the new custom property for retrieve mapping.
 - Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant Instrumentation Tool Schema entries are displayed.

- Expand the **Map Classes**, scroll down and right-click on the *TEFRetrieve_PIDDrawing_Instrument* entry in the tree and choose **Edit** from the dynamic menu.
 - Use the **Retrieve map browse (...)** button to add the **IInlineInstrumentOcc** and **IInstrumentOcc** interfaces to the *Retrieve from SmartPlant class/interface* field.
 - Select the **Advanced** tab in the *Edit Map Class Definition* dialog.
 - Use the **Realizes browse (...)** button to add the **IInstrumentUDF RealizesMapClass** to the *Realizes* field.
 - Select the **Retrieve** tab in the *Edit Map Class Definition* dialog.
11. Highlight the *SystemCode* MapProperty in the **Unmapped applications properties** control and the *SystemCode* MapProperty in the **Unmapped SmartPlant properties** control in order to perform the schema mapping.
- Make sure that **SystemCode** is highlighted in the middle control in the application section and that **SystemCode** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPI tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *SystemCode* maps to *SystemCode* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Class Definition* dialog box.
12. Save the changes to the all schema files.
- From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **No**.
13. Use the SPI Schema Configuration Wizard to modify the SPI database schema and realize the interface containing the new property (E:\SmartPlant\Instrumentation\spiscw.exe).
- On the *Schema Options* screen, enable the **Change existing schema definition** option.
 - Select the **TEFRetrieve (retrieve)** schema.
 - Select the *Document type *PID Drawing*.

- Take the defaults until the *Select Object Interfaces* screen. Choose **IInstrumentUDF** from the *Available interfaces* list, select **Add >** and add this to the *Selected interfaces* list.
 - Take the defaults until the *Completing the Schema Configuration Wizard* screen. Choose the **Finish** button.
14. Using the example in section 10.9.3, retrieve the P&ID instrument into SP Instrumentation.
- Select the **SmartPlant > Retrieve...** command to retrieve the changes/additions. Select **Show All documents** and use only 128-5001 and NOT 128-5001.pid.
 - Use the **To Do List...** command to view the changes/additions.
 - Review the *Properties* of one of the instrument objects that have a status of **Update** to see the *Old* value and the *New* value once this task is executed.
 - Select and **Run** the *Update* tasks to apply the new retrieved values.
 - Use the Instrument Index Module to review the results of the retrieve.
15. OPTIONAL: Configure publish mapping for a custom property, publish a change to an instrument and use the schema editor to see the change in the XML file.
16. When you are finished with this activity, you may take a short break until the other students have finished.

11

C H A P T E R

Mapping with SmartPlant Electrical

11. Mapping with SmartPlant SPEL

In the last chapters, the concept of extending the delivered schema was introduced along with using the SmartPlant Schema Editor to perform the necessary mapping. In the examples, new enumerated list items and enumerated lists were created, as well as properties to use those lists and string properties. The process involved using the Data Dictionary Manager to add the desired Select List entries to the SPPID application. The next step was to use the Schema Editor and the SPPID meta data adapter to read the new enumerated additions and automatically synchronize those additions with the SPPID tool map schema. The final steps were to add the new entries to the SmartPlant Schema and map the two schemas together using the schema editor Map Environment.

In this chapter, we will do the same kind of steps to make addition to the SmartPlant SPEL application and and SPEL tool map schema. The same properties that we added to SPPID will be propogated forward for use in SPEL, and then will be mapped. The goal is to created the properties and mapping so that these values can be shared between SPPID and SPEL using SmartPlant Foundation as an integration tool.

Again, there are two different methods for making these changes. However, in this chapter we will focus on the “outside-in” approach, where changes are made on the tool side and then moved forward into SPF.



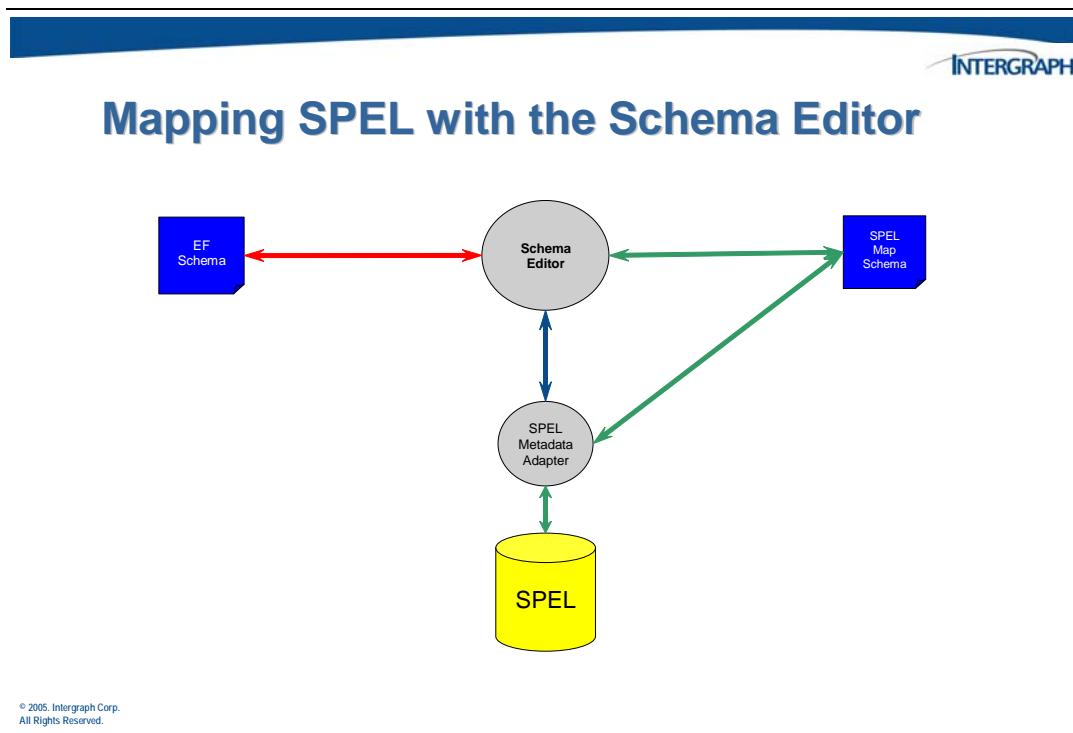
Mapping SPEL with the Schema Editor

Just like with SPPID, there are two methods for extending the delivered schema:

- Make any necessary additions in the SPEL Meta-Schema using the Data Dictionary Manager**
 - Recommended process for adding new properties

- Make any changes using the schema editor, which will allow you to add those changes to the SPEL Meta-Schema, the SPEL tool map schema, and the SmartPlant Schema**
 - Recommended process for adding a new enumerated list

The figure below shows the function of the metadata adapter with the schema editor.

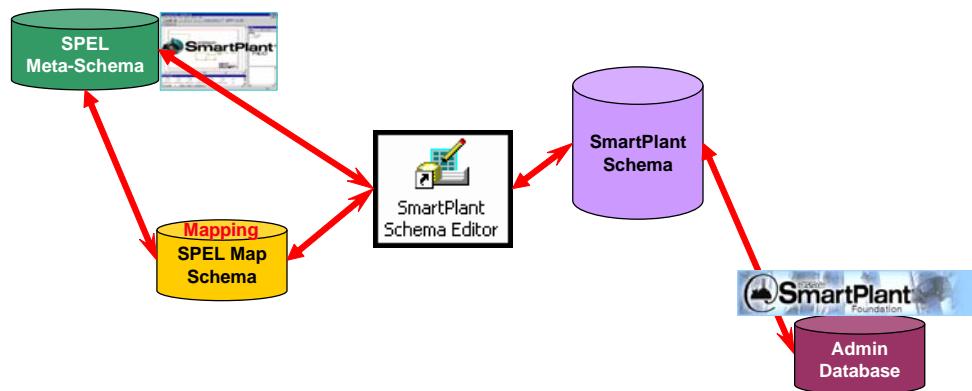


As in the previous examples, the desired change will first be made in the SmartPlant SPEL application using the Data Dictionary Manager tool. There is a specific reason for approaching the problem in this manner. The property to be added could be used by several different types of objects in SmartPlant SPEL; Electrical Motor, Cable, Wire, Instrument, etc. Rather than add the same property multiple times, if the new property will be used as a “global” *Plant Item* item type property, then it can be added to the Plant Item table in the data dictionary but be available to other classes through inheritance. The *PlantItem* item type is a classification that provides common relationships and attribution for all the types of objects that may exist independently of any other object. This item type is derived from the *ModelItem* class. The subclasses of *ModelItem* include all those types whose existence is not dependent on another.

Once the new property has been added with the Data Dictionary Manager, the schema editor will be used to read this new property from the application database, which is also called the tool meta schema. This is accomplished by a part of the software known as the metadata adapter. The task of the metadata adapter is to extract information from the meta schema and automatically update the contents of the tool map schema so that the meta schema and the tool map schema are synchronized.



Mapping SPEL with the Schema Editor



© 2005. Intergraph Corp.
All Rights Reserved.

- Changes are added to the **SPEL Meta schema** via the *Data Dictionary Manager* (new property defined).
- Any changes are automatically added to the **SPEL Map Schema** when the schema editor synchronizes the schema files.
- The same changes get added to the **SmartPlant Schema** using the schema editor *Map Environment*.
- Any unmapped changes to the schemas can be mapped using the schema editor in preparation for publish/retrieve operations.

This will allow for the smooth flow of published data from SmartPlant SPEL into the SmartPlant Foundation database.

11.1 Adding New Simple Properties

The procedure for adding properties to the authoring tools differs from tool to tool. The following example describes the procedure for adding a new property to the SmartPlant SPEL data model using SmartPlant Data Dictionary Manager (delivered with SmartPlant Engineering Manager). For more information about adding properties to SmartPlant SPEL, see the SmartPlant Data Dictionary Manager documentation.

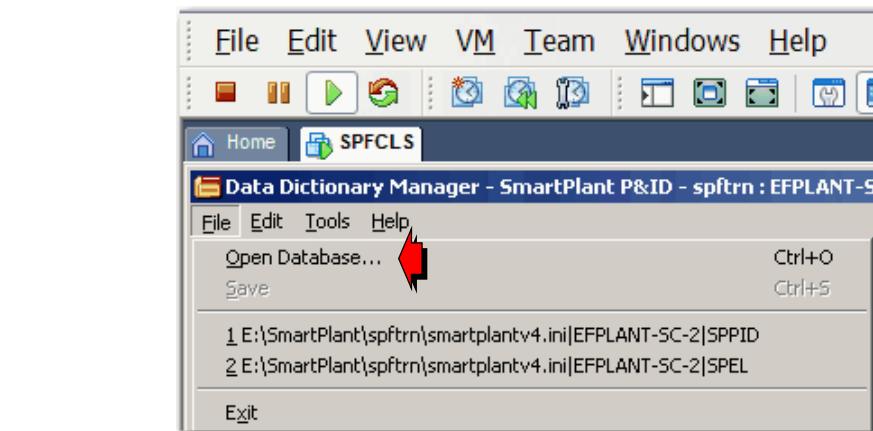
To add a property to SmartPlant Electrical, open Data Dictionary Manager by clicking **Start > All Programs > Intergraph SmartPlant Engineering Manager > Data Dictionary Manager**.

As we have been working in the SPPID database with the Data Dictionary Manager tool, you will need to use the **File > Open Database** command to open the SmartPlant Electrical database.



Add Properties to the Meta-Schema

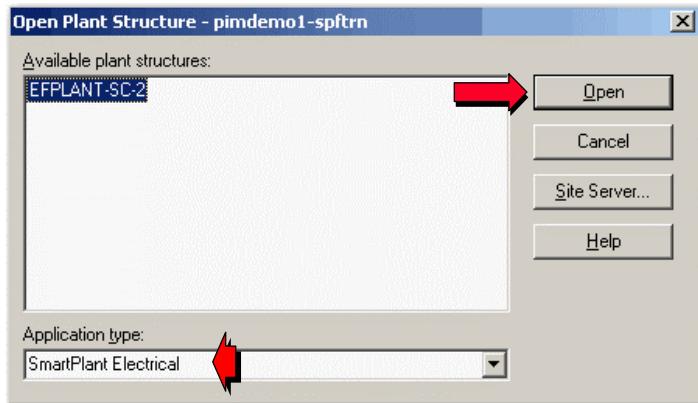
- Open the SmartPlant Electrical Database in SmartPlant Engineering Manager**





Add Properties to the Meta-Schema

- ❑ Select **SmartPlant Electrical** under *Application type*, and click **Open**.

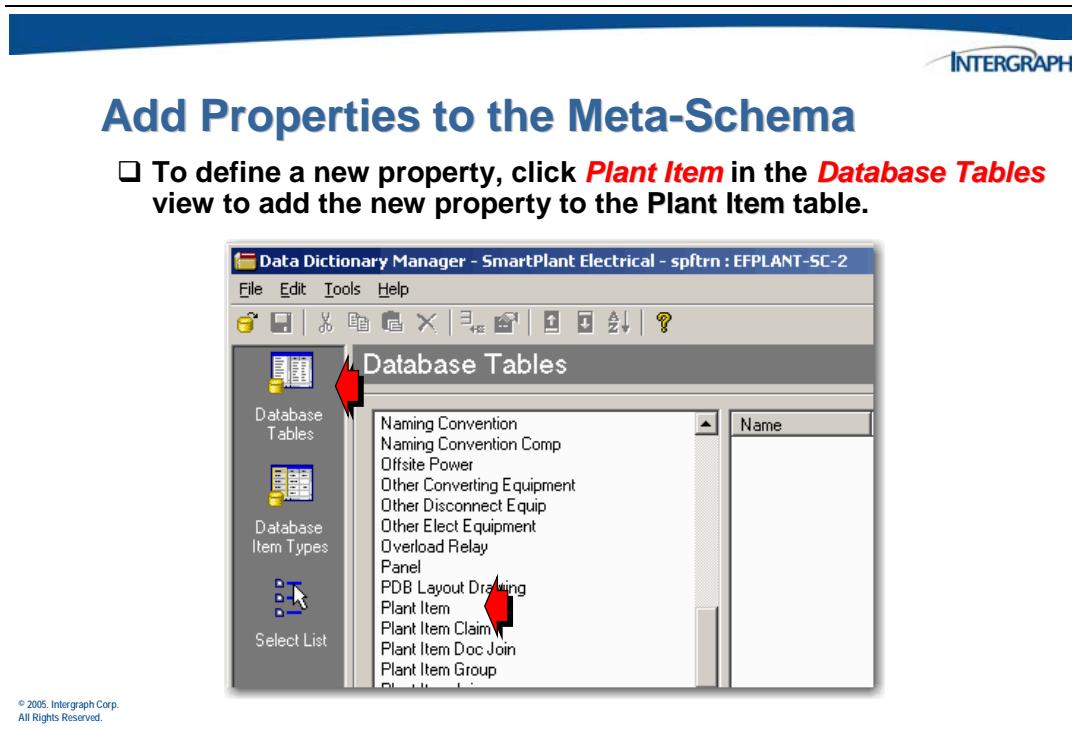


© 2005, Intergraph Corp.
All Rights Reserved.

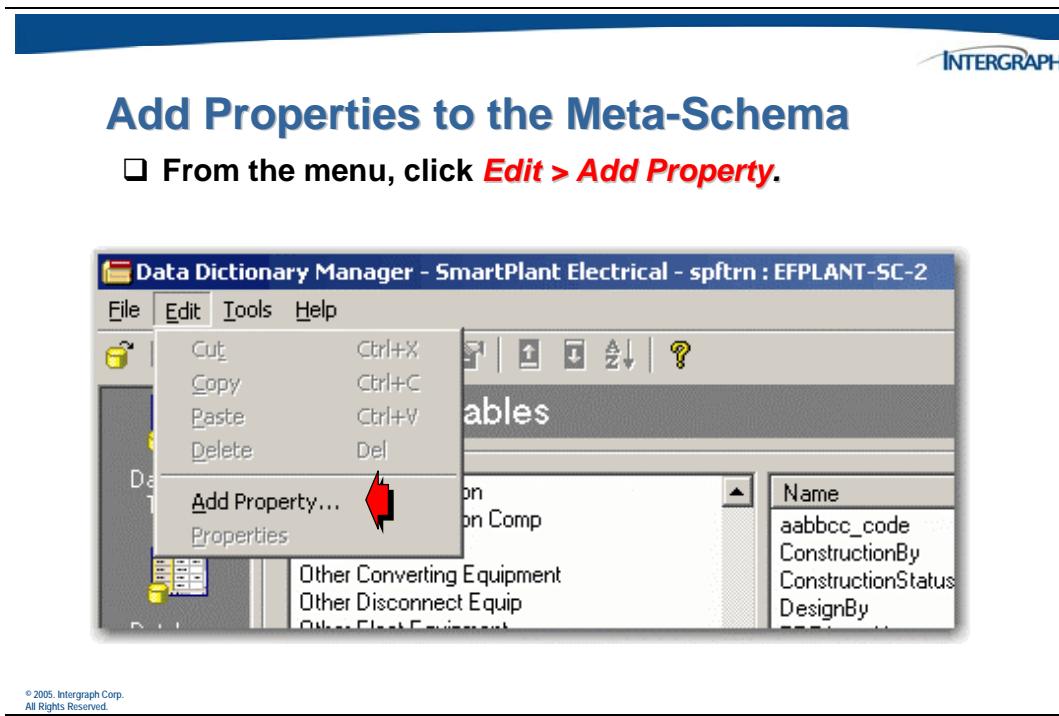
The following example describes how to add a new property to the SmartPlant SPEL data model. This property includes the following:

- A simple property without an enumerated list

When you open Data Dictionary Manager, the **Database Tables** view is already selected. To select the database table to which you want to add a property, click the name of the table in the list.



After you select the table that you want, you can add a property to that table.

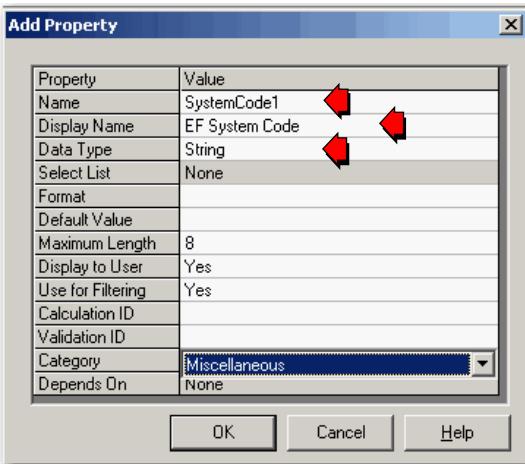


The *Add Property* dialog box appears. Define information for the new property in the *Add Property* dialog box.

In this example, we will be adding a new property called SystemCode1, because the SmartPlant Electrical database already contains a property called SystemCode.

Add Properties to the Meta-Schema

- In the *Add Property* dialog box, define attributes for the new property, including the **Name**, **Display Name**, and **Data Type**.

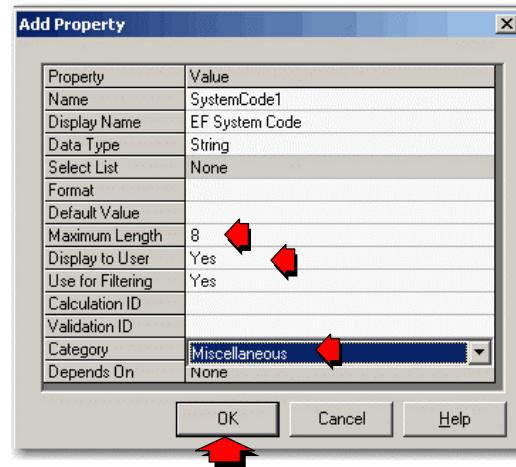




Add Properties to the Meta-Schema

- Continue to define attributes for the new property, including the **Maximum Length**, **Display**, **Filtering**, and **Category**.

- Click **OK** to add the new property to the Plant Item table.



© 2005, Intergraph Corp.
All Rights Reserved.

When you click **OK**, the list of properties for the *Plant Item* table updates with the new property.



Add Properties to the Meta-Schema

The new **SystemCode1** property appears in the Plant Item table.

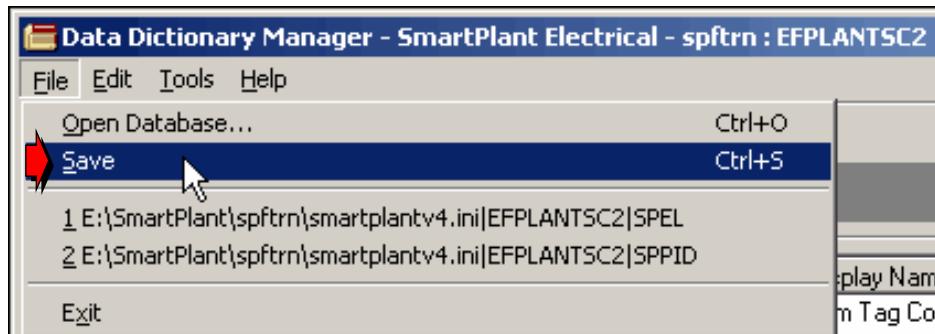
Name	Display Name	Data Type	Select List	Format
ItemTagFullName	Item Tag Compl...	String	None	a80
Name	Name	String	None	Variable Length
PlantItemType	Plant Item Type	Select List	Plant Item Type	Variable Length
ReferenceDataC...	Reference Dat...	Select List	Boolean Values	Variable Length
RequisitionBy	Requisition By	Select List	Requisition Res...	Variable Length
RequisitionNo	Requisition Nu...	String	None	Variable Length
SP_AllowPublish...	Allow Publish Fl...	Select List	Boolean Values	Variable Length
SP_KKSEquip...	KKS Equipment...	String	None	Variable Length
SP_KKSCcompon...	KKS Componen...	String	None	a2
SP_KKSEquipm...	KKS Equipment...	String	None	Variable Length
SP_KKSEquipU...	KKS Equipment...	String	None	Variable Length
SP_KKSSystem...	KKS System Key	String	None	Variable Length
SP_KKSSystem...	KKS System Se...	String	None	a2
SP_KKSTotalPlant	KKS Total Plant	String	None	Variable Length
SP_PartNo	Part Number	String	None	Variable Length
SP_PIDDrawing...	P&ID Drawing ...	String	None	Variable Length
SupplyBy	Supply By	Select List	Supply Respon...	Variable Length
SystemCode1	EF System Code	String	None	None
TagPrefix	Tag Prefix	String	None	a50
TagSequenceNo	Tag Sequence ...	String	None	Upper Case - V...
TagSuffix	Tag Suffix	String	None	a50

© 2005, Intergraph Corp.
All Rights Reserved.

After you add the property, save your changes to the SmartPlant SPEL meta schema.

Add Properties to the Meta-Schema

- From the menu, click **File > Save**.

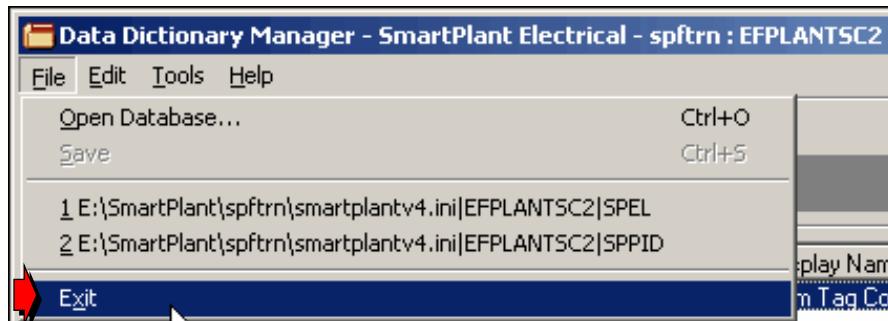


© 2005. Intergraph Corp.
All Rights Reserved.

Click the **File > Exit** command to close out of Data Dictionary Manager.

Add Properties to the Meta-Schema

- Click **File > Exit** from the menu.



© 2005. Intergraph Corp.
All Rights Reserved.

11.1.1 Performing a Schema Analysis

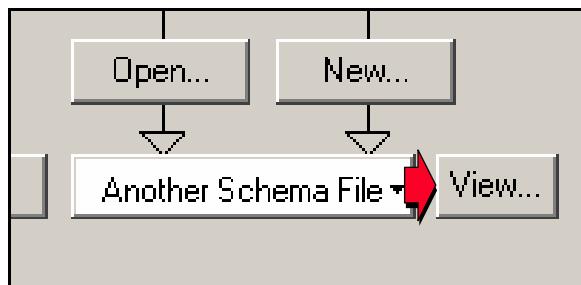
Before adding a new property to the SmartPlant schema, use the Schema Editor to view and analyze the available class definitions. The key to this analysis is to determine the best interface definition to use to expose any new properties that will be added to the extended schema. In the following example, the SystemCode property will need to be exposed by an interface that is realized by a majority if not all of the needed class definitions.

To begin the analysis, start the schema editor and open the schema view to be used to perform this analysis.

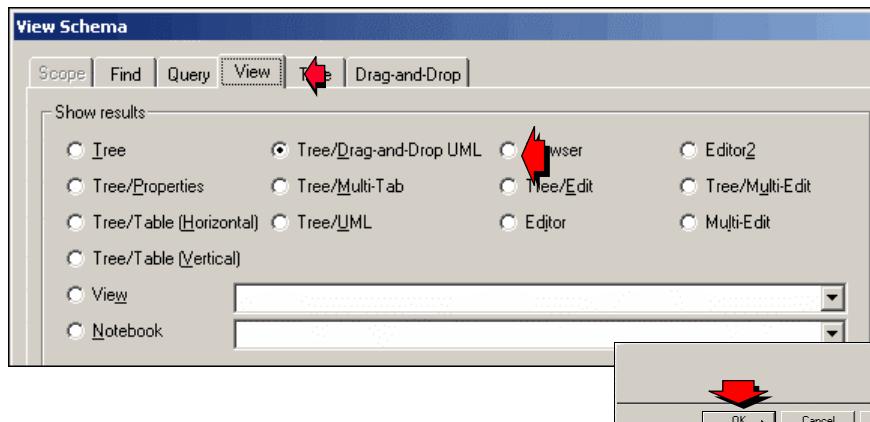


Schema Analysis

- Click the **View** button to the right of the **Another Schema File** button.



Click the **View** tab in the *View Schema* dialog box. For the following example, the **Tree/Drag-Drop UML View** will be used to perform the necessary schema analysis.

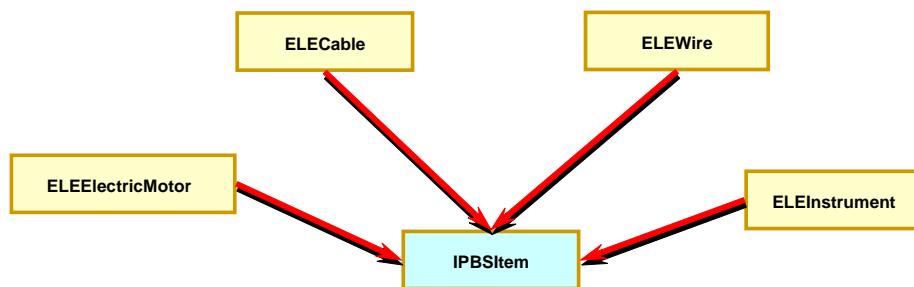


© 2005. Intergraph Corp.
All Rights Reserved.

The SmartPlant schema class definitions to be reviewed will correspond to the tables listed in the Data Dictionary Manager. Here is a list of some of the classes (class defs) to be displayed: **ELEElectricMotor**, **ELECable**, **ELEWire** and **ELEInstrument**.

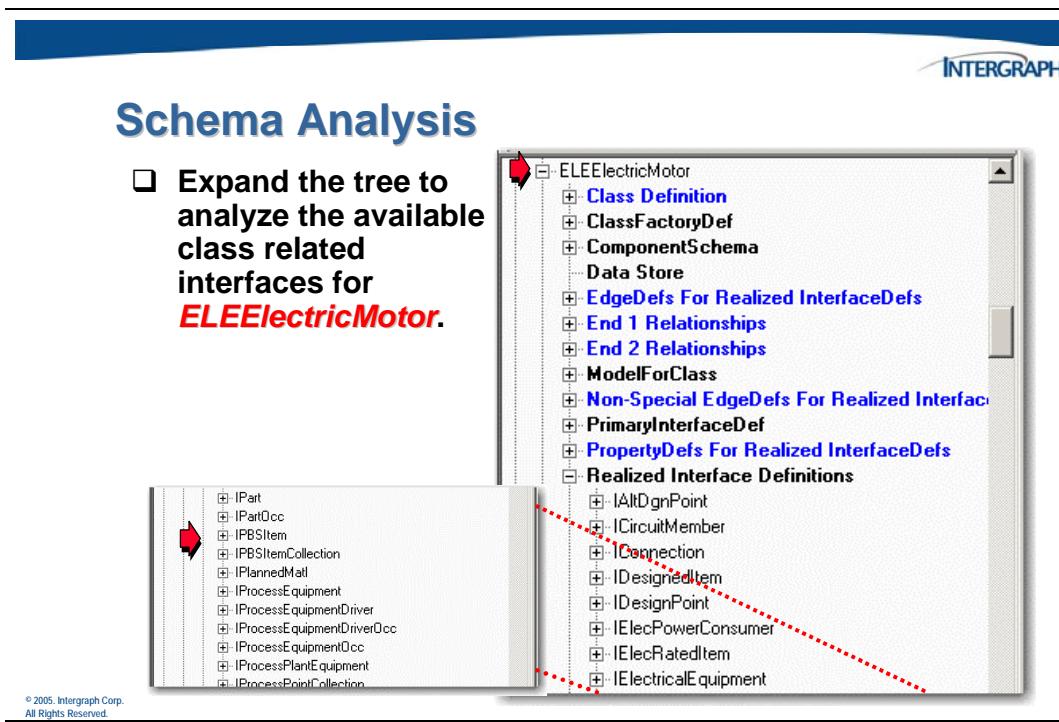
Schema Analysis

One of the common interfaces realized by the SPEL classes
is **IPBSItem**.



© 2005. Intergraph Corp.
All Rights Reserved.

Expand the **ELEElectricMotor** class in the tree to view the realized interfaces. The **IPBSItem** interface is a common interface realized by many classes and is a possibility.



Note:

When adding properties to the SmartPlant Schema, you will need to do similar research to determine what interface is shared by all the classes of objects that you want to be able to use the new property. Another option, however, would be to create a new interface for your customer property (or properties).



Adding a New SmartPlant Property

Add a Property
to Tool Map
Schema

Add a Property
to SmartPlant
Schema

© 2005. Intergraph Corp.
All Rights Reserved.

In the above figure, the process of adding a new property to the tool map schema will be automated by the tool metadata adapter.

11.1.2 Updating the Tool Schema

Typically, before you can map a property from the tool map schema to the SmartPlant schema and back, you must use the Schema Editor to add the new property to the SmartPlant schema. However, in this case, we already created the property to the SmartPlant Schema when we mapped the property from SPPID.

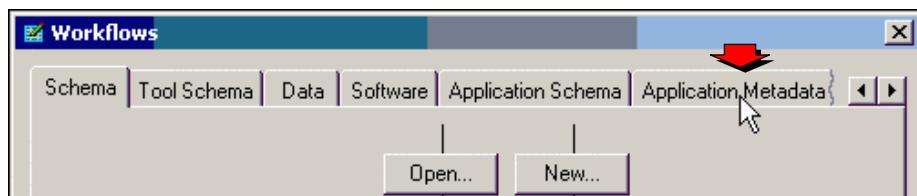
So, we just need to synchronize our map our tool map schema with the tool meta schema and then map the tool map schema to the property already in the SmartPlant Schema.

Start the Schema Editor and connect to SmartPlant.

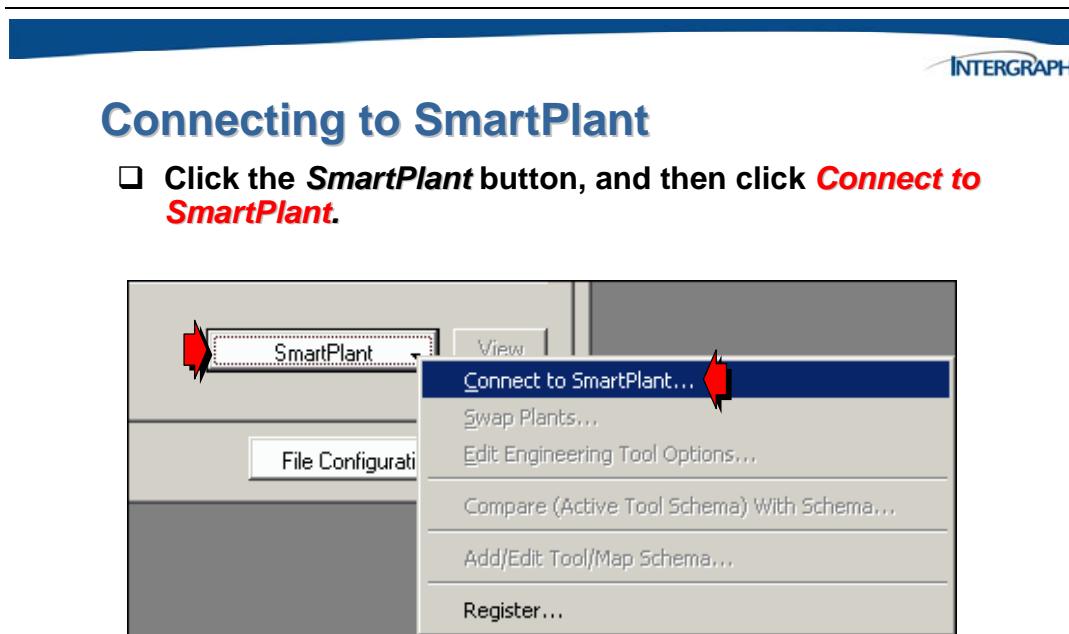


Connecting to SmartPlant

- On the **Workflows** dialog box, go to the **Application Metadata** tab to connect to the SmartPlant Map Environment.

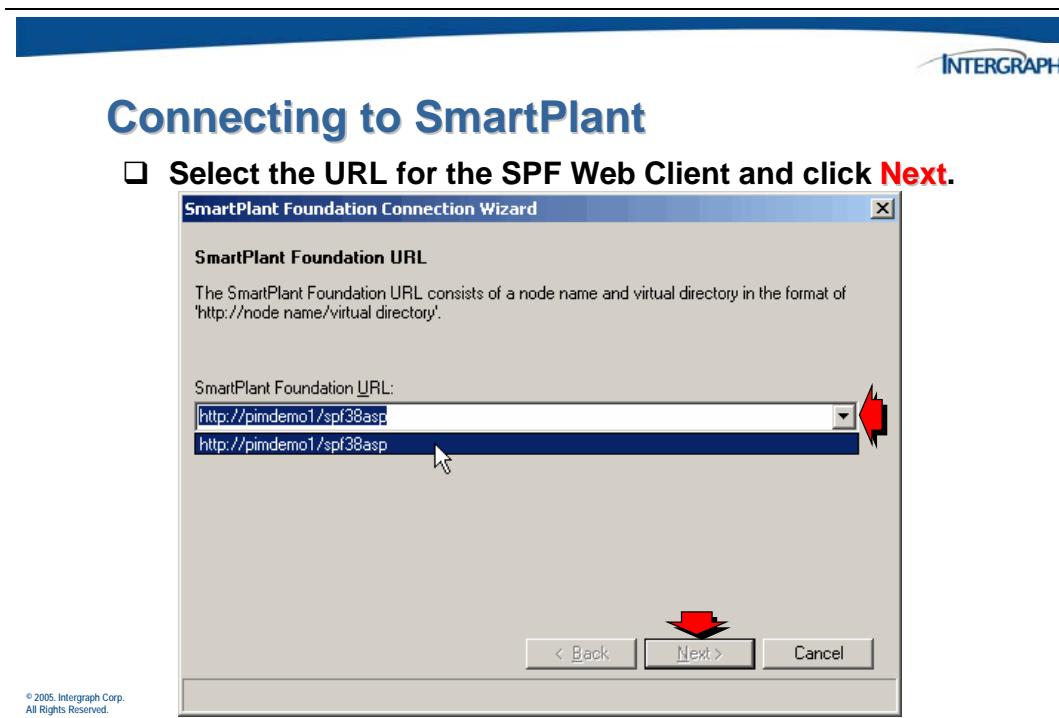


The *Application Metadata* dialog will appear.



© 2005, Intergraph Corp.
All Rights Reserved.

The *SmartPlant Foundation Connection Wizard* will be displayed.



In the *SmartPlant Foundation URL* field, select the SmartPlant Foundation database with which you want to connect, and click **Next**.



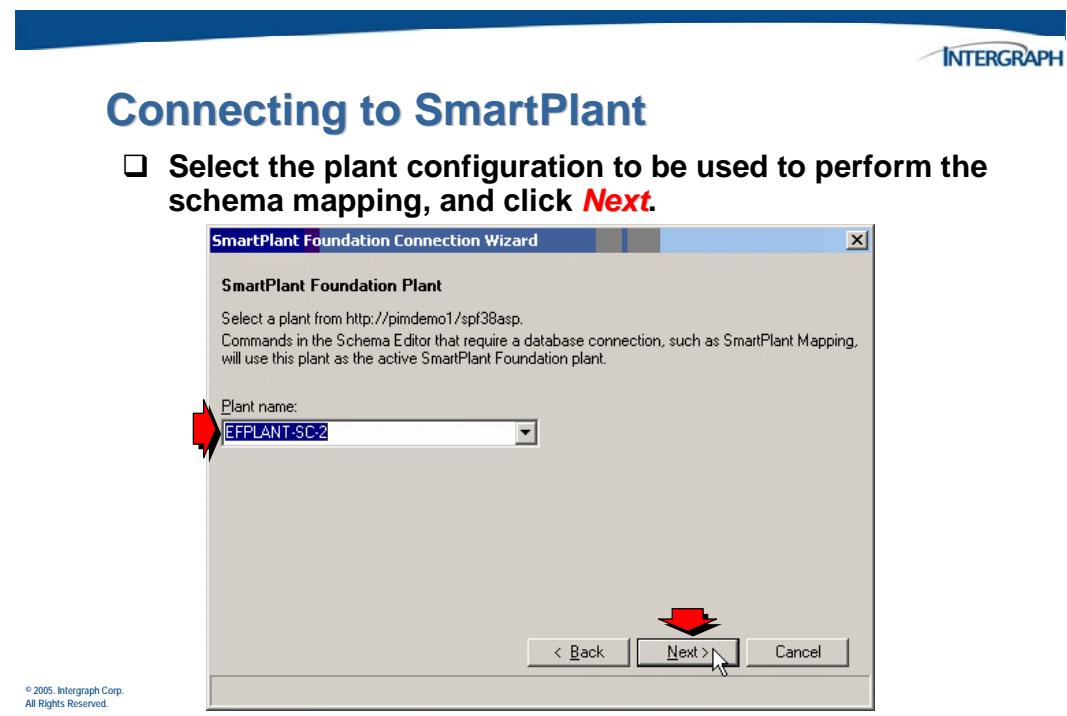
Connecting to SmartPlant

- Enter a valid SPF **User name** and **Password** to log on to the SPF server from the schema editor.

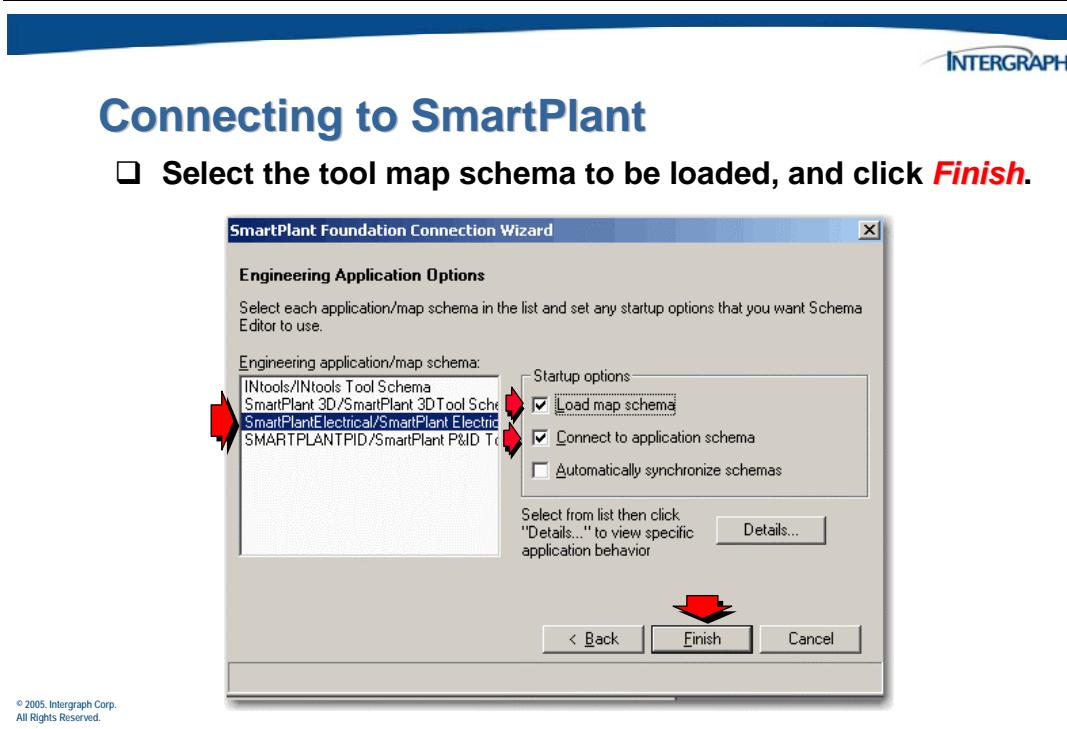


© 2005, Intergraph Corp.
All Rights Reserved.

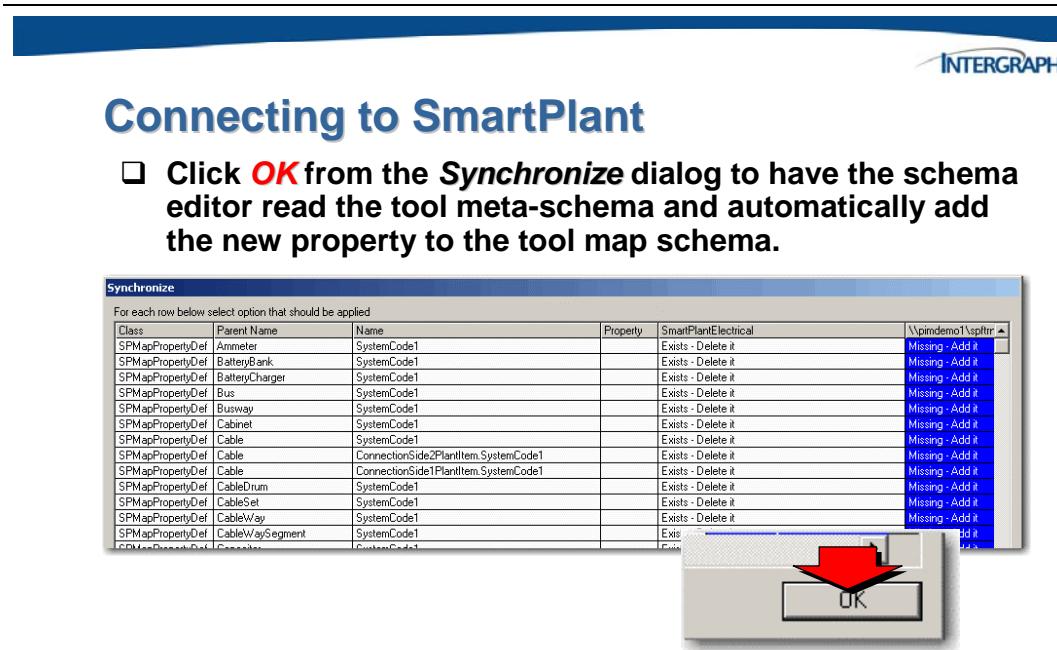
Select the SmartPlant Foundation plant configuration that you will use for schema extension and mapping.



The Schema Editor will find the SmartPlant schema and all **registered** tool map schemas.



After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema. The differences between the two schemas are displayed in the **Synchronize** dialog.



© 2005, Intergraph Corp.
All Rights Reserved.

During the connection process, a parsing of the tool map schema file will occur and the metadata adapter will also connect to its metadata data store (application tool database) and generate a tool map schema based on its application metadata. In the above example, you will note all the instances of **SystemCode** have been parsed for many of the SmartPlant SPEL classes.

11.1.3 Mapping the New Property

So far in this chapter, we have created a new property called **SystemCode** with a property type of *string* and added it to our tool database. Next, we used the metadata adapter to synchronize our tool meta data (the tool database) with the tool schema.

In some cases, the next step would be to add the new property to the SmartPlant schema so that the properties in the tool schema and the SmartPlant schema can be mapped.

However, in some cases, like this one, we may have already created the new property in the SmartPlant schema when we add the property to another authoring tool and mapped the property from that tool's schema. In this case, the new properties we added to the SmartPlant Electrical database already exist in the SmartPlant schema, because we added them when we mapped the same properties from SmartPlant P&ID.

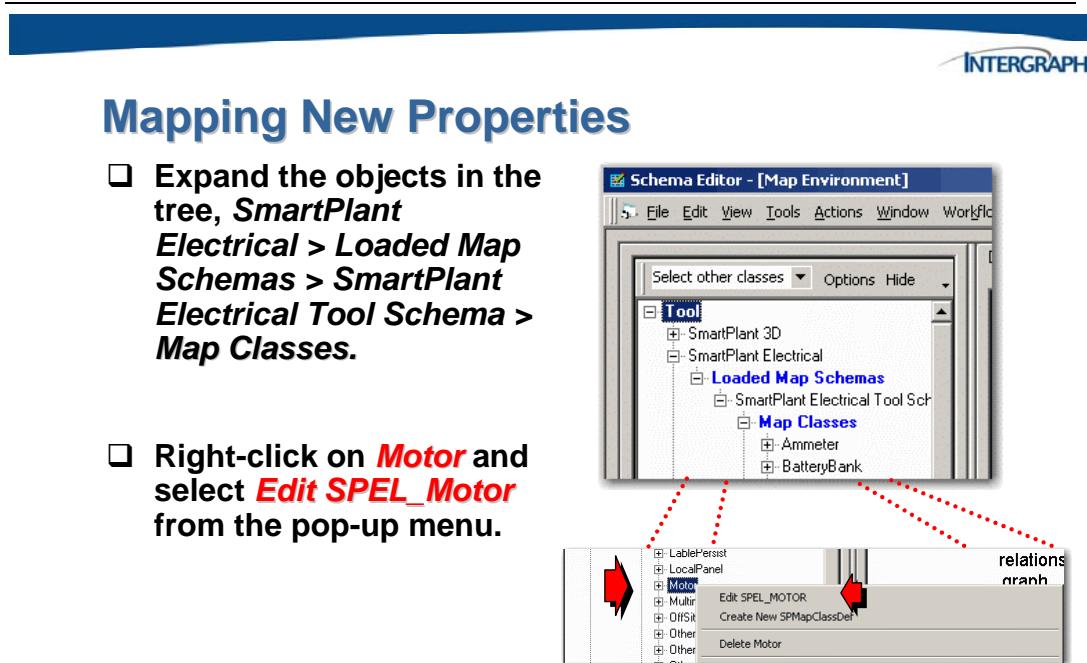
In these cases, we can skip the steps for creating the property in the SmartPlant schema and start mapping the tool schema to the SmartPlant schema.



Mapping New Properties

Map Property in
Tool Map
Schema

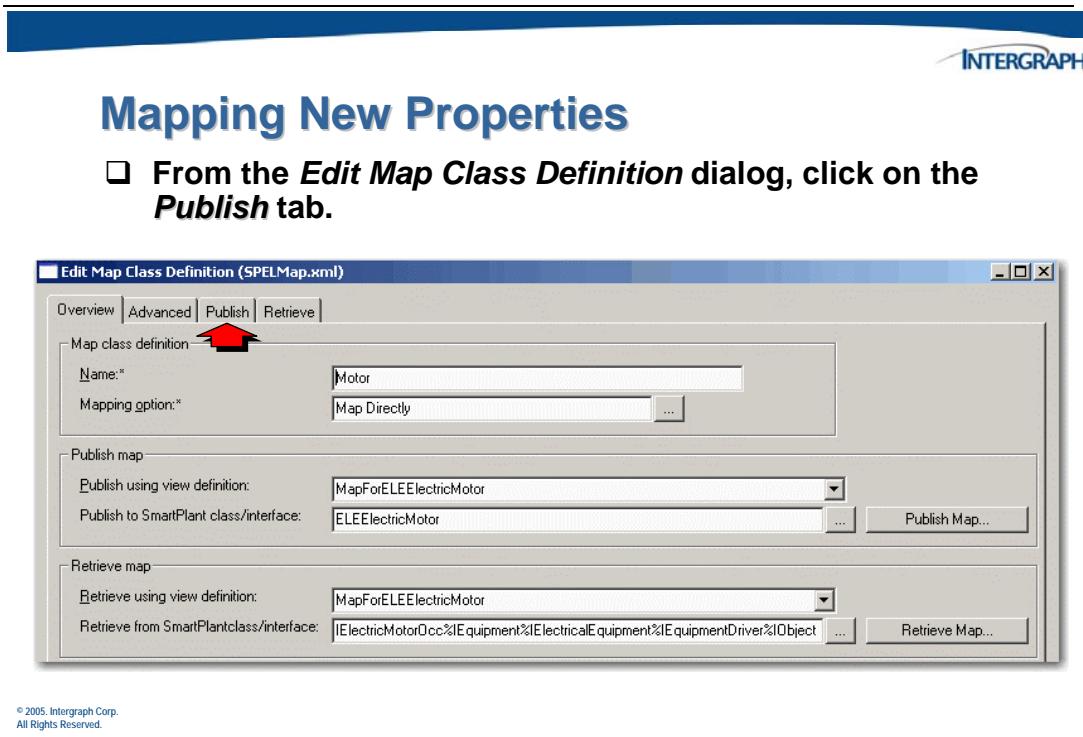
Use the Map Environment window to map the new property in the SmartPlant SPEL tool map schema to the existing property in the SmartPlant Schema that we created when we mapped the property from SPPID. When the tool map schema is expanded, it shows the map classes, map enumerated lists, map unit of measure lists and map relationship definitions for that tool map schema.



© 2005, Intergraph Corp.
All Rights Reserved.

Locate the existing **Motor Map Class** in the tree.

The *Edit Map Class Definition* dialog will be displayed.



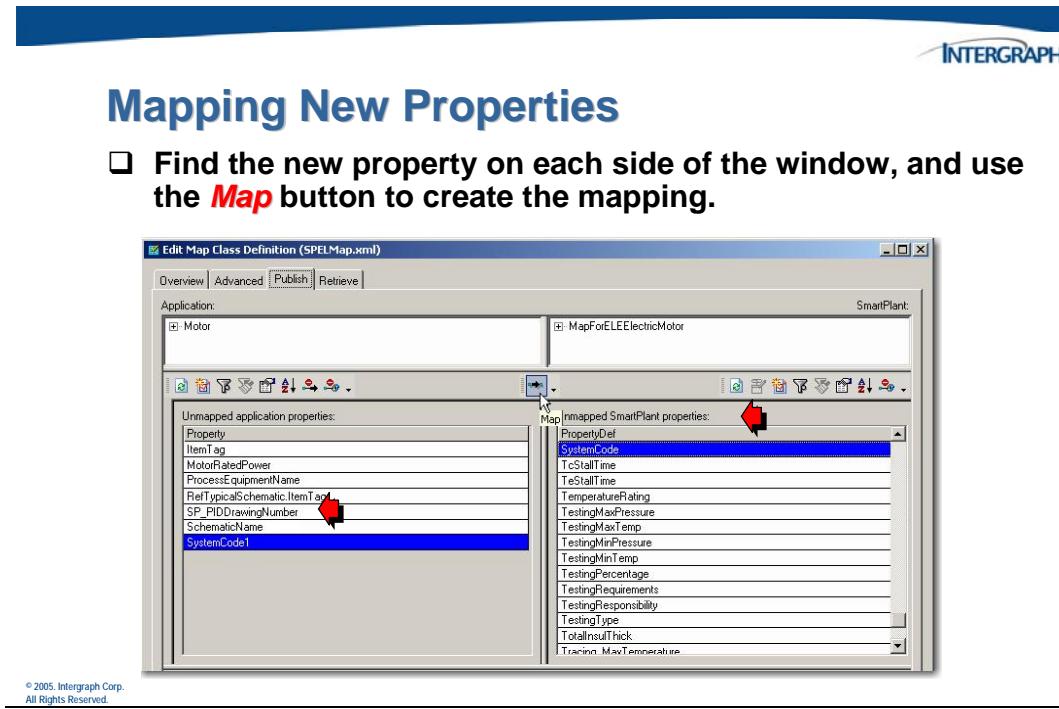
When the Publish tab is selected, the contents of the tool map schema will be displayed on the left side of the window and the SmartPlant schema contents will be displayed on the right side.

The Schema Editor will create a dynamic View Definition that will be used to display the SmartPlant schema properties. Selecting the mapped class will show all of the property definitions contained in the view definition.

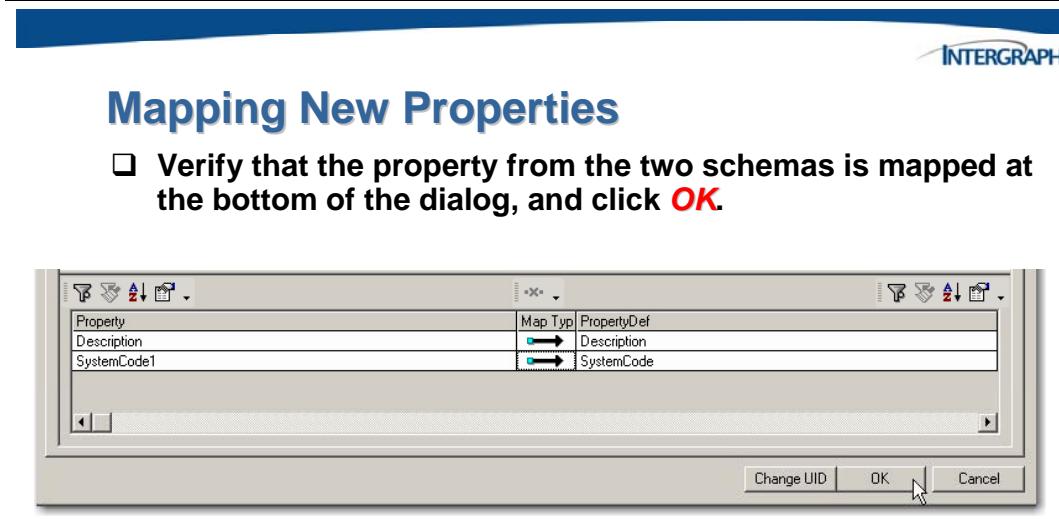
Note:

- If you have not already created the new property in the SmartPlant schema, you must create it now before you can map it.

Select the properties from the *Application* and *SmartPlant* sections of the dialog box, and use the **Map** button to link the two properties in the map file.



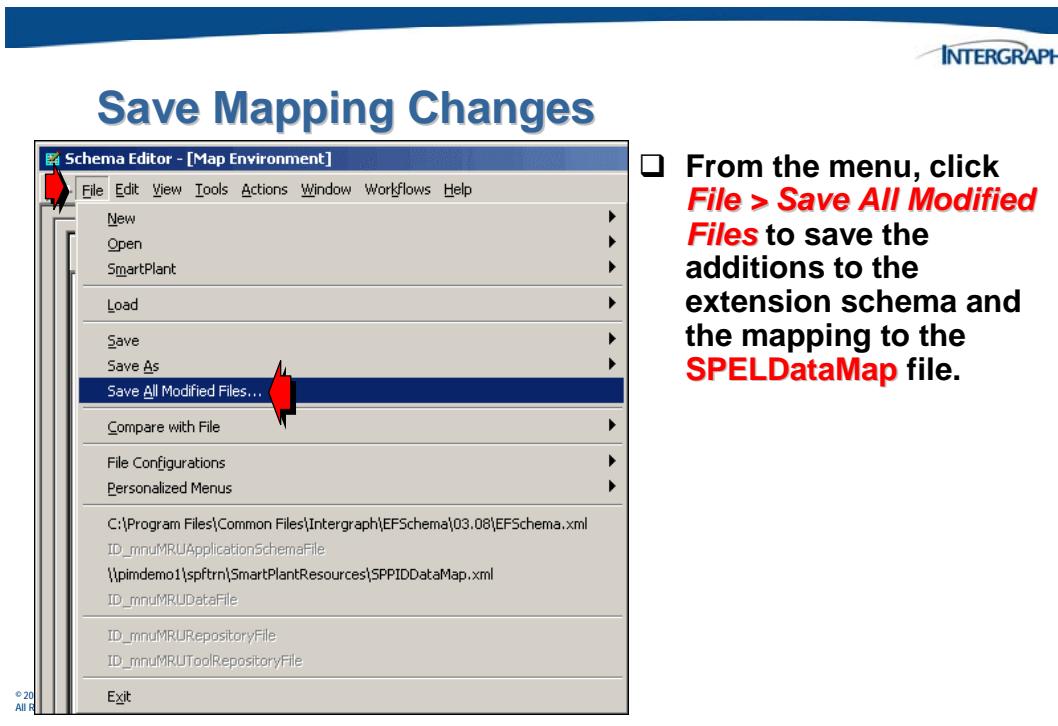
The results of the mapping will be shown in the bottom control. The **Unmap** button will delete the mapping relationship for any of the selected rows in this control.



Since multiple SPEL classes inherited the **SystemCode** property, you can complete additional mapping at this time. The System Code property will automatically appear for any class def that realizes the IPBSItem interface.

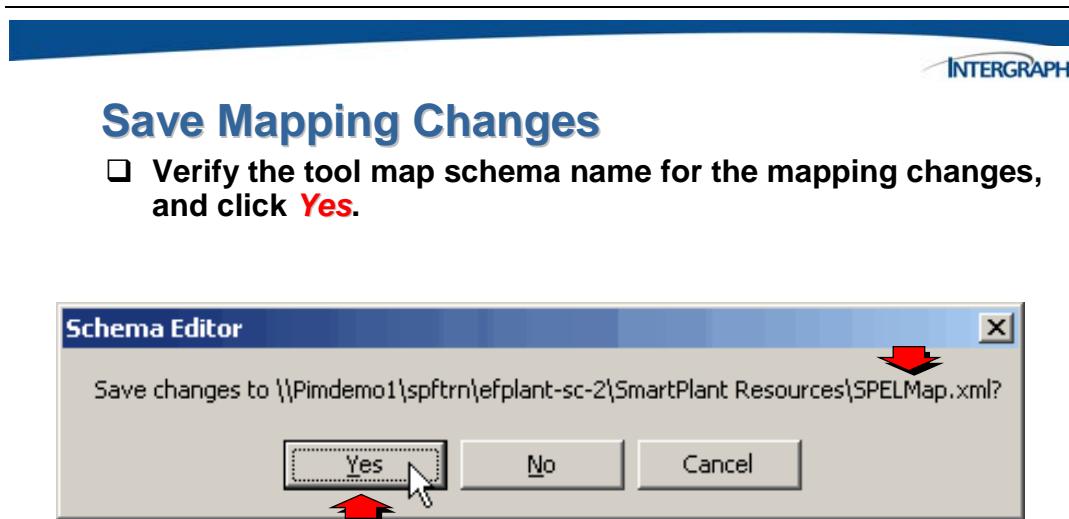
11.1.4 Saving Mapping Changes

At this point, the additions to the tool schema and the mapping information is stored in memory and will need to be saved.



- From the menu, click ***File > Save All Modified Files*** to save the additions to the extension schema and the mapping to the **SPELDataMap** file.

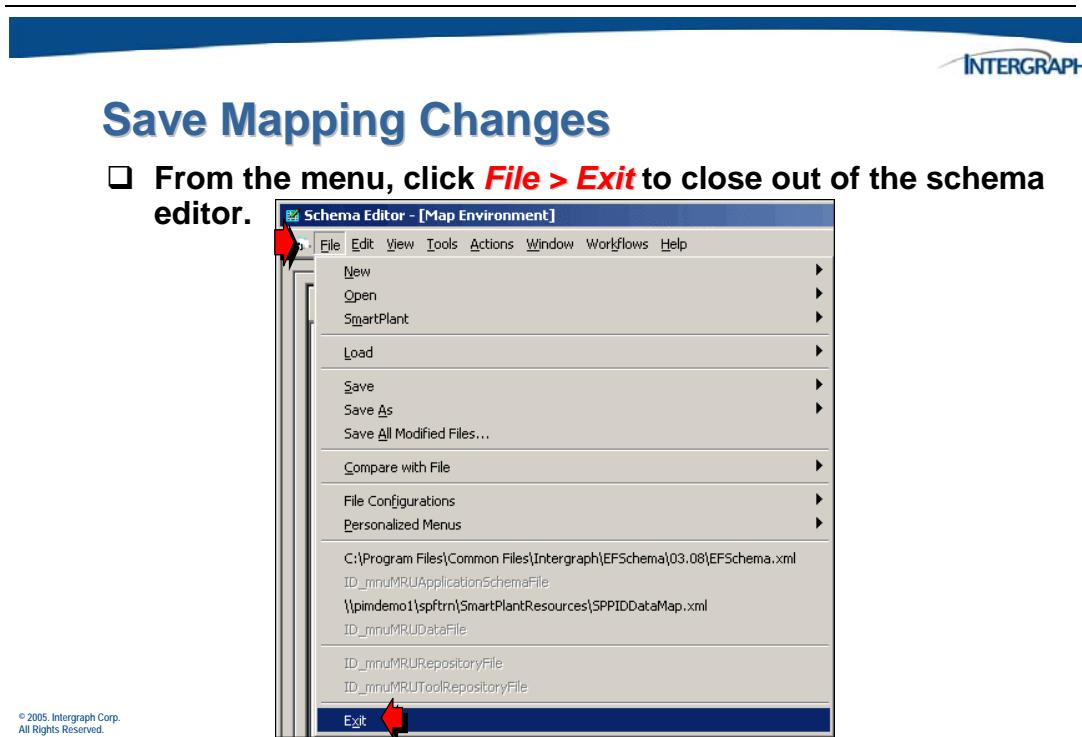
You will be prompted to save the changes to the tool map schema.



© 2005, Intergraph Corp.
All Rights Reserved.

Notice that we were not prompted to save the SmartPlant schema extension file, because we made no changes to the SmartPlant schema. All the changes we made, including the mapping information, is stored in the tool schema file.

Once the schema file has been saved exit the Schema Editor.



The schema files have been copied to a temporary location in order to perform the additions and mapping. The last step is to upload these files to the schema location on the SmartPlant Foundation server if changes were made. However, in this case, we made no changes to the SmartPlant schema, so there is no reason to upload the files.



11.2 Activity 1 – Adding and Mapping a Simple Property

The goal of this activity is to give you the opportunity to create a custom property in the SmartPlant SPEL application meta data, the SmartPlant SPEL tool map schema and the SmartPlant schema. You will then perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant engineering Manager > Data Dictionary Manager** to start the *SmartPlant Data Dictionary Manager*.

Click **File > Open Database** and from the dialog that appears, choose **SmartPlant Electrical** from the **Application** field. Be sure that **EFPLANTSC2** is selected in the **Available plant structures** field.

3. Open the Plant Item database table.
 - Click **Plant Item** in the Database Tables window.
 - Select **Edit > Add Property** from the menu.
 - In the *Add Property* dialog box, enter the following information:

• Name –	SystemCode1
• Display Name –	EF System Code
• Data Type –	String
• Maximum Length –	8
• Display to User –	Yes
• Use for Filtering –	Yes
• Category –	Miscellaneous
• Depends on –	None
 - Click **OK** to close the *Add Property* dialog box.
4. Save the changes to the SmartPlant Data Dictionary Manager.
 - Click **File > Save**.

- Click **File > Exit.**
5. Click **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
6. Open the schema XML files for analysis.
- Click the **File Configurations** button and select **Open Configuration...** from the *Schema Editor* Workflows dialog
 - Select the **EFSchema.cfg** file and click **Open** (Path is C:\Program Files\Common Files\Intergraph\EF Schema\03.08)
8. View the open schema file using the **Tree/Drag-Drop UML** view.
- On the *Workflows* dialog box, click the **View** button beside the **Another Schema File** button.
 - When the *View Schema* dialog box appears, select the **Tree/Drag-Drop UML** option and click **OK**.
9. In the base classes tree, expand the list of classes and then the ELEElectricMotor class. Expand and display the **Realized Interface Definitions**.
- List the Realized Interface Definitions below:

10. Repeat this for the following classes and list their **Realized Interface Definitions**.

- Class: ELEWire
- Interfaces: _____

- Class: ELECable
- Interfaces: _____

-
-
-
- Class: ELEInstrument
- Interfaces: _____

11. List the common interfaces for all classes from steps 9 and 10 above.

Use one of these interfaces as the expose relationship for any new properties to be added to the schema.

12. Exit and re-start the *Schema Editor* by selecting **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor**.
13. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant**.
14. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
- When prompted for a SmartPlant Foundation login, use admininuser with no password.
 - Choose the EFPLANT-SC-2 SmartPlant Foundation plant to work with
 - Click Next
 - Select the tool map schema to be loaded, SMARTPLANTElectrical/SmartPlant SPEL Tool Schema
 - Enable the Load map schema toggle
 - Enable the Connect to application schema toggle
 - Click Finish

15. When the *Synchronize* dialog displays, confirm that the new property values have been imported by the meta data adapter and click **OK**.
16. Map the *SystemCode1* property:
 - Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant SPEL Tool Schema entries are displayed.
 - Expand the **Map Classes** and right-click on the *Motor* entry in the tree and choose **Edit SPEL_MOTOR** from the dynamic menu.
 - Select the **Publish** tab in the *Edit Map Map Class Definition* dialog. Expand the tree for both the application/tool map schema and the SmartPlant schema in the upper control.
 - Highlight the *SystemCode1* MapProperty in the **Unmapped applications properties** control and the *SystemCode* MapProperty in the **Unmapped SmartPlant properties** control in order to perform the schema mapping.
 - Make sure that **SystemCode1** is highlighted in the middle control in the application section and that **SystemCode** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPEL tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *SystemCode1* maps to *SystemCode* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Class Definition* dialog box.
17. Repeat the mapping for the following classes:
 - Instrument**
 - Cable**
18. Save the changes to the all schema files.
 - From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to the modified tool schema file.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **No**.

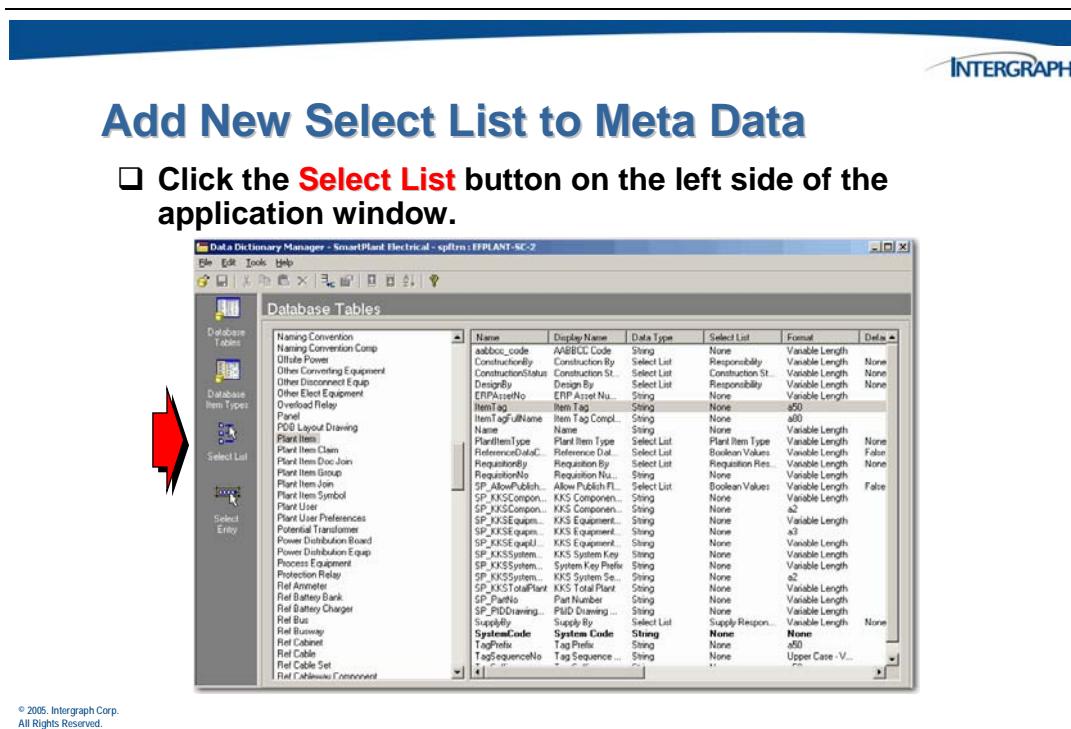
19. When you are finished with this activity, you may take a short break until the other students have finished.

11.3 Adding a New Select List/Enum List

In the last section, the task of extending the SmartPlant SPEL meta data/SmartPlant Schema and adding a new simple property was demonstrated. A simple property is a property definition that uses a simple property type such as a string property type.

Another property type that is useful in SmartPlant is the **Select List**, which is a picklist. This is a more complex type of property. The select list will equate to an **Enumerated List** in the tool map schema and the SmartPlant schema.

To add a new property to SPEL that uses a select list, begin by starting the **Data Dictionary Manager**.



You will need to define a property that has a select list associated with it, but first, you need to create the select list that will be associated with the property.

You can add a new select list by typing the name of the list in the blank row at the bottom of the **Select List** table.

INTERGRAPH

Add New Select List to Meta Data

- Scroll to the bottom of the **Select List** table, and in the empty row at the bottom of the table, enter the name of the new select list.
- Select the **EngSys** row, and click the **Select Entry** button.



Temperature Class	None
Temperature Detector Type	None
Temperature Rise	None
Terminal Type	None
Text Orientation	None
Thread Size	Thread Type
Thread Type	None
Transformer Connection Group	None
Tray Fill Validation Method	None
Tray Shape	None
Trip Class	None
Unit of Measure	None
Update Flag	None
Vertical Alignment	None
Volts Per Hz Setting	None
Winding Voltage Source	None
Windings Material	None
Wire Type	None
Withdrawable Cubicle Const	None
EngSys	None

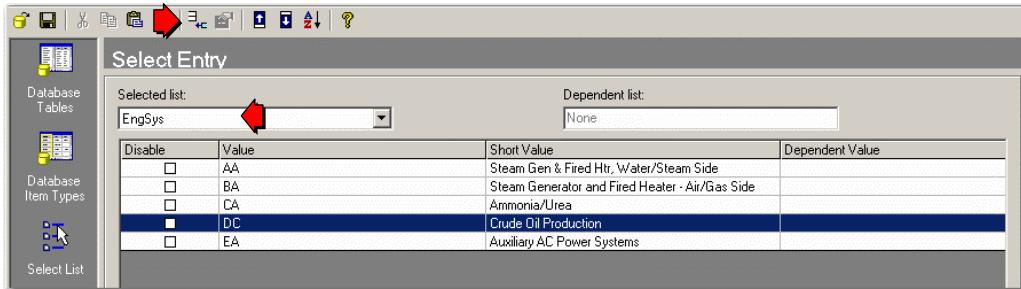
© 2005, Intergraph Corp.
All Rights Reserved.

All that is required to define a new select list is the name of the select list. To define entries for the new select list, click the **Select Entry** button. Add the first select entry to the list by typing it in the first row of the table. Click the **Add Row** button at the top of the window to add a new empty row and continue adding your new entries until you have defined them all.



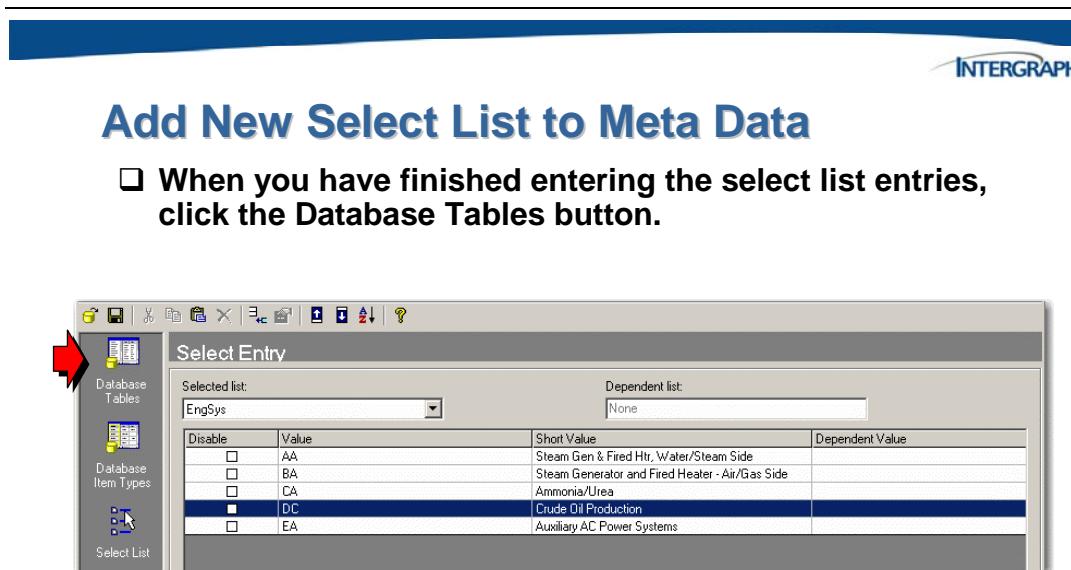
Add New Select List to Meta Data

- The select list that was highlighted when you clicked the **Select Entry** button appears in the **Selected list** box.
- In this table, define the options that will be available in the enumerated list.



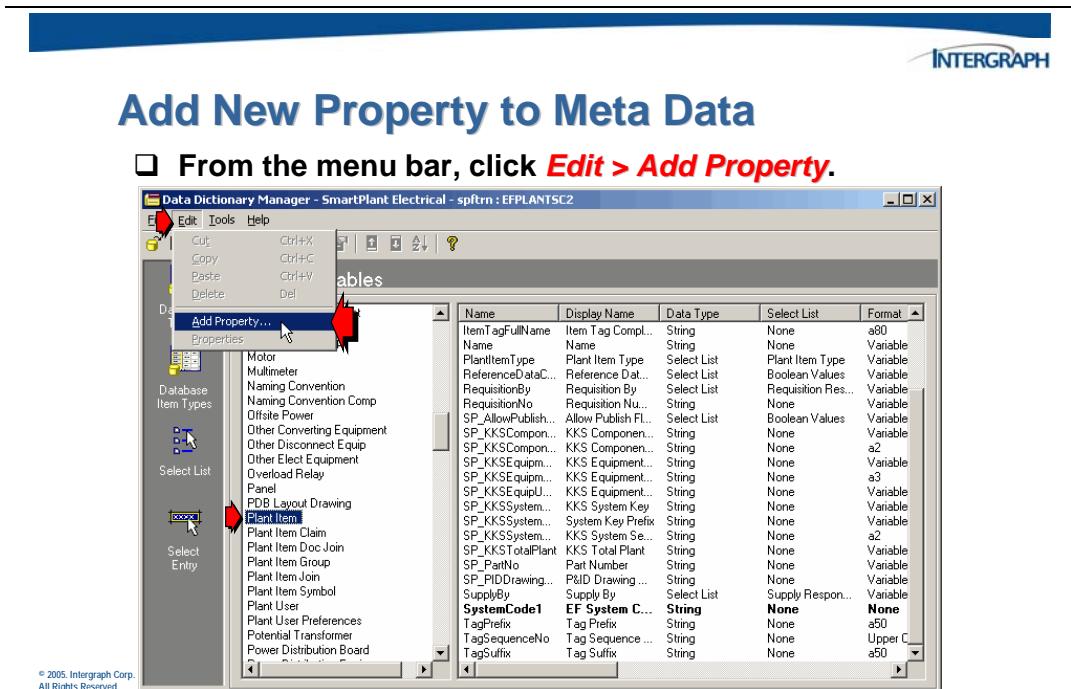
© 2005, Intergraph Corp.
All Rights Reserved.

When you have defined all your entries, click the **Database Tables** button to return to the screen where you will add your new property that will use the select list you just created.



© 2005, Intergraph Corp.
All Rights Reserved.

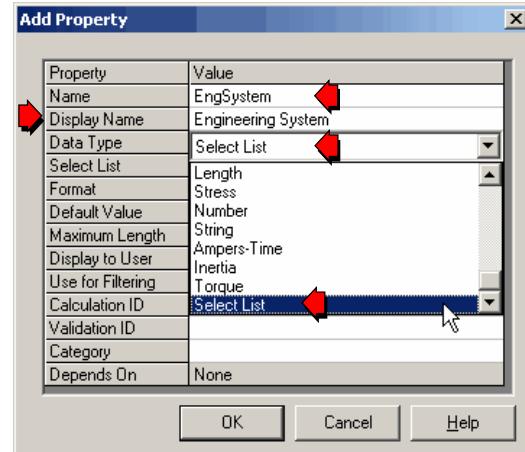
To create the property, select *Plant Item* in the list of database tables, and click **Edit > Add Property**.



Add the new **EngSystem** property to the *Plant Item* table. Choose **Select List** as the data type and then choose the appropriate list in the **Select List** box.

Add New Property to Meta Data

- In the **Add Property** dialog box, enter the **name** and **display name** for the property.
- In the **Data Type** list, choose **Select List**.

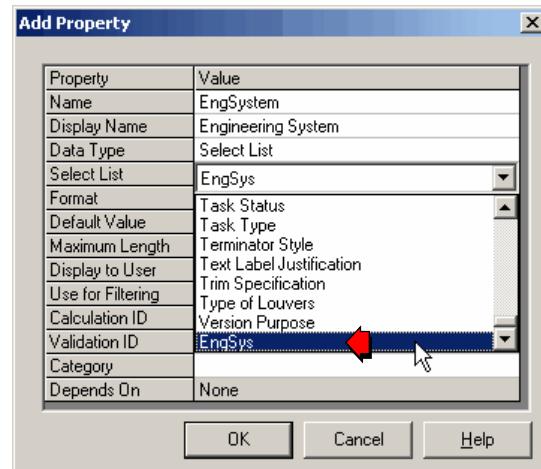


© 2005, Intergraph Corp.
All Rights Reserved.

The new property will use the custom select list, **EngSys**, defined earlier.

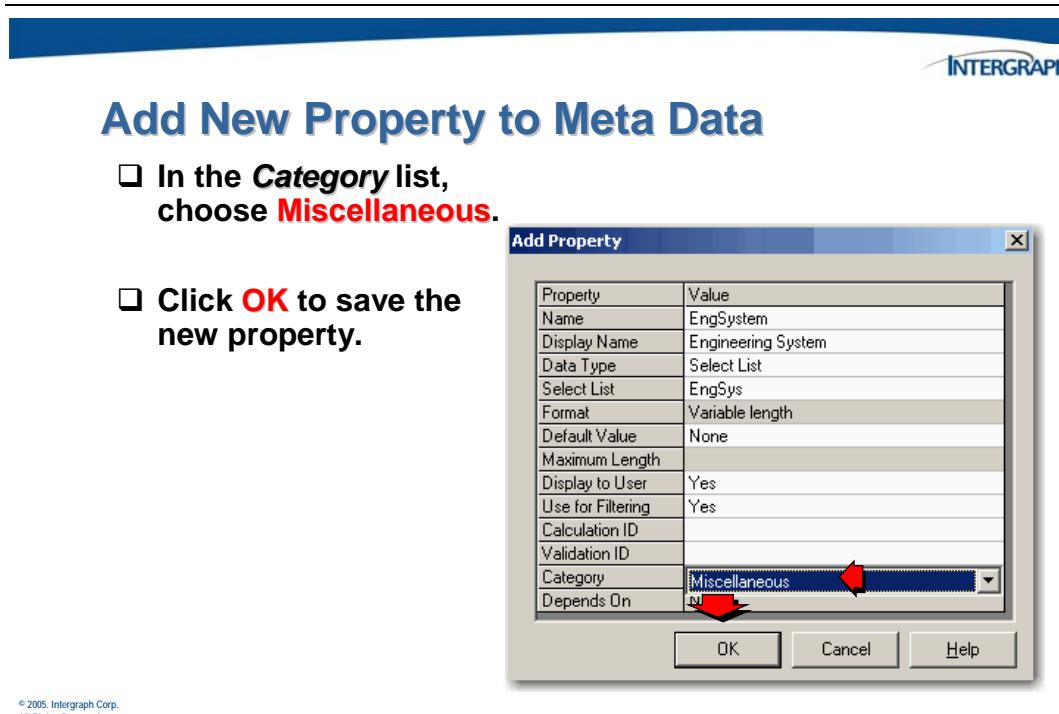
Add New Property to Meta Data

- In the **Select List** list, choose the name of the select list you want to associate with the property.



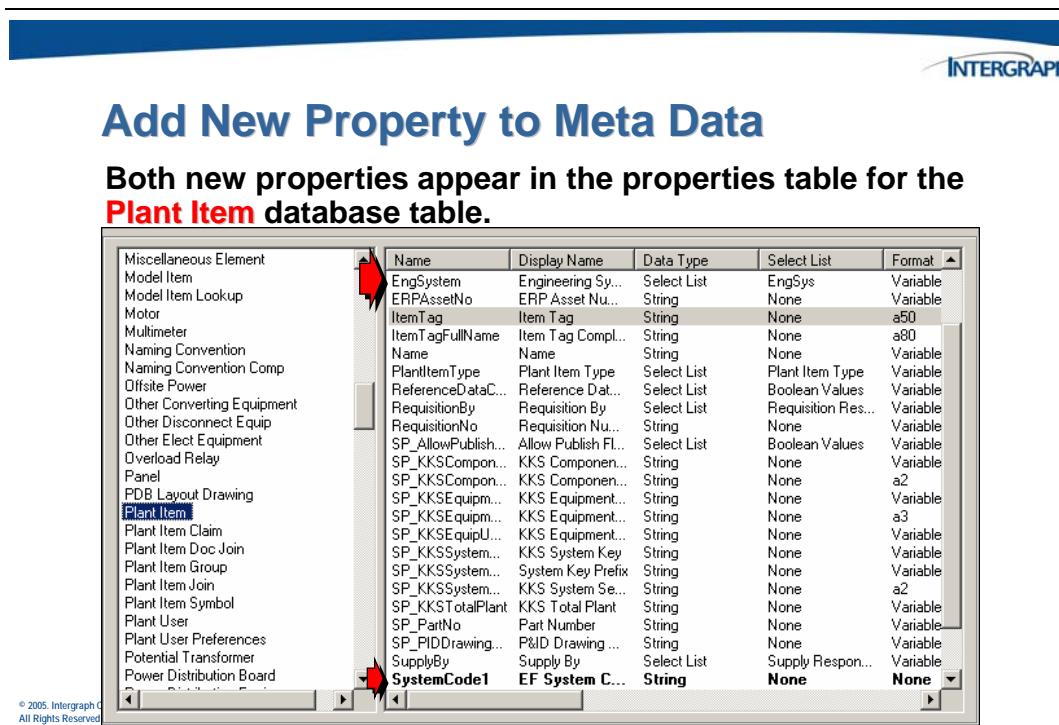
© 2005, Intergraph Corp.
All Rights Reserved.

Finish specifying the new property characteristics.

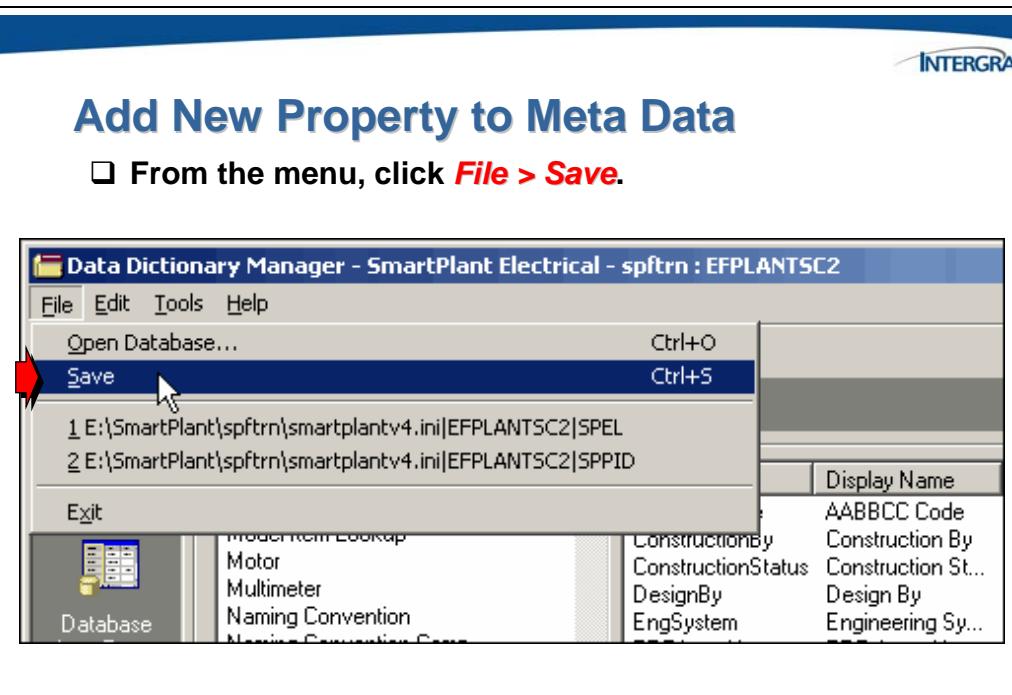


© 2005, Intergraph Corp.
All Rights Reserved.

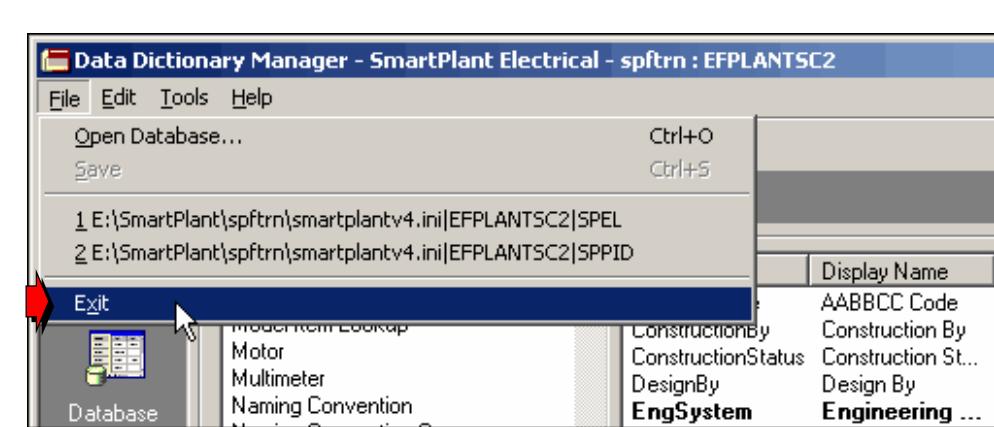
When you click **OK**, the list of properties for the *Plant Item* table updates with the new property.



After adding the property, save the changes to the SmartPlant SPEL meta data.



When you finish click ***File > Exit***, to close the Data Dictionary Manager.



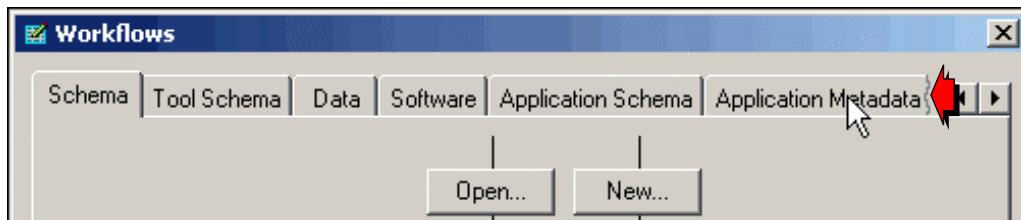
11.3.1 Adding Enum Extensions to the Tool Schema

Once the new property/select list has been added to the SPEL meta data, use the Schema Editor to add the new property to the tool schema file.

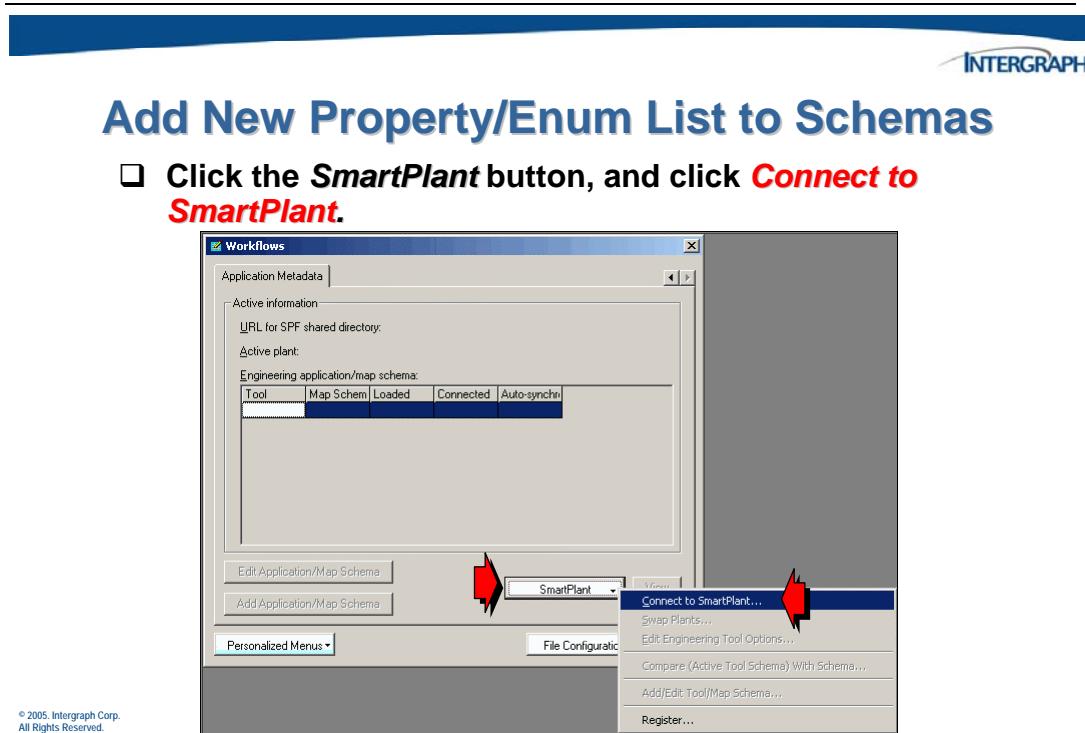
Start the Schema Editor and connect to SmartPlant.

Add New Property/Enum List to Schemas

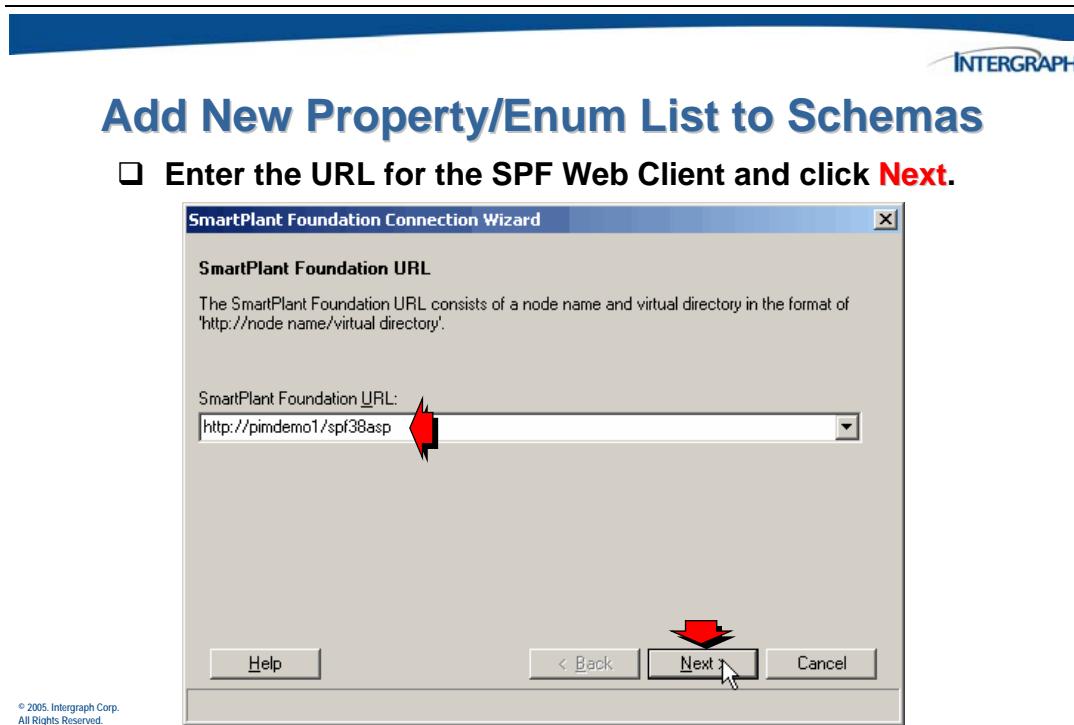
- Click on the **Application Metadata** tab on the Workflows dialog to connect to SmartPlant.



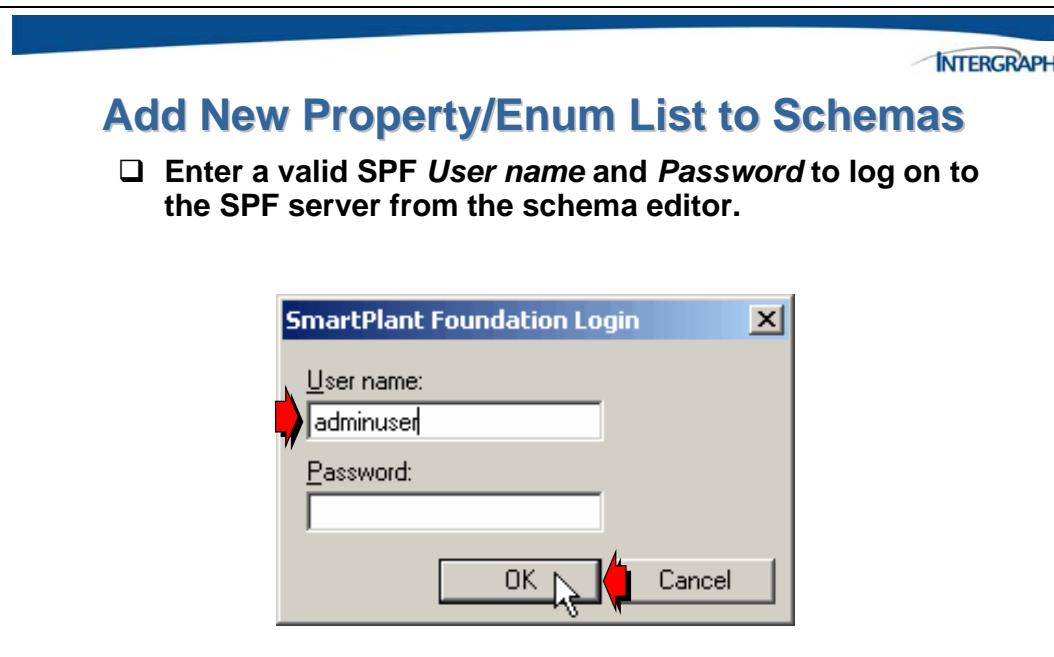
The *Application Metadata* dialog will appear.



The *SmartPlant Foundation Connection Wizard* will be displayed.

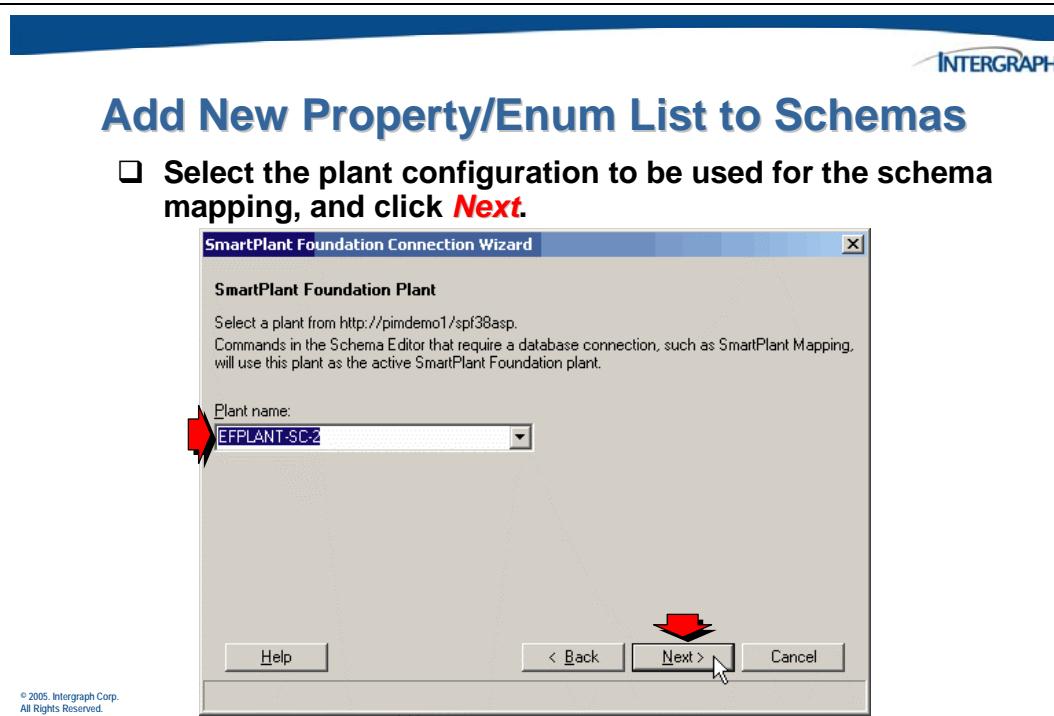


In the *SmartPlant Foundation URL* field, select the SmartPlant Foundation database with which you want to connect and click **Next**.



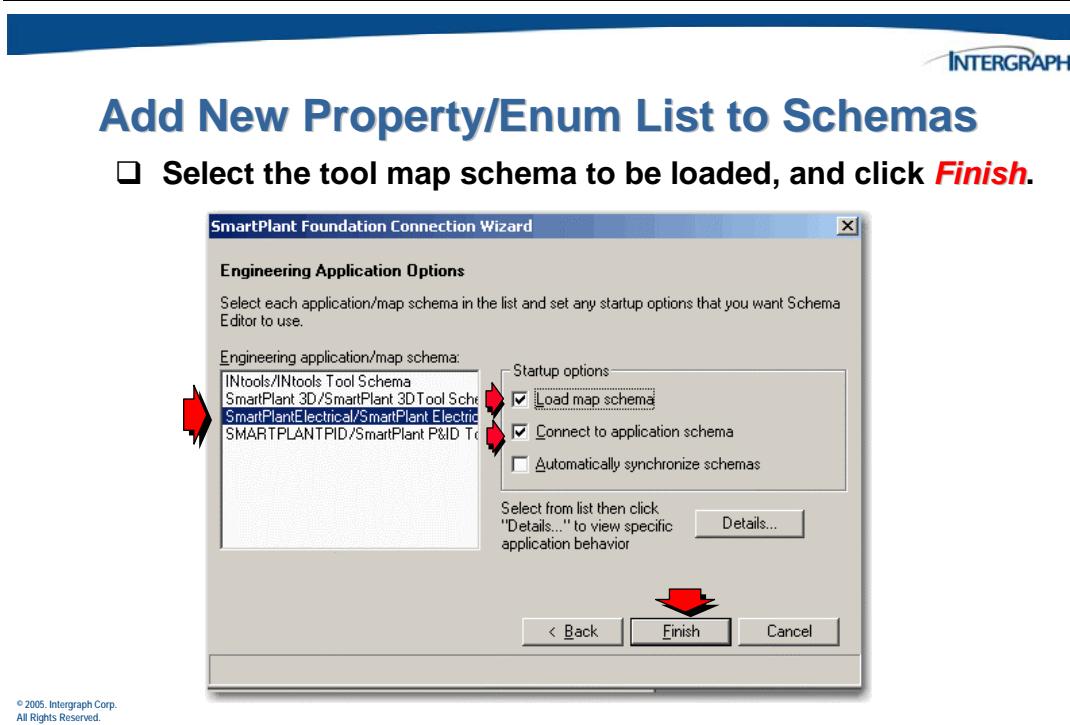
© 2005, Intergraph Corp.
All Rights Reserved.

Select the SmartPlant Foundation plant configuration that you will use for schema extension and mapping.



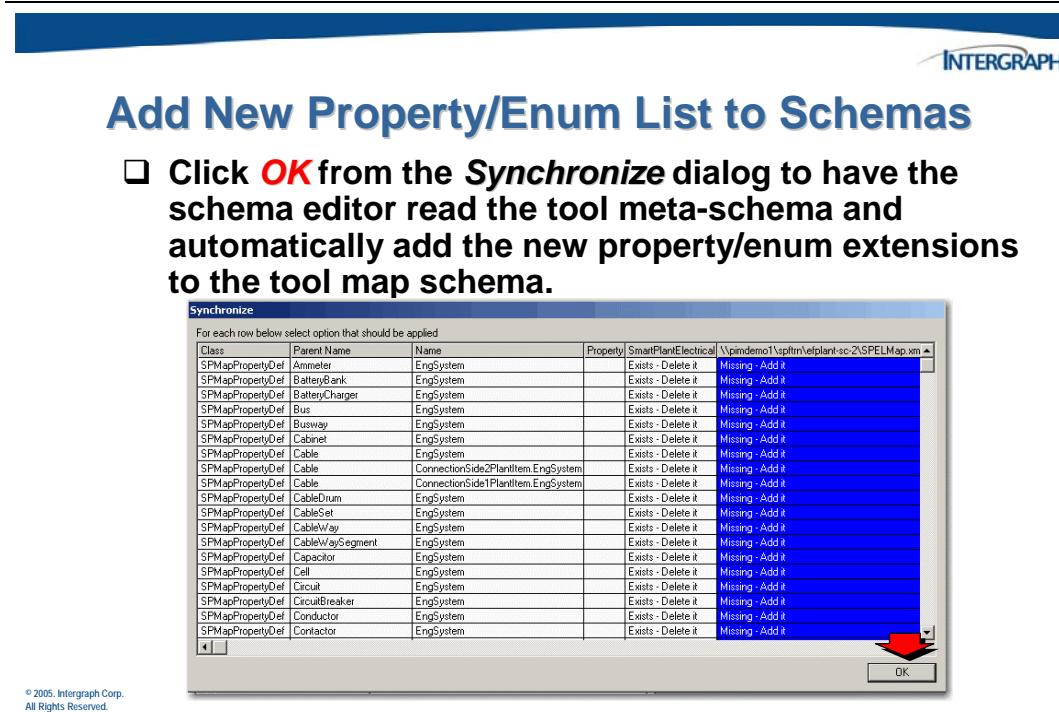
© 2005, Intergraph Corp.
All Rights Reserved.

The Schema Editor will find the SmartPlant schema and all **registered** tool map schemas.



After the metadata adapter generates a map schema based on its application metadata, a comparison is performed between the generated tool map schema and the parsed map schema.

The differences between the two schemas are displayed in the **Synchronize** dialog.



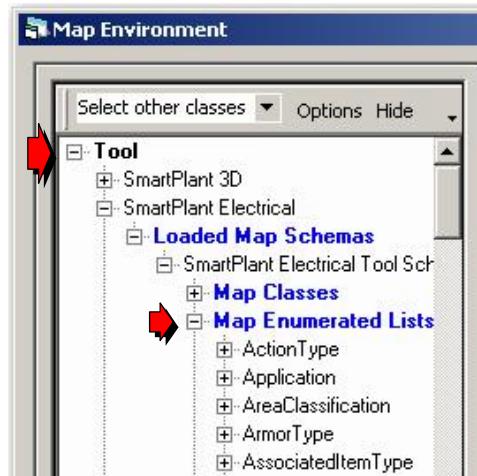
In the above example, you will note all the instances of the EngSystem SPMapProperty and the EngSys SPMapEnum have been parsed for many of the SmartPlant SPEL classes.

11.3.2 Mapping Enumerate List Entries

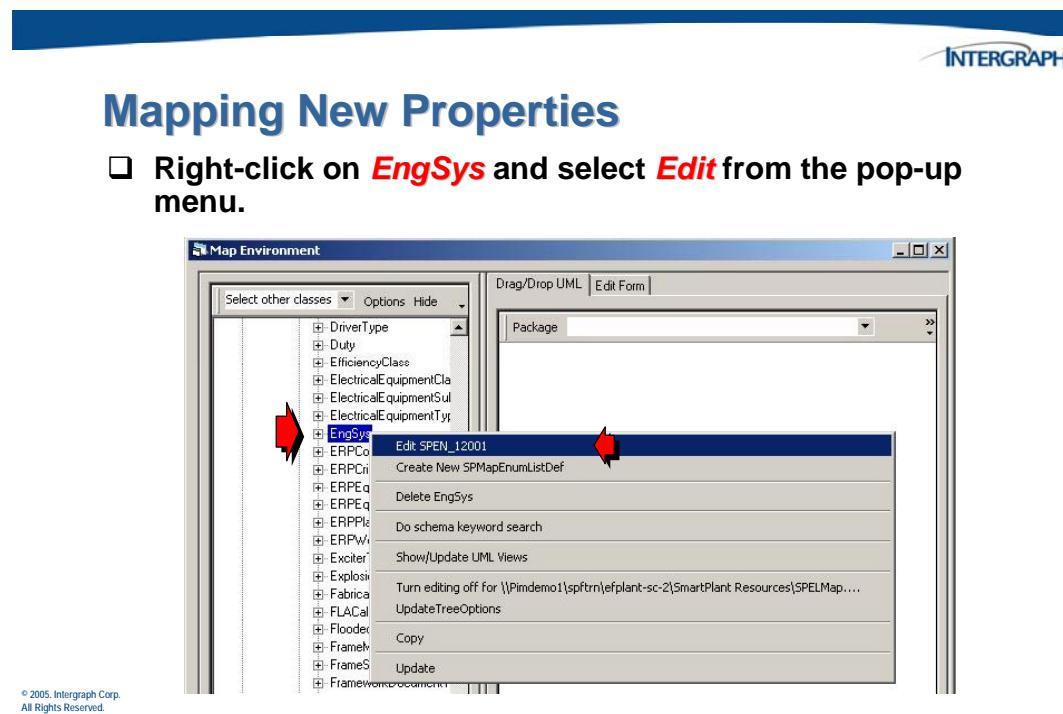
Next, we need to map the values available in the select list (enumerated list) to the enum enums as defined in the SmartPlant schema. Expand the tree in the Map Environment to show the Map Enumerated Lists under SmartPlant Electrical.

Mapping New Properties

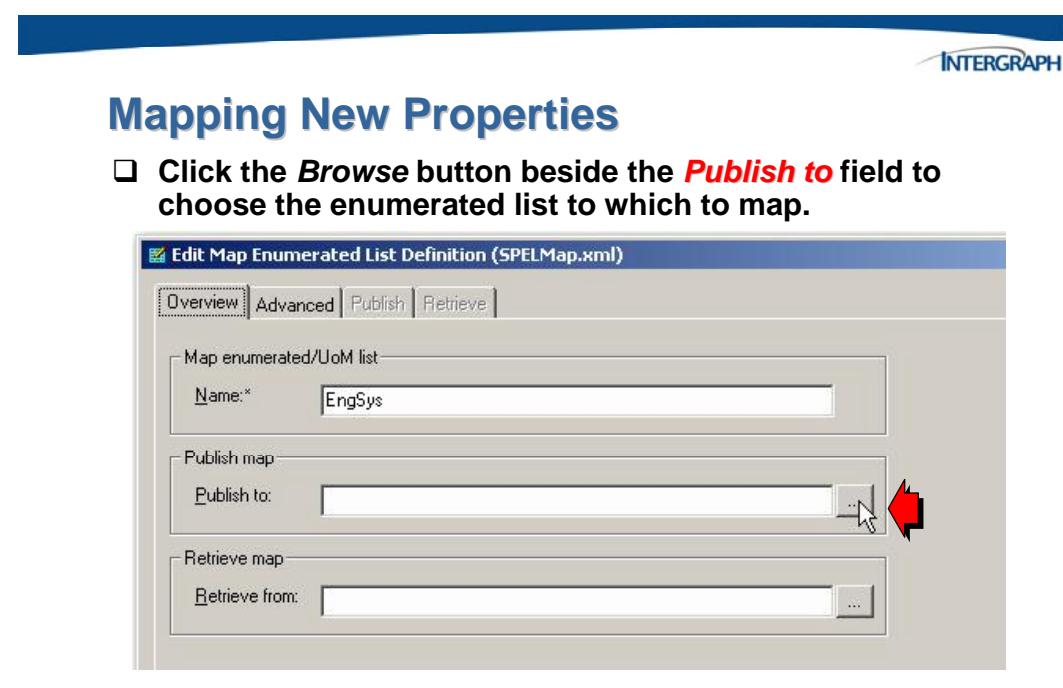
- Expand the objects in the tree, *SmartPlant Electrical > Loaded Map Schemas > SmartPlant Electrical Tool Schema > Map Enumerated Lists*.**



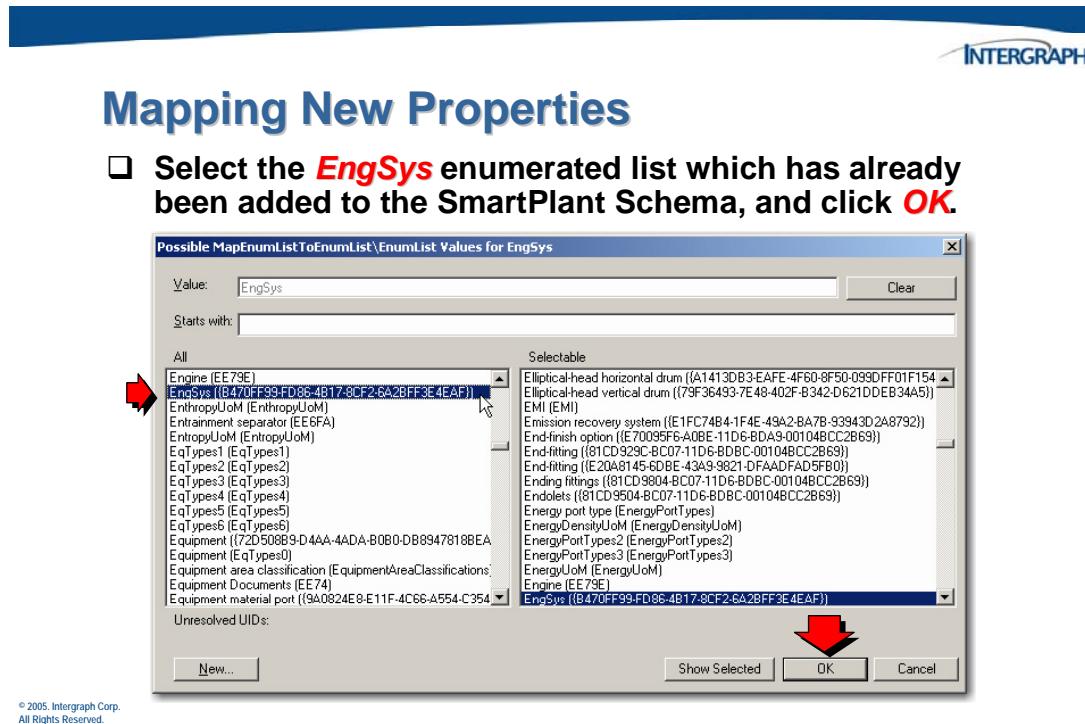
Find the *EngSys* select list that you created in the Data Dictionary Manager.



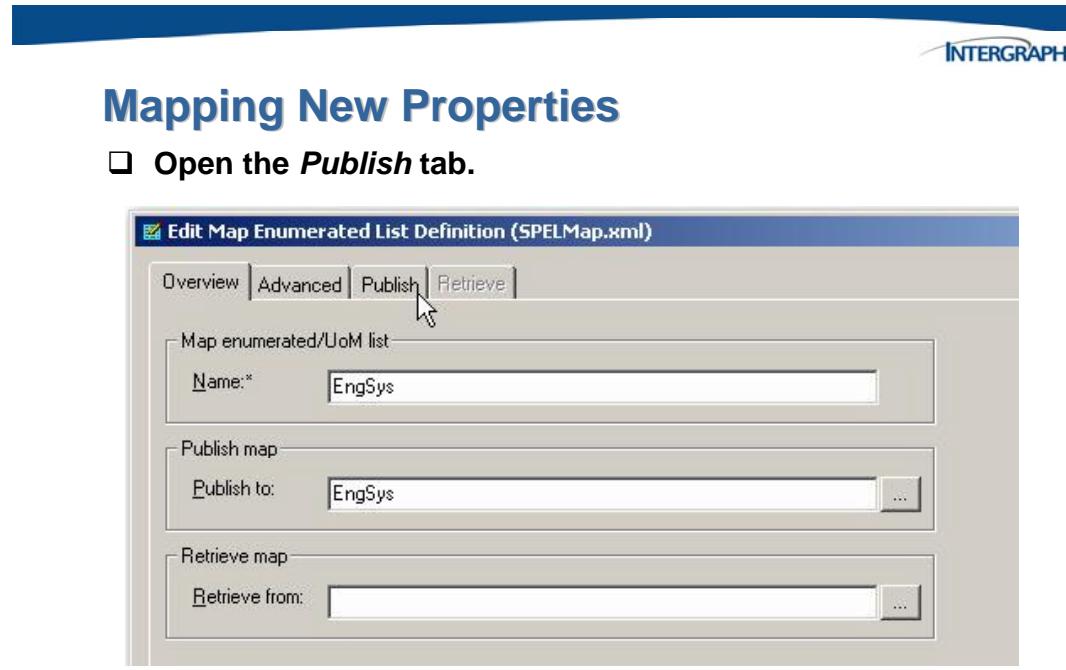
On the **Overview** tab, click the *Browse* button beside the **Publish to** field to select the enumerated list in the SmartPlant schema to which you want to map the new list.



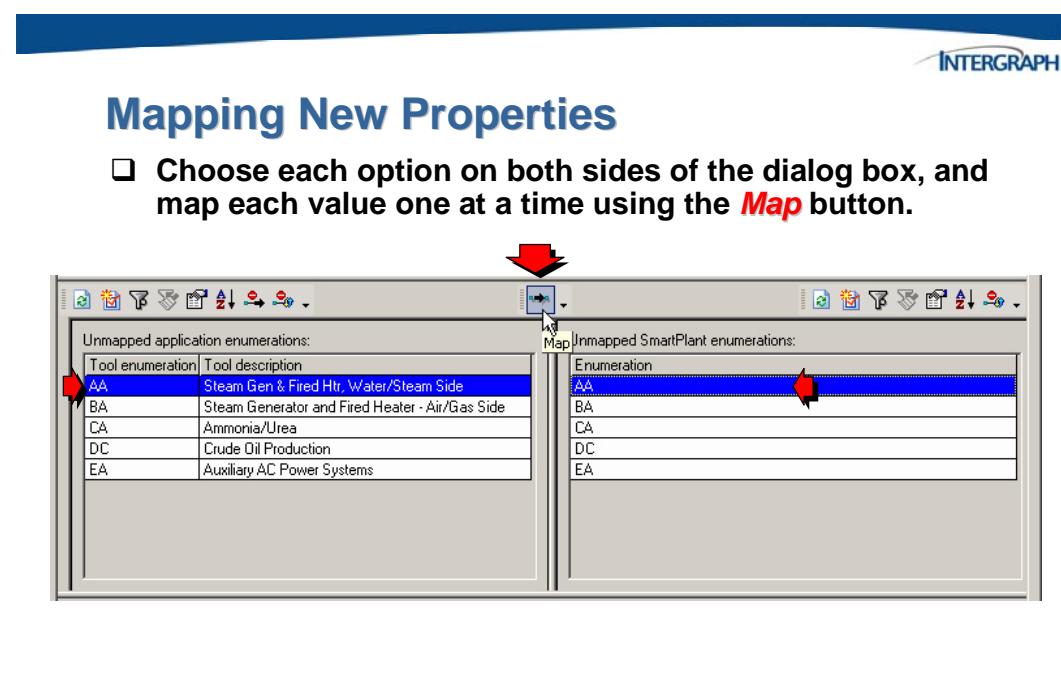
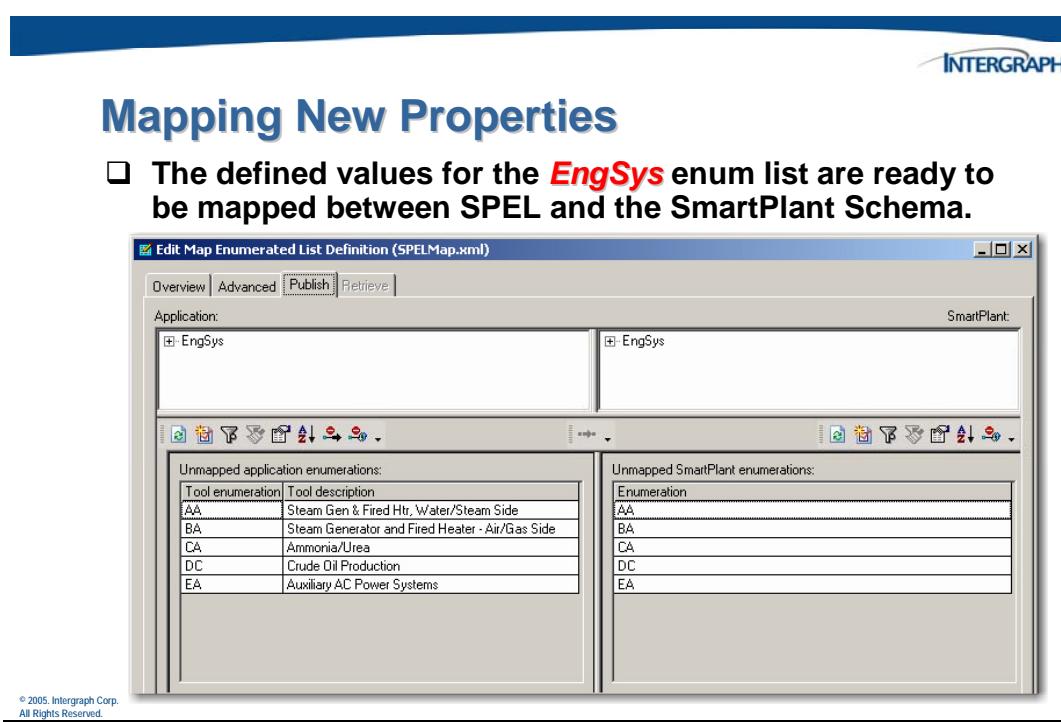
Find the *EngSys* enumerated list in the SmartPlant Schema, select it, and click **OK**.



The **Publish** tab is now available. Click to open the **Publish** tab.



On each side of the screen, you should find the entries of the enumerated list unmapped. To map the values, select one from the **Application** side and the corresponding value from the **SmartPlant** side, and then click the **Map** button.

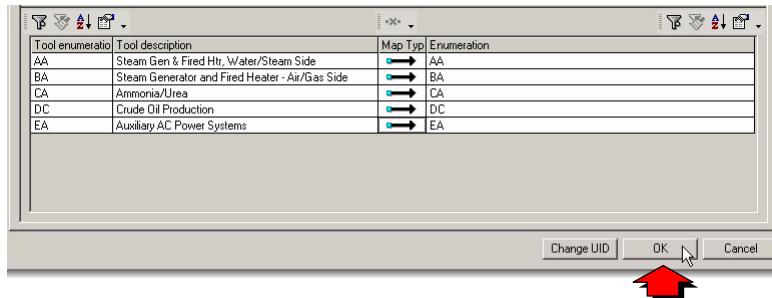


Verify that the mapping was completely successfully, and then click the **OK** button.



Mapping New Properties

- When you have finished mapping each enumerated list value, confirm the mapping in the bottom of the window.
- Click OK, when you are done.



11.3.3 Mapping the Complex Property

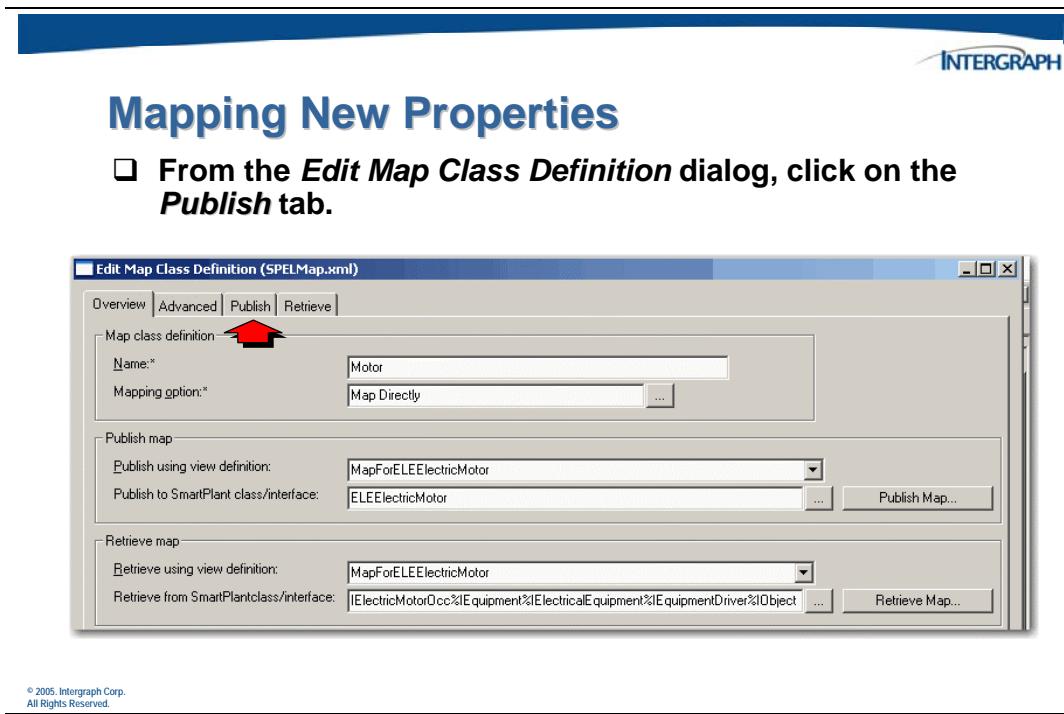
Use the Map Environment window to map the new complex from the SmartPlant SPEL tool map schema to the property in the SmartPlant schema.

Locate the existing **Motor Map Class** in the tree.

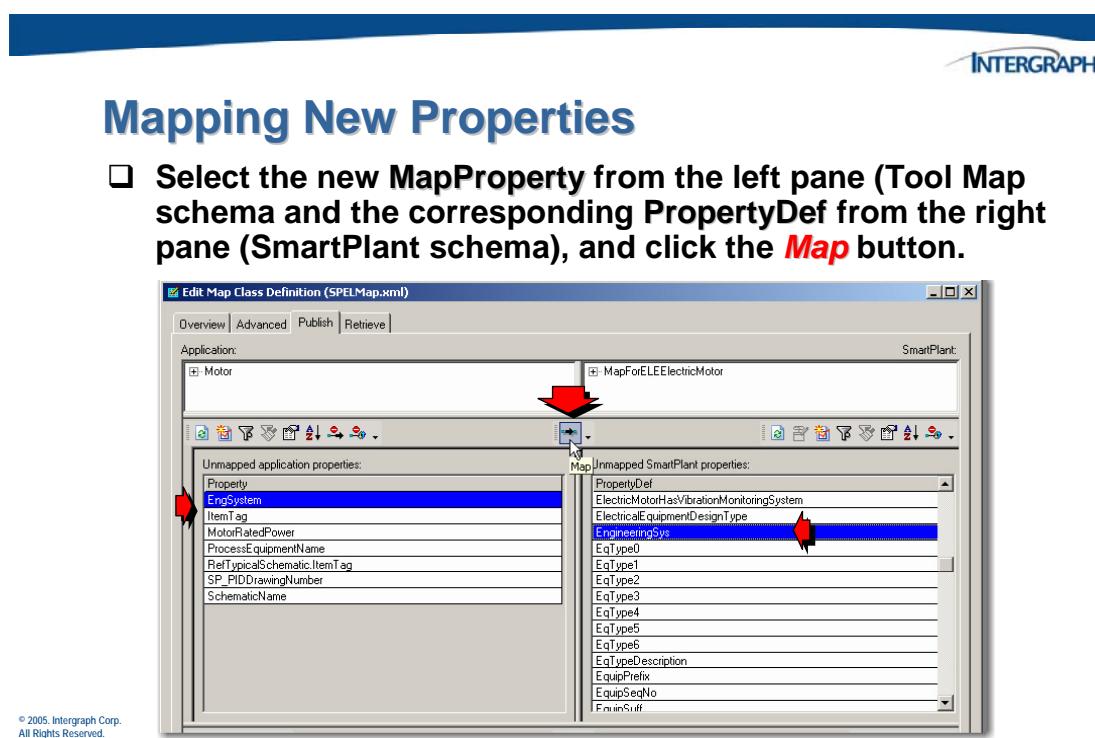


- Right-click on **Motor**, and click **Edit SPEL_MOTOR** from the pop-up menu.

The *Edit Map Class Definition* dialog will be displayed. Go to the **Publish** tab.

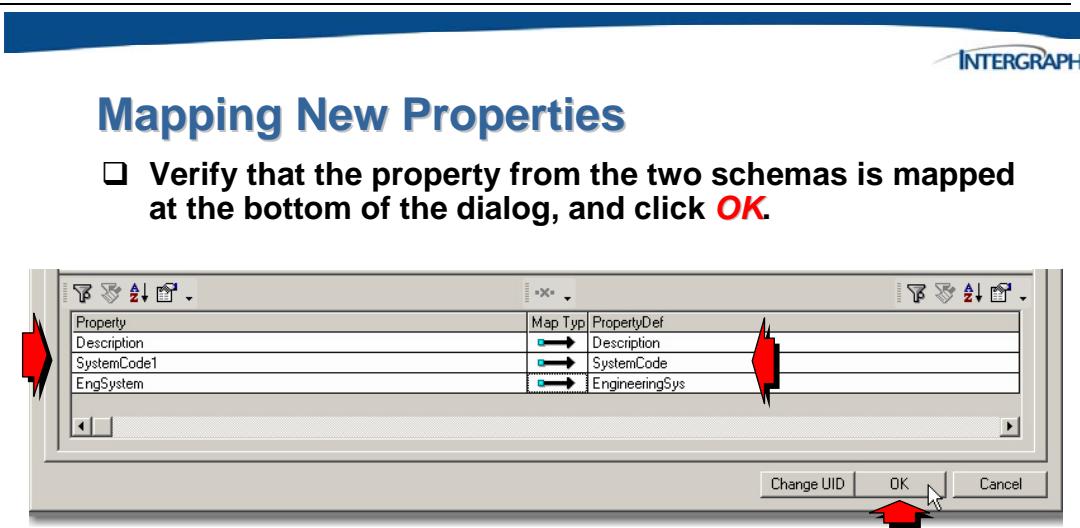


Select the property on both the left and right of the middle (unmapped) section of the dialog box, and click the **Map** button to create the mapping.



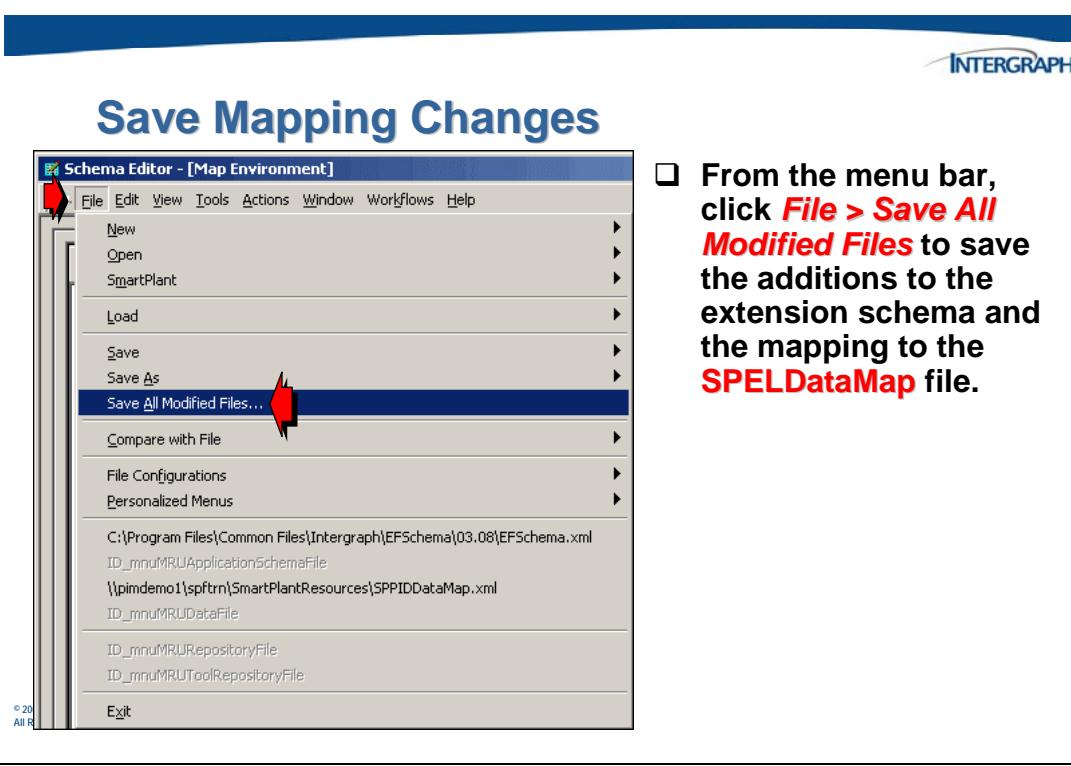
The **Map** button is used to map an unmapped application property to an unmapped SmartPlant property definition. Note that the two properties to be mapped **DO NOT** have to have the same name.

The results of the mapping will be shown in the bottom control.



11.3.4 Saving Enum/Property Additions

Again, the additions to the mapping information will need to be saved to the xml file. You will be prompted to save the changes to the tool map schema.

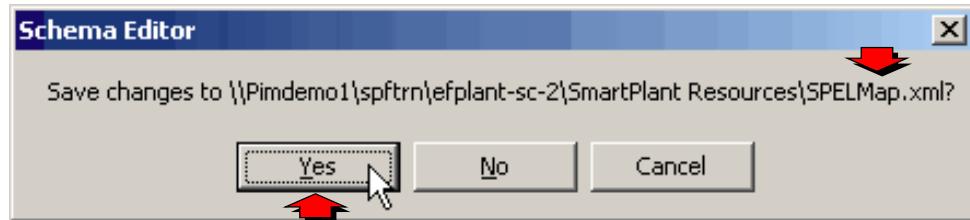


- From the menu bar, click **File > Save All Modified Files** to save the additions to the extension schema and the mapping to the **SPELDataMap** file.



Save Mapping Changes

- Verify the tool map schema name for the mapping changes, and click **Yes**.



© 2005. Intergraph Corp.
All Rights Reserved.

Once the schema file has been saved, exit the Schema Editor.



Save Mapping Changes

- From the menu bar, click **File > Exit** to close out of the schema editor.

© 2005. Intergraph Corp.
All Rights Reserved.

Since no changes were made to the SmartPlant schema, you are not prompted to save the extension file, and there is no need to upload files back to the SmartPlant server.



Save Mapping Changes

- As we made no changes to the SmartPlant Schema, there is no need to upload the files.



11.4 Test Mapped Properties

The next steps will be to test the configuration. Perform a Retrieve on a published P&ID document, verify the data to your Motors in SPEL, and Publish the updated data. Verify the published data is in the XML.



Test Mapped Properties

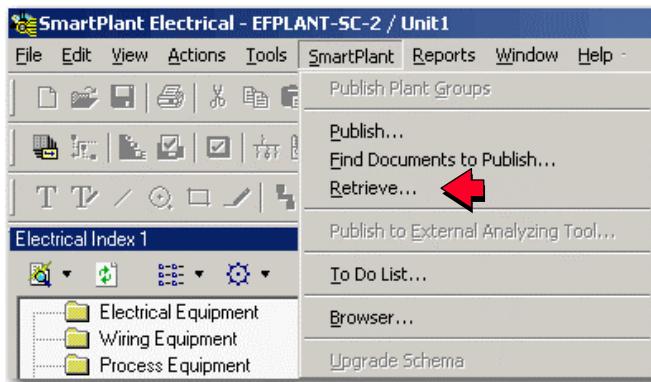
- Open SmartPlant Electrical and perform a Retrieve of a document published by P&ID.**
- Verify the entries for Motor, and change the system code for objects to Sys2.**
- Create a Document Report for publishing.**
- Revise the new Document.**
- Publish the new Document**
- Verify that the data was published.**

Open SmartPlant Electrical, and retrieve a document that was published by SmartPlant P&ID.



Retrieve the Motor Properties

- Open SmartPlant Electrical, and retrieve the latest P&ID-published document.



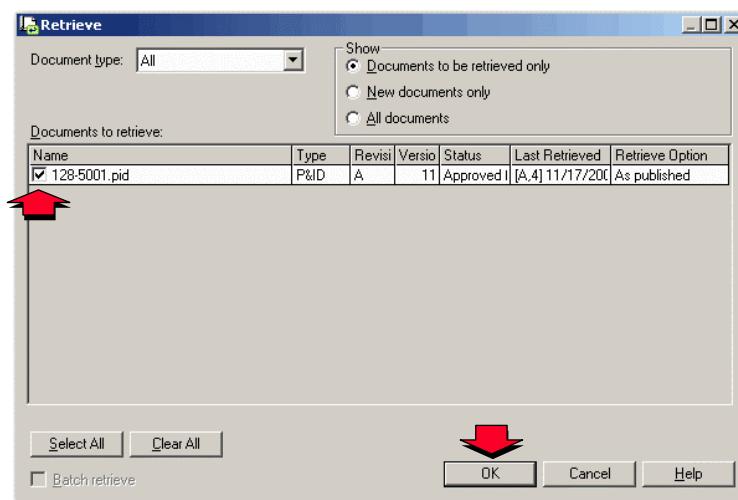
© 2005, Intergraph Corp.
All Rights Reserved.

On the *Retrieve* dialog box, select a document to retrieve from SPPID.



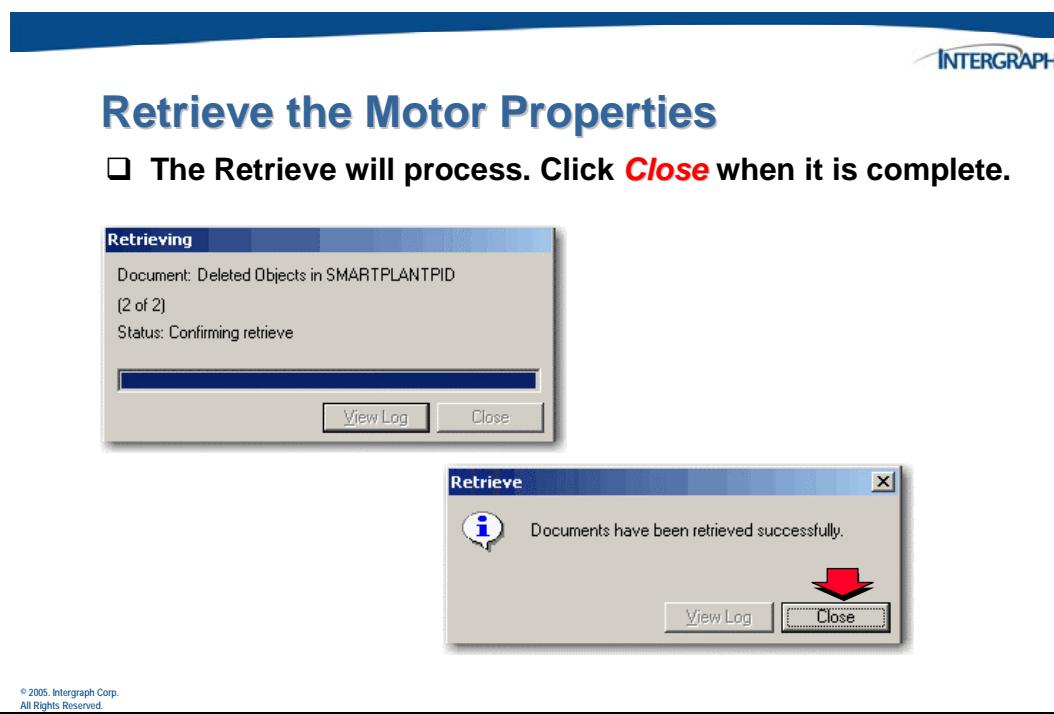
Retrieve the Motor Properties

- Check the latest P&ID document, and click **OK**.

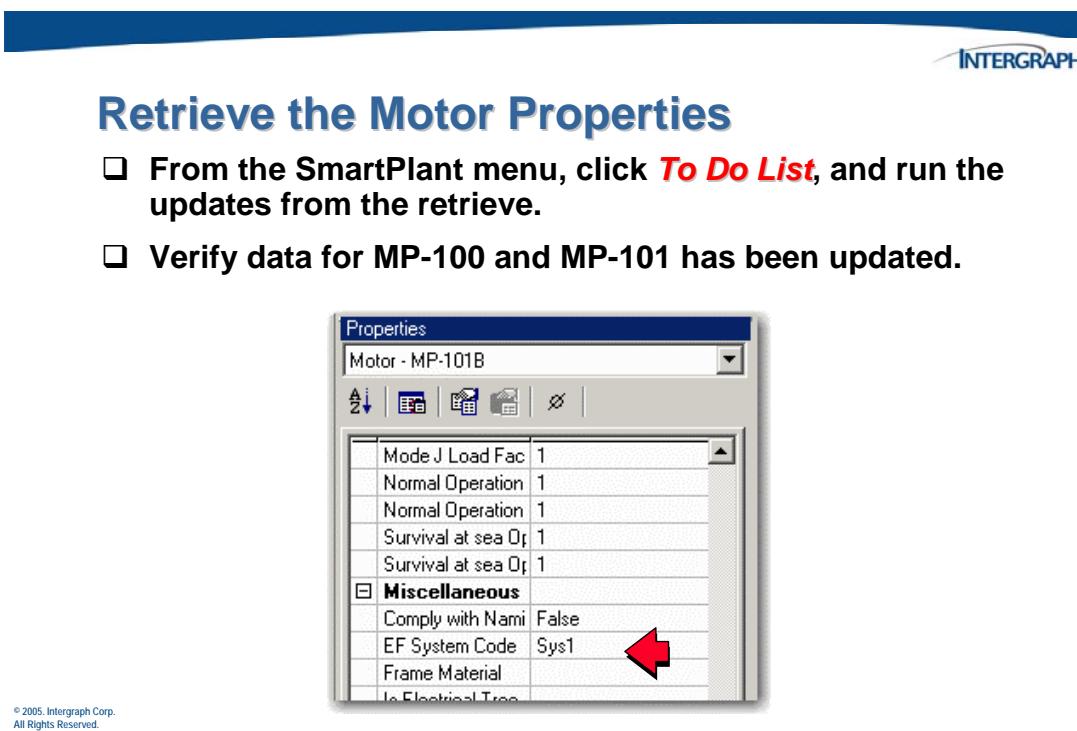


© 2005, Intergraph Corp.
All Rights Reserved.

The Progress dialog box will appear, followed by the Success dialog box when the retrieval process is complete.



Open the *To Do List* and run the updates required as a result of the retrieve.

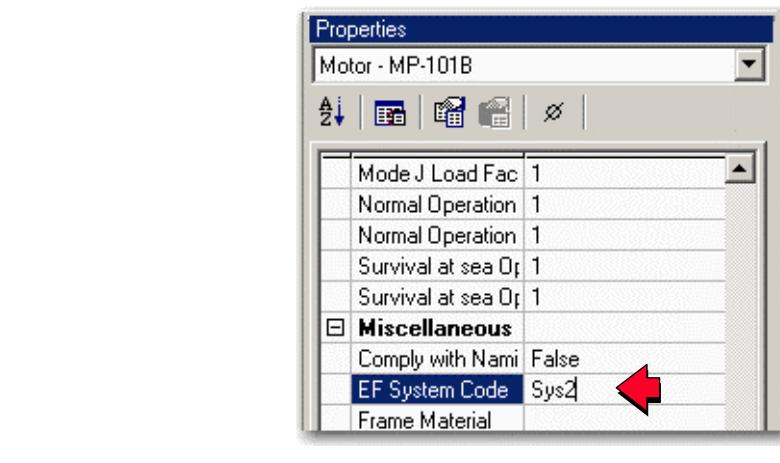


Change the value for *EF System Code*.



Retrieve the Motor Properties

- Change the ***EF System Code*** value to **Sys2**.



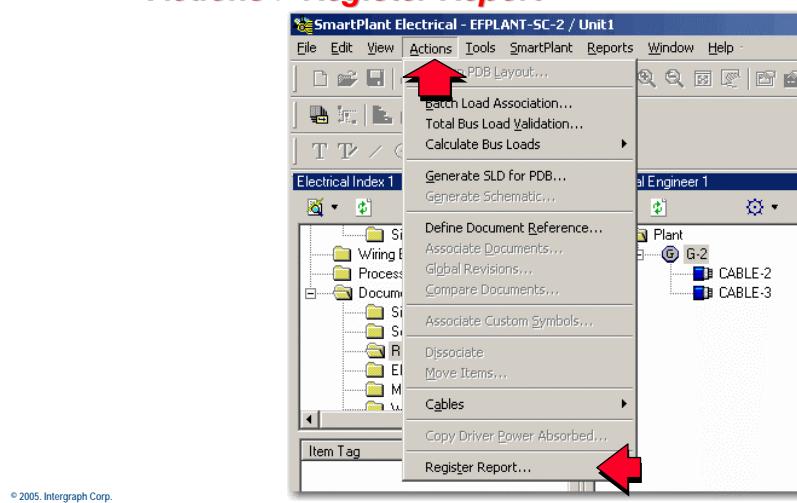
© 2005, Intergraph Corp.
All Rights Reserved.

Create a report to publish from SmartPlant Electrical.



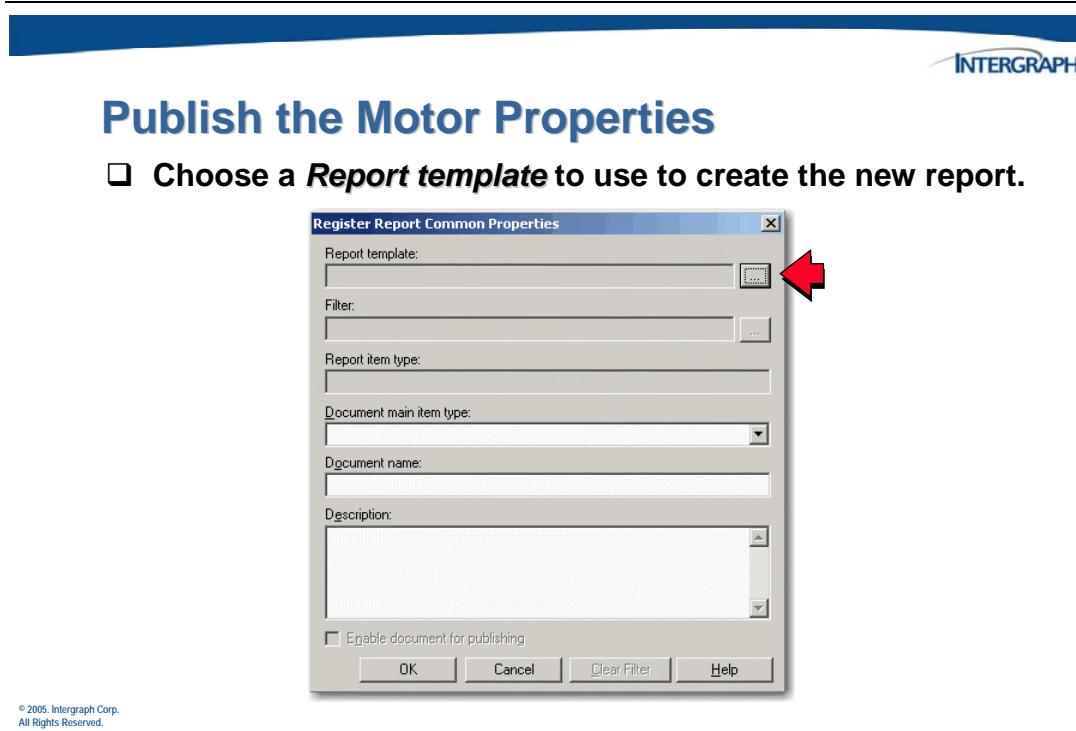
Publish the Motor Properties

- To create a report to publish the new properties, click ***Actions > Register Report***.

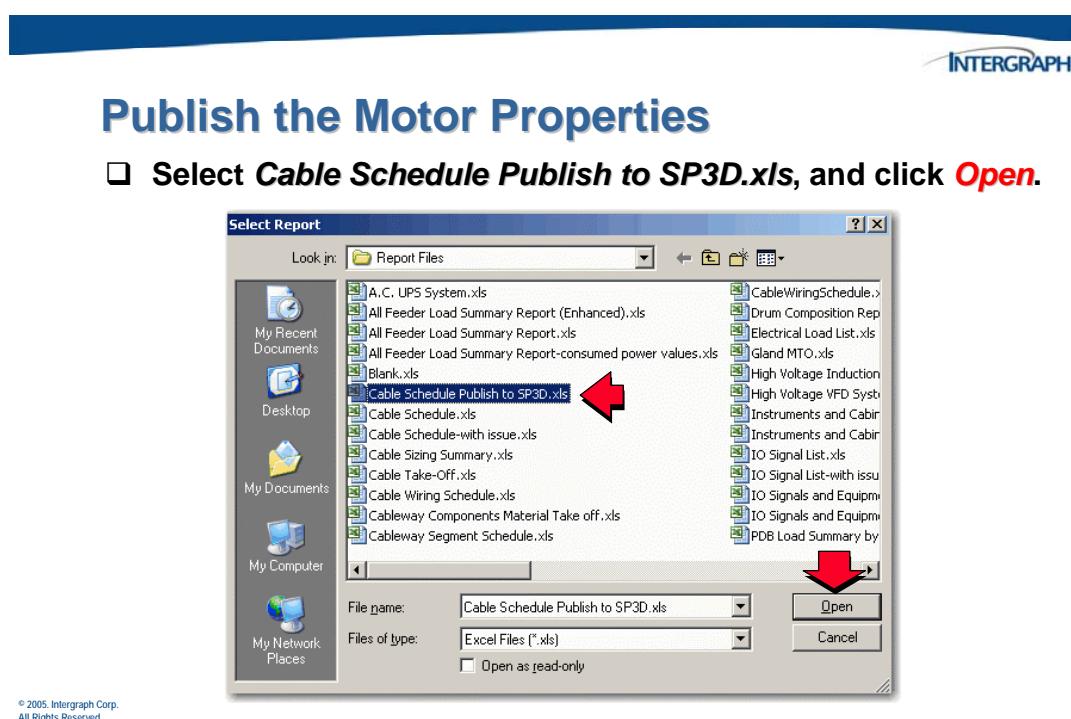


© 2005, Intergraph Corp.
All Rights Reserved.

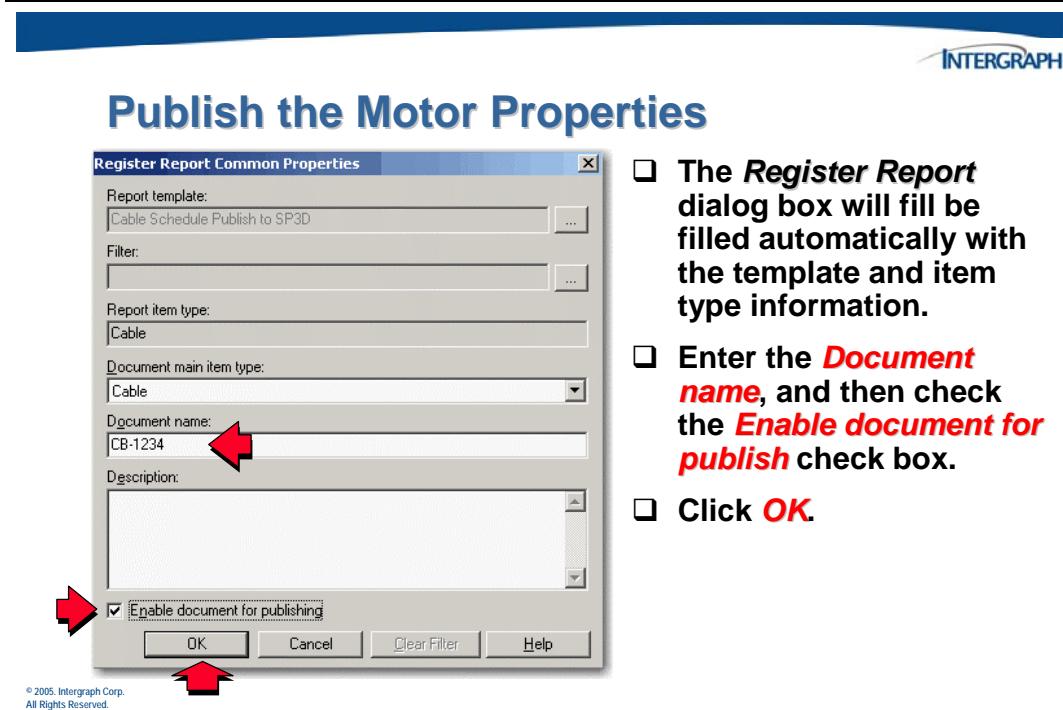
Click the Browse button by the report template field to select a template to use.



Choose the *Cable Schedule Publish to SP3D* template from the list.

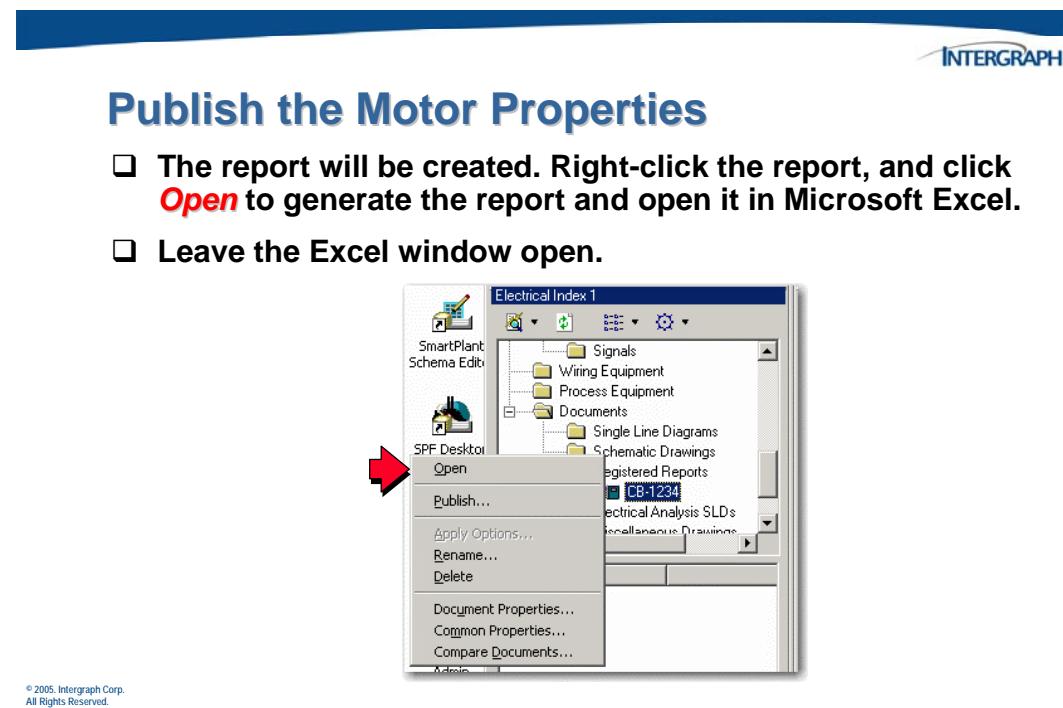


Provide a name for the report, and make sure the *Enable document for publishing* check box is checked.



- ❑ The **Register Report** dialog box will be filled automatically with the template and item type information.
- ❑ Enter the **Document name**, and then check the **Enable document for publish** check box.
- ❑ Click **OK**.

Open the report in Microsoft Excel.

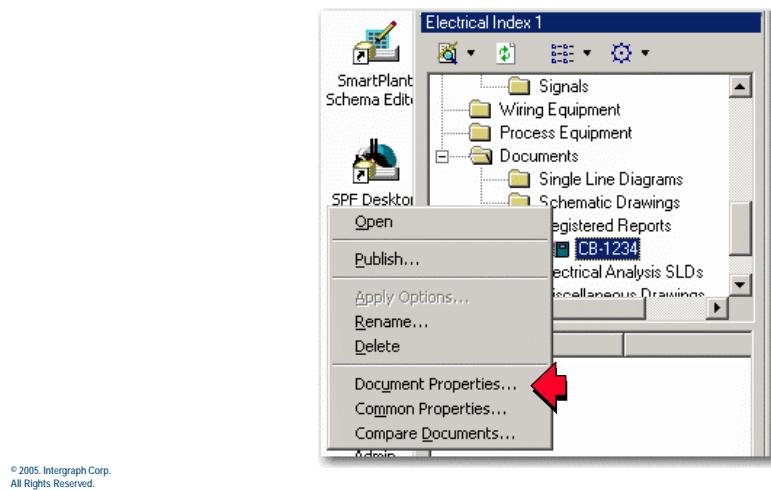


Open the *Document Properties* dialog box.



Publish the Motor Properties

- Right-click on the report, and click **Document Properties**.

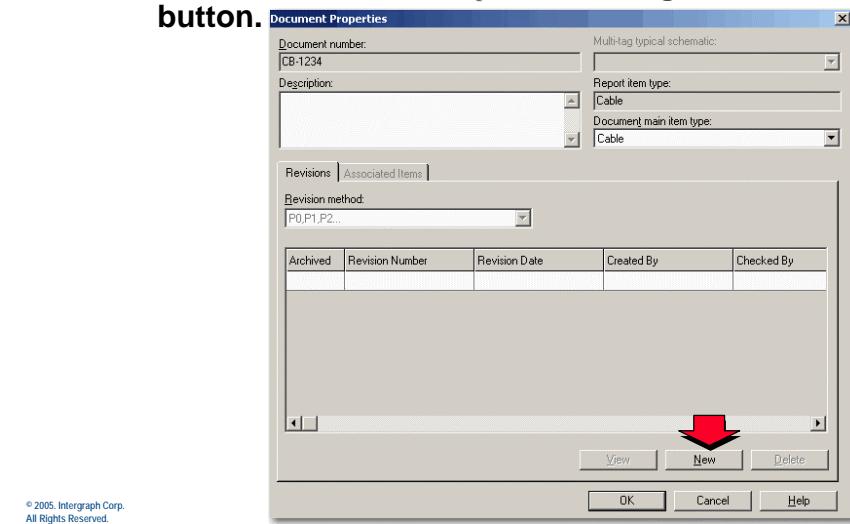


Click **New** on the *Document Properties* dialog box to open the *Revise* dialog box.

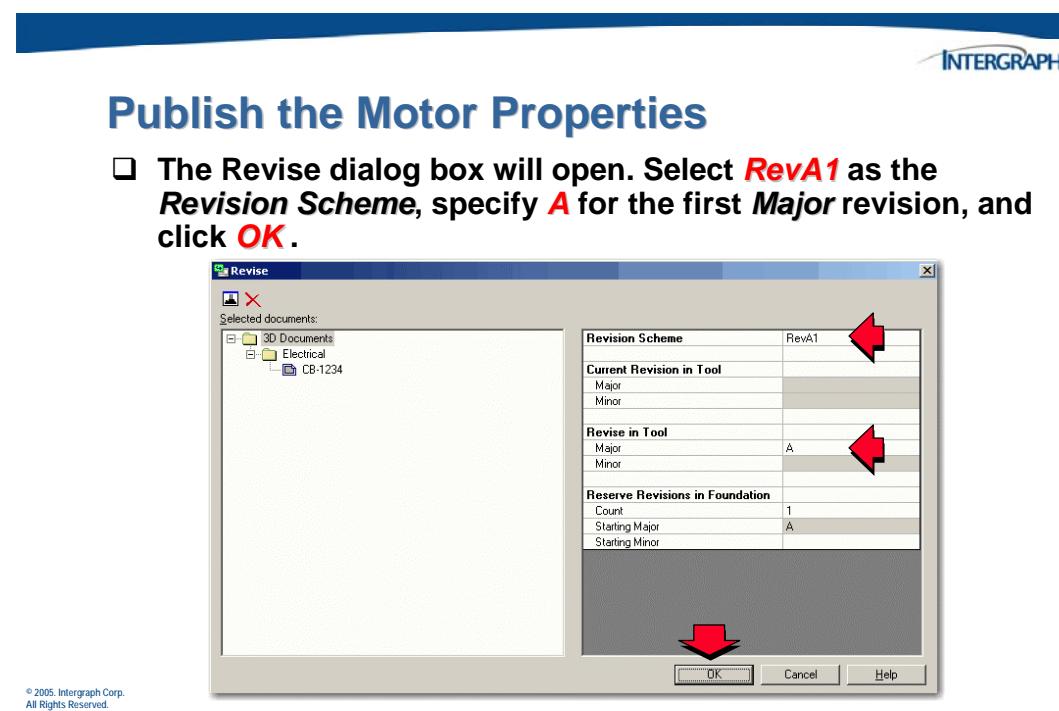


Publish the Motor Properties

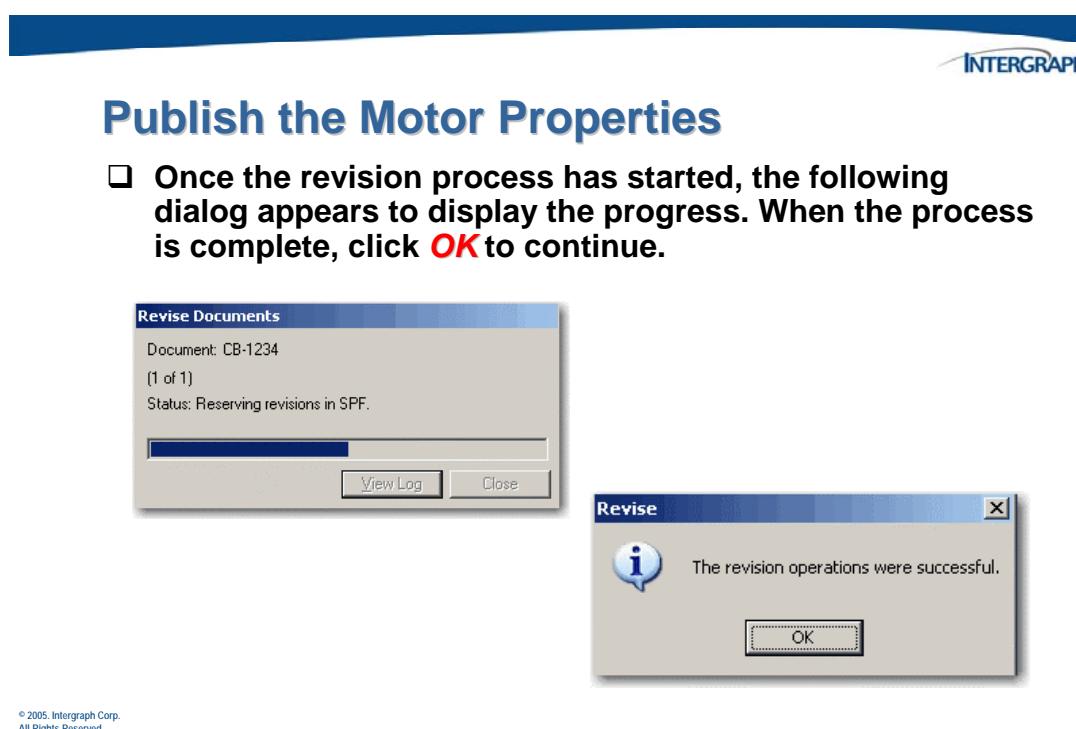
- On the *Document Properties* dialog box, click the **New** button.



On the *Revise* dialog box, select a *Revision Scheme* and specify the first major revision.



The Progress dialog box will indicate the progress of the revision creation.

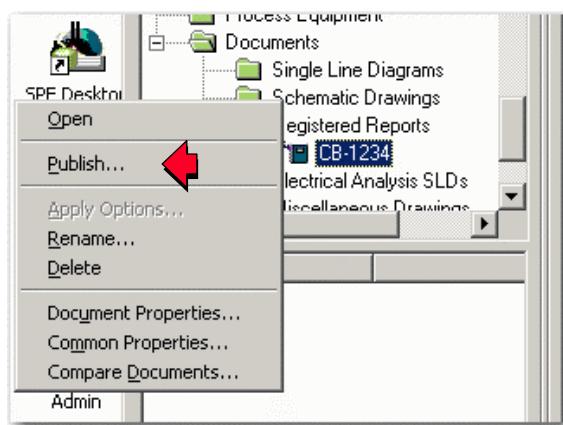


Once you have created a new revision, publish the report.



Publish the Motor Properties

- Right-click on the report, and click **Publish**.



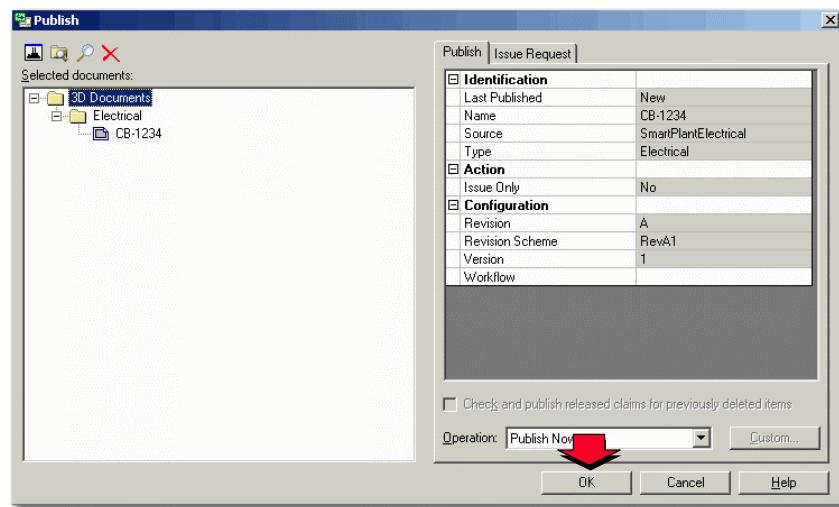
© 2005. Intergraph Corp.
All Rights Reserved.

Click **OK** on the *Publish* dialog box.



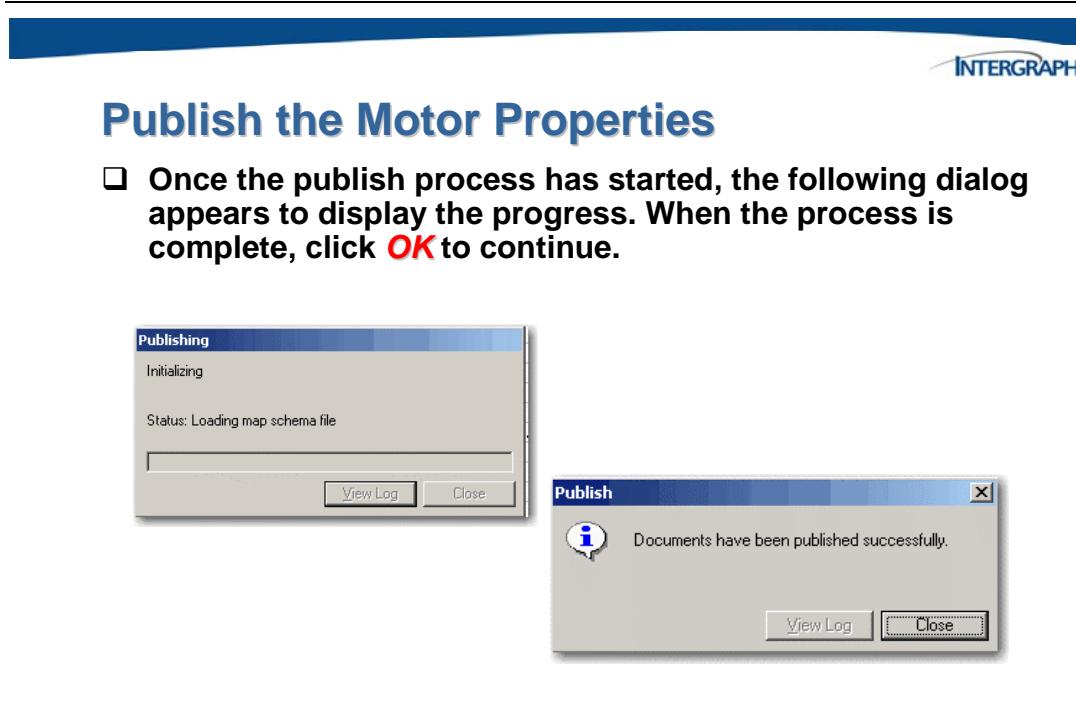
Publish the Motor Properties

- On the *Publish* dialog box, click **OK**.

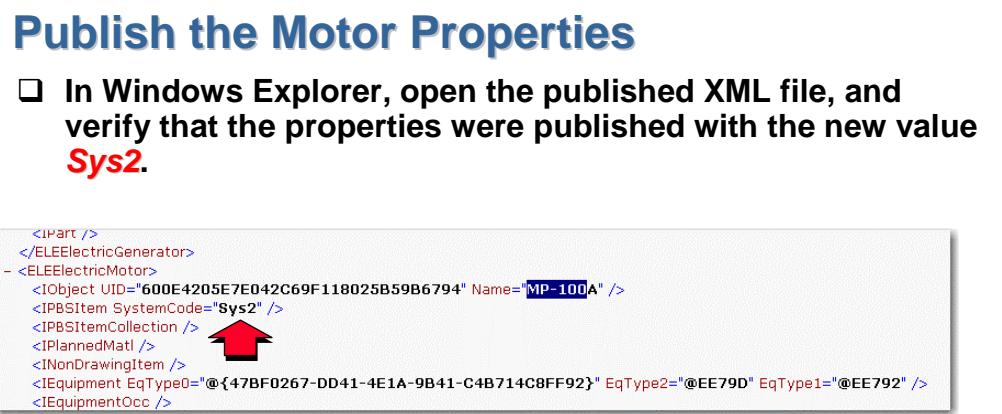


© 2005. Intergraph Corp.
All Rights Reserved.

The Progress dialog box will indicate the progress of the publish process.



Confirm that the new information was published in the XML file.



11.5 Activity 2 – Adding and Mapping a Complex Property

In this activity you will be creating custom properties in the SmartPlant SPEL authoring tool. You will also create a select list that will be used with a new property. Finally, you will perform the necessary mapping steps that would be used in a Publish and Retrieve operation.

1. Log on to your operating system as *spfuser* with no password (if not already logged in).
2. Click **Start > All Programs > Intergraph SmartPlant engineering Manager > Data Dictionary Manager** to start the *SmartPlant Data Dictionary Manager*. Verify you are connected to the SPEL database.
 - ρ **File > Open Database...**
 - ρ Select *Application type:* **SmartPlant Electrical**
 - ρ Select the **Site Server...** button and look in \\spftrb\\efplant-sc-2. Choose **smartplantv4.ini**
 - ρ Verify the plant **EFPLANT-SC-2** and click **Open**
3. Add a new select list (picklist).
 - ❑ Click **Select List** on the left side of the application window.
 - ❑ Scroll to the bottom of the Select List table and enter a new name in the empty row at the bottom of this table.
 - Name – **EngSys**
 - Dependent List – **None**
 - ❑ Click **Select Entry** on the left side of the application window.
 - ❑ In the *Select Entry* window, enter the following information in the blank row shown:
 - Value – **AA**
 - Short Value – **Steam Gen & Fired Htr, Water/Steam Side**
 - Dependent Value – <blank>
 - ❑ Select the **Add Row** command from the tool bar to add the next entry to the new Select List:

- Value – **BA**
 - Short Value – **Steam Generator and Fired Heater - Air/Gas Side**
 - Dependent Value – <blank>
- Select the **Add Row** command from the tool bar to add the next entry to the new Select List:
- Value – **CA**
 - Short Value – **Ammonia/Urea**
 - Dependent Value – <blank>
- Select the **Add Row** command from the tool bar to add the next entry to the new Select List:
- Value – **DC**
 - Short Value – **Crude Oil Production**
 - Dependent Value – <blank>
- Select the **Add Row** command from the tool bar to add the last entry to the new Select List:
- Value – **EA**
 - Short Value – **Auxiliary AC Power Systems**
 - Dependent Value – <blank>
4. Add a new property to the Instrument database table that utilizes the new Select List just created.
- Click **Plant Item** in the Database Tables window.
- Select **Edit > Add Property** from the menu.
- In the *Add Property* dialog box, enter the following information:
- Name – **EngSystem**
 - Display Name – **Engineering System**
 - Data Type – **Select List**
 - Select List – **EngSys**
 - Format – **Variable length**
 - Default Value – **None**
 - Display to User – **Yes**

- Use for Filtering – **Yes**
 - Category – **Miscellaneous**
 - Depends on – **None**
- Click **OK** to close the *Add Property* dialog box.
5. Save the changes to the SmartPlant Data Dictionary Manager.
 Click **File > Save**.
 Click **File > Exit**.
- ## Adding a Schema Property and Enumerated List
6. Click **Start > Programs > Intergraph SmartPlant Foundation > Schema Component > SmartPlant Schema Editor** to start the *Schema Editor*.
7. Connect to SmartPlant by selecting the **Application Metadata** tab at the top right portion of the *Workflows* dialog. Select the *SmartPlant* button and click **Connect to SmartPlant**.
8. Enter the correct URL to connect to *SmartPlant Foundation* <http://pimdemo1/spf38asp> and click **Next**.
ρ When prompted for a SmartPlant Foundation login, use **adminuser** with no password.
ρ Choose the **EFPLANT-SC-2** SmartPlant Foundation plant to work with
ρ Click **Next**
ρ Select the tool map schema to be loaded, *SmartPlant SPEL Tool Schema*
ρ Enable the **Load map schema** toggle
ρ Enable the **Connect to application schema** toggle
ρ Click **Finish**
9. When the *Synchronize* dialog displays, confirm that the new property values have been imported by the meta data adapter and click **OK**.

10. If needed, add new entries to a new Enumerated List in the SmartPlant schema to correspond to new entries that have been added to a tool map schema during the synchronization.
- Expand the **Tool** object in the tree of the *Map Environment* window so that the SmartPlant SPEL Tool Schema entries are displayed.
 - Expand the **Map Enumerated Lists** objects and verify that the entry for **EngSys** was added to the SmartPlant SPEL Tool Map Schema as a result of the synchronization with the SPEL application meta schema.
 - Right-click on the EngSys entry in the tree and choose Edit **SPEN_12001** from the dynamic menu.
 - What does *SPEN_12001* represent?

 Where does the number originate from?

 Select the **Advanced** tab in the *Edit Map Enumerated List Definition* dialog. Verify that the new list is displayed and has all of the correct entries in it.

 At the bottom of the dialog, locate and select the **New SmartPlant Enumerated List with Correlated Entries** button.

 What does this button do?

 - Click the **OK** button to close the *Edit Map Enumerated List Definition* dialog.
 - Right-click on the EngSys entry in the tree and choose Edit **SPEN_12001** from the dynamic menu once more to change the child entry numbers in the EngSys list.
 - Select the **Publish** tab in the *Edit Map Enumerated List Definition* dialog. Then right-click on the EngSys enum list in the SmartPlant control and choose **Edit** from the dynamic menu.
 - In the *Edit Enumerated List* dialog box, change the **Number** information:
- | Text | Description | Number |
|------|---|--------------|
| AA | Steam Gen & Fired Htr, Water/Steam Side | 10111 |
| BA | Steam Generator and Fired Heater - Air/Gas Side | 10112 |
| CA | Ammonia/Urea | 10113 |
| DC | Crude Oil Production | 10114 |
| EA | Auxiliary AC Power Systems | 10115 |

- Click the **OK** button to close the *Edit Enumerated List* dialog.
11. If needed, add a new property to the SmartPlant schema to correspond to the new property that has been added to a tool meta schema and tool map schema.
- Expand the **Map Classes** and right-click on the *Motor* entry in the tree and choose **Edit SPEL_MOTOR** from the dynamic menu.
 - Select the **Publish** tab in the *Edit Map Map Class Definition* dialog. Expand the tree for both the application/tool map schema and the SmartPlant schema in the upper control.
 - Highlight **Identification** in the tree and select the **New Property Definition** button **below** the SmartPlant tree in the upper control.
 - In the *New Property Definition* dialog box, enter the following information:
 - Name – **EngineeringSys**
 - Description – **Engineering System property**
 - Display Name – **Engineering System**
 - Click  next to the *Exposed by interface definitions* box and select the **IPBSItem InterfaceDef** (if that was on your list from step 10 in hands on 1) from the *Possible ExposedByInterfaceDefs for EngineeringSys* dialog box.
 - Click  next to the *Scoped by property type* box and select the **EngSys** enumerated list from the *Possible ScopedBy.PropertyType Values for EngineeringSys* dialog box.
 - Click **OK** to close the *New Property Definition* dialog box.

Defining Schema Mapping

12. Highlight the *EngSystem MapProperty* in the **Unmapped applications properties** control and the *EngineeringSys MapProperty* in the **Unmapped SmartPlant properties** control in order to perform the schema mapping.
- Make sure that **EngSystem** is highlighted in the middle control in the application section and that **EngineeringSys** is highlighted in the SmartPlant section.
 - Click the **Map** button to define the mapping between the SPEL tool map schema and the SmartPlant schema.
 - Verify in the bottom control that *EngSystem* maps to *EngineeringSys* in the two schemas. You may have to scroll to the bottom of the list to see this.
 - Click **OK** to close the *Edit Map Enumerated List Definition* dialog box.

13. Save the changes to the all schema files.
 - From the menu, click **File > Save All Modified Files...**
 - Answer **Yes** to all of the individual xml file save prompts.
 - Click **File > Exit** to close and exit the Schema Editor.
 - When prompted to Upload files to SmartPlant Foundation, select **No** (unless you had to add a new SmartPlant property definition).
14. Use Windows Explorer to verify that all the files were uploaded to **C:\Program Files\Common\Intergraph\EFSchema\03.08** with the latest time and date.
15. When you are finished with this activity, you may take a short break until the other students have finished.

C H A P T E R

12

Mapping with SmartPlant 3D

12. Defining Mapping for SmartPlant 3D

The section that follows provides the assumptions, rules, and limitations imposed by the SmartPlant 3D adapter on the tool mapping.

12.1 SmartPlant 3D Schema Mapping: An Overview

When the schema in SmartPlant is extended, such as when new classes, properties, or select list entries are added, you or an administrator must react to the change. You must update the mapping data in SmartPlant 3D. Mapping data is necessary for the publish and retrieve operations, as well as for correlating design basis objects with model objects. You also must update the Catalog Schema database in SmartPlant 3D, if the new information does not already exist in the database.



Chapter Objectives

The objective of this **SP3D Mapping** chapter is to demonstrate how to do the following:

- Describe the SP3D mapping mechanism**
- Configure SPF Schema with custom properties**
- Setup SP3D for the new properties to be Retrieved**
- Setup SP3D for Publishing the new properties**
- Test the Retrieve and Publish process**

12.2 General Information about SmartPlant 3D Schema Mapping

Mapping ensures a meaningful transfer of data between SmartPlant 3D and the SmartPlant environment for use in other tools that share that environment through integration.

12.2.1 Adapters

Each authoring tool has an adapter that processes information during the publish and retrieve operations. One of the adapter's functions is to map information between SmartPlant and the particular authoring tool. SmartPlant 3D currently has two adapters: one for retrieving and another for publishing. The retrieve adapter for SmartPlant 3D requires that each document type have a registry entry present. These registry entries define the map file used and define callback objects that perform additional processing. It is possible that each document type could have its own map file. The retrieve adapter uses a map file, which is defined by tool schema files. The publish adapter uses a tool schema file (generated from the SmartPlant 3D Catalog Schema and supplemented with mapping information) and a schema called the P3DComponent Schema, which is derived from the SmartPlant Schema.



SP3D Mapping

- SmartPlant 3D currently has two adapters.**
- One is for retrieving, and the other is for publishing.**
- The retrieve adapter requires that each document type have a registry entry present.**
- The registry entries define the map file to be used.**
- The registry also defines callback objects that perform additional processing.**
- The publish adapter uses a tool schema file.**
- The schema file is generated from the SmartPlant 3D Catalog Schema and supplemented with mapping information.**
- The adapters interact with the P3DComponent created from the EFSchema configuration in SPF.**

12.2.2 Map Files

For retrieve, the delivered map file is DesignBasis_map.xml. For publish, the map file is SP3DPublishMap.xml. The map file for correlating design basis objects with SmartPlant 3D is SP3DToEFWClassMap.xml.

Notes:

- The software does not currently provide a tool to manage change in the SP3DToEFWClassMap.xml file. You should save a copy of this file before you make changes to it. All the map files are located in the symbol share directory that is associated with the catalog for the site.
- To ensure piping properties are passed correctly from SPP&ID the value **SmartPlant 3D** must be set for the **Use Piping Specification in Options Manager**.

12.3 How Mapping is Configured for Retrieve

Mapping for retrieve is configured from a set of component schemas contained in the EFSchema.xml file on the SPF server. The information in these component schemas is processed into a SmartPlant 3D schema package and a map file. The schema package is loaded into the SmartPlant 3D Catalog Schema (metadata database), and the map file is made available to each SmartPlant 3D client. Modification to the map file is automatic and does not require manual editing. For retrieve, the Generate Design Basis tool provided by Core does the mapping. The **Generate Design Basis** tool updates the map file that is, in turn, used to generate the metadata package for the design basis objects. During a retrieve operation, the map file is used to translate the data received from SmartPlant into the form defined in the Catalog. For example, attributes in a SmartPlant document are typically passed as strings. The corresponding attribute in SmartPlant 3D could be a string, a long, a double, a codelist, or other type. The map file could translate a string, such as TRUE, to the Boolean value True in SmartPlant 3D.



SP3D Mapping (Map Files)

- The Retrieve map file is *DesignBasis_map.xml*.
- For publish, the map file is *SP3DPublishMap.xml*.
- The map file for correlating design basis objects with SmartPlant 3D is *SP3DToEFWClassMap.xml*.
- The software currently does not provide a tool to manage change in the *SP3DToEFWClassMap.xml* file.
- In SPP&ID Options Manager, set SmartPlant 3D for the **Use Piping Specification** entry.

12.4 How Mapping is Configured for Publish

Mapping for publish is done by adding information to the tool schema. The base tool schema is defined by the SmartPlant 3D Catalog Schema. Then, this base tool schema is modified using utilities provided by the Core, including a merge utility and an extract utility. The Schema Editor from SmartPlant is also used in the modification. The net result of the modifications is that the tool schema includes applicable mapping information. During a publish operation, the tool schema (with added mapping information) and the SP3DComponent (derived from the SmartPlant Schema) are used to convert SmartPlant 3D objects to SmartPlant objects.



SP3D Mapping

- Component Schemas from the other applications are loaded into the catalog database using the Generate Design Basis.**
- The Design Basis will update the map file.**
- The map file is used to translate the data received from the SmartPlant environment into the form defined in the Catalog.**

12.5 Other Information about Mapping

You might wonder how the software knows which object corresponds to a property. During a retrieve operation, each object is identified by its UID. A one-to-one relationship exists between an object and its UID. Once the object is identified and brought into memory, the mapped properties are stored. You might also wonder if you can map attributes on the document object itself. This mapping is possible as long as the attributes are defined in both the SmartPlant Schema and the SmartPlant 3D Catalog Schema. Typically, a document object implements the IDocument interface in addition to other special purpose interfaces.



SP3D Mapping

- The base tool schema is defined by the SmartPlant 3D Catalog Schema.**
- When the Catalog is modified, the changes are passed to the Tool Schema by the SmartPlant Schema Editor.**
- The Tool Schema is synchronized with the Catalog during the mapping workflow in the SmartPlant Schema Editor.**
- After they are synchronized, the properties are then mapped to the SP3Dcomponent schema entries.**

12.6 Publish Limitations

Mapping for publish carries the following limitations:

- The mapping is done against the P3D Component only.
- The Schema Editor does not currently show the properties for mapped edge definitions. Any mapping spreadsheet generated using the Schema Editor will not have all the mappings.
- Some properties are set in the publish code and cannot be changed for this release.



SP3D Mapping

- During a retrieve operation, each object is identified by its UID.
- The UID is used because of the one-to-one relationship between an object and its UID.
- You can map attributes on the document object itself by insuring the attributes are defined in both the SmartPlant Schema and the SmartPlant 3D Catalog Schema.
- Document objects will implement the IDocument object in SmartPlant Schema.

12.7 Retrieve Limitations

During a retrieve operation, the adapter assumes that the data it receives is synchronized with the metadata defined in the Catalog Schema. The map file cannot have a one-sided definition. That is, a class or property must exist both in the SmartPlant Schema and in the SmartPlant 3D Catalog Schema. The map file is responsible for making this connection between the two definitions. An important limitation of the retrieve process is that shared objects (objects that contain information from two or more tools) are completely deleted when a single tool deletes an object.



SP3D Mapping

- Mapping for publish carries the following limitations:**
 - The mapping is done against the P3D Component only.
 - The Schema Editor does not currently show the properties for mapped edge definitions. Any mapping spreadsheet generated using the Schema Editor will not have all the mappings.
 - Some properties are set in the publish code and cannot be changed in this release.

12.8 Design Basis Mapping

For the mapping from the design basis objects to the SmartPlant 3D objects, the SmartPlant Schema is not used. SmartPlant 3D uses a custom mapping definition. The filename for this definition is SP3DToEFWClassMap.xml.



SP3D Mapping

- During a retrieve operation, the adapter assumes that the data it receives is synchronized.
- The map file cannot have a one-sided definition.
- The class or property must exist both in the SmartPlant Schema and in the Catalog Schema.
- An important limitation of the retrieve process is that shared objects (objects that contain information from two or more tools) are completely deleted when a single tool deletes an object.

12.9 Limitations on the SmartPlant Schema

The retrieve map is updated directly from the contents defined in the SmartPlant Schema; therefore, a few restrictions on the SmartPlant Schema exist.

- Codelist entries (enumerated lists) must not have duplicated indices within a single list. A duplicated value in another level within a hierarchical codelist is permissible. When duplicate indices are present, the bulkload process will fail.
- In rare cases, if an entry is removed from the SmartPlant Schema, you may need to edit the retrieve map file to remove the deleted entry. You should use the Schema Editor delivered with SmartPlant Schema Component for this purpose.



SP3D Mapping

- Mapping from the design basis objects to the SmartPlant 3D objects, the SmartPlant Schema is not used.**
- SmartPlant 3D uses a custom mapping definition *SP3DToEFWClassMap.xml*.**

12.10 Mapping Configuration for Retrieve in SP3D

This is an example of extending two Enumeration lists and a Property Definition (Fluid System, EngineeringSystem, System Code) between PID, SPF, and SP3D.

- Open the EFSchema.cfg configuration in the C:\Program Files\Common Files\Intergraph\EFSchema\03.08 directory.
- Next to **Another Schema File**, click **View**, then open in one of the Editors. In this example, chose **Editor**.
- Expand out **EnumListType**, and locate Fluid system.



SP3D Mapping

- There are few restrictions on the SmartPlant Schema because the retrieve map is updated directly from the contents defined in the SmartPlant Schema.**
- Codelist entries (enumerated lists) cannot have duplicated indices within a single list.**
- Duplicated values in another level within a hierarchical codelist is permissible.**
- In rare cases, if an entry is removed from the SmartPlant Schema, you may need to edit the retrieve map file to remove the deleted entry.**



SP3D Mapping

- The following is an example of extending two Enumeration lists and a Property Definition (Fluid System, EngSys, System Code).
- Because these objects have been created in earlier chapters, we will verify the entries before continuing.
- Open the Schema Editor, and then open the *EFSchema.cfg*.
- Click the **View** button next to **Another Schema File**, and in the View dialog, click **OK**.
- Expand **EnumListType**, and locate **Fluid system**.

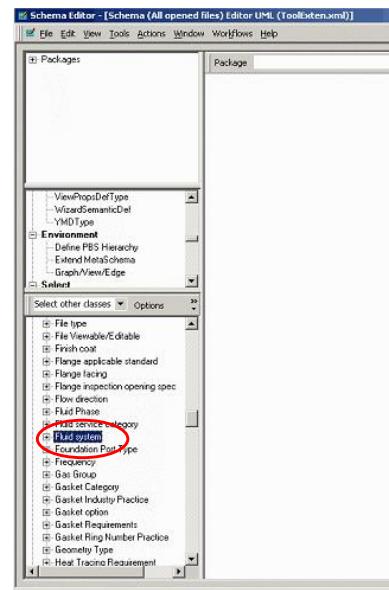
© 2005. Intergraph Corp.
All Rights Reserved.

Right-click, and click **View Fluid Systems**.



SP3D Mapping

- Click the **View** button next to **Another Schema File**, and in the View dialog, click **OK**.
- Expand **EnumListType**, and locate **Fluid system**.



© 2005. Intergraph Corp.
All Rights Reserved.

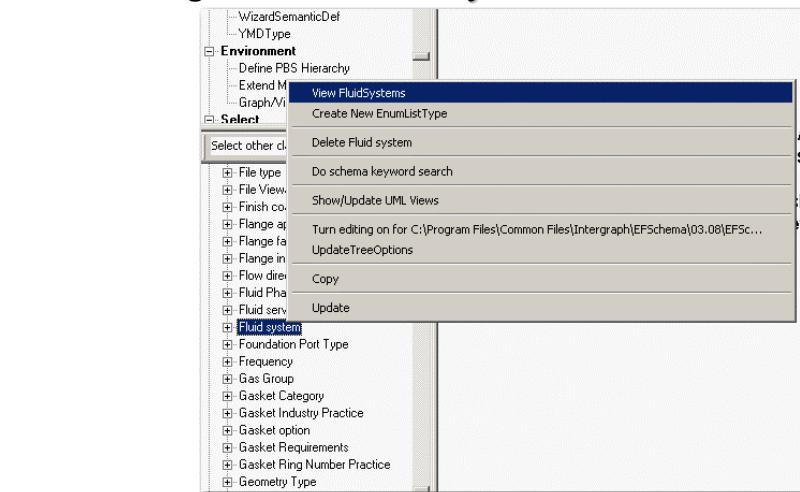
In this example, we are going to verify the Fluid System and the Fluid Codes. The Fluid System is Corrosive. Click the (+) to expand Fluid system and to display four Fluid Codes. These entries were added in the earlier P&ID Mapping chapter.

Note:

- Make note of these values (Short and Long Description, and Number) for future use with the tool schemas. This is especially important for SP3D, due to the fact the number given must match the value bulkloaded into the catalog. This must be done for the Fluid System and Fluid Codes.

SP3D Mapping

- Right-click on *Fluid system*, and click **View Fluid Systems**.

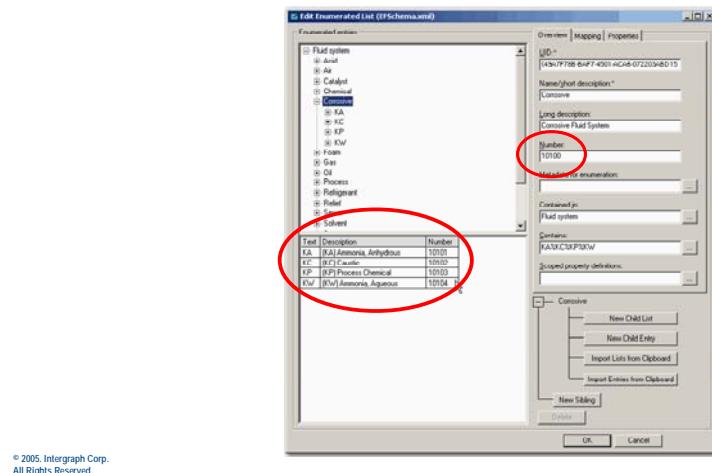


© 2005, Intergraph Corp.
All Rights Reserved.



SP3D Mapping

- Select **Corrosive**, and expand that option to see four (4) fluid codes.



© 2005, Intergraph Corp.
All Rights Reserved.

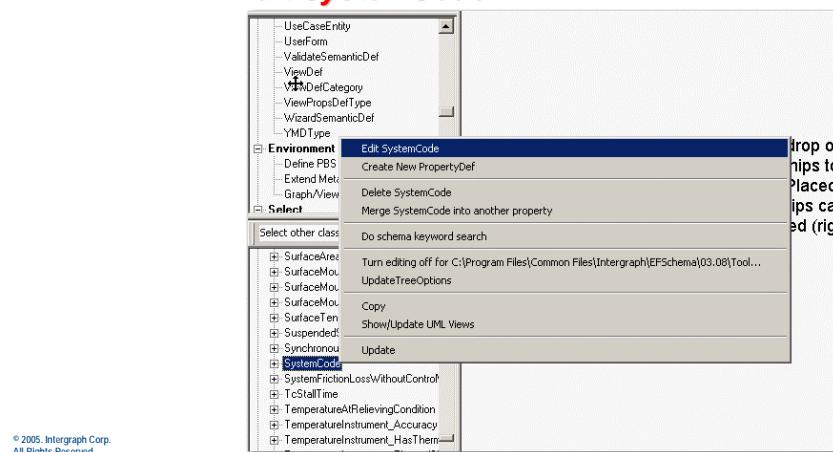
Click **Cancel** to exit this dialog.

Next, review the **SystemCode** Property. Under the tree view, expand the **Property Def** list, and find the **SystemCode** property. Right-click, and click **Edit SystemCode**.



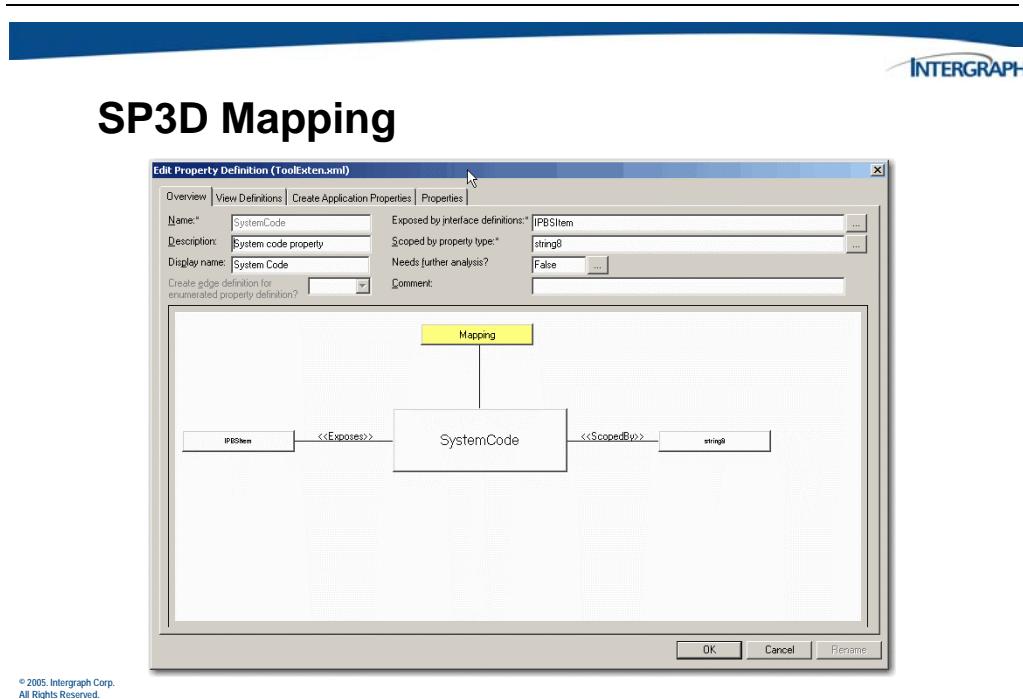
SP3D Mapping

- In the tree view, expand the **Property Def** list, and find the **SystemCode** property. Right click on **SystemCode**, and click **Edit SystemCode**.



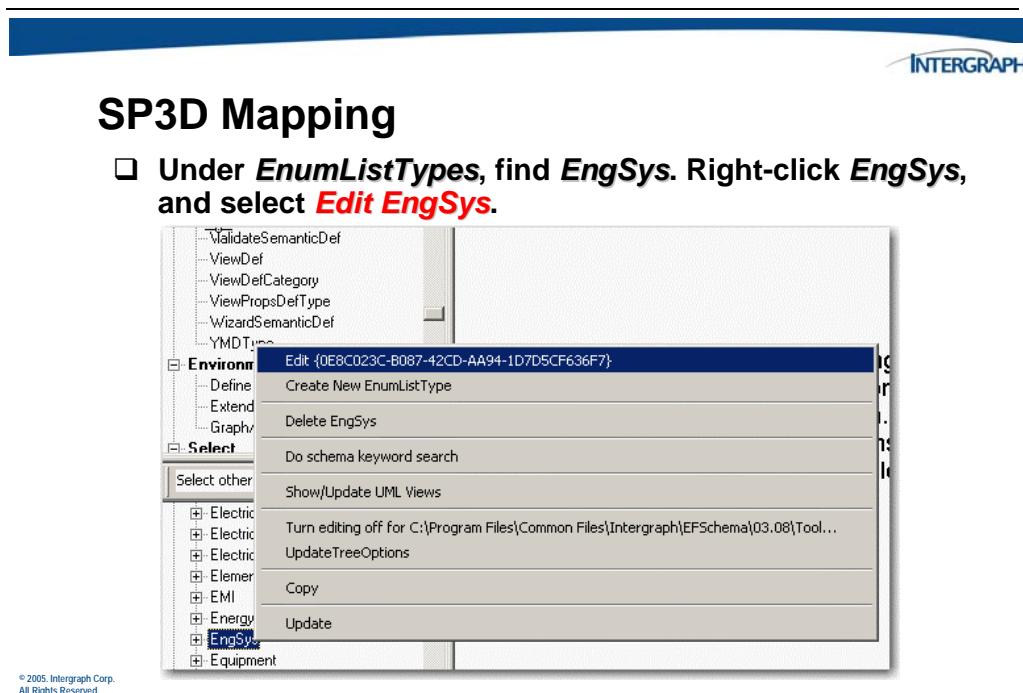
© 2005, Intergraph Corp.
All Rights Reserved.

Verify the properties for the **SystemCode**, and click **Cancel**.

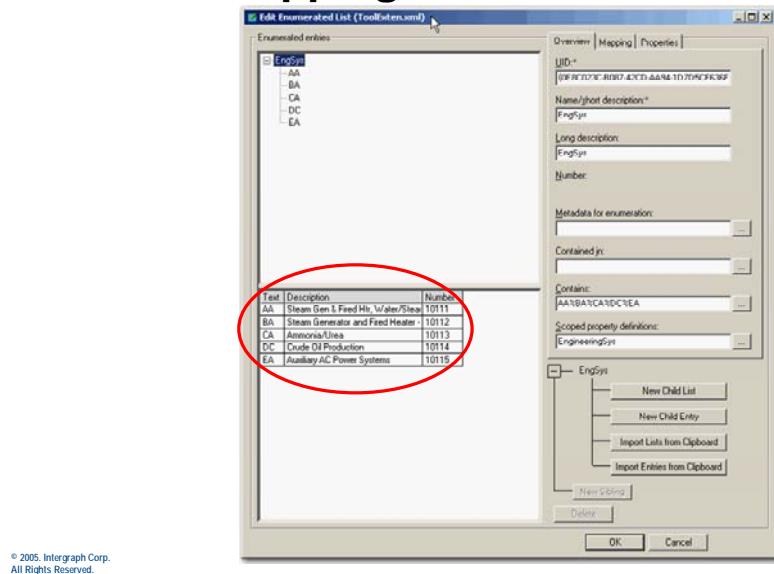


Verify the **EnumListType EngSys** and its entries.

Under **EnumListTypes**, find the **EngSys**, right-click it, and select **Edit EngSys**.



SP3D Mapping



© 2005. Intergraph Corp.
All Rights Reserved.

Verify the properties for the **EngSys**, and click **Cancel**.

Note:

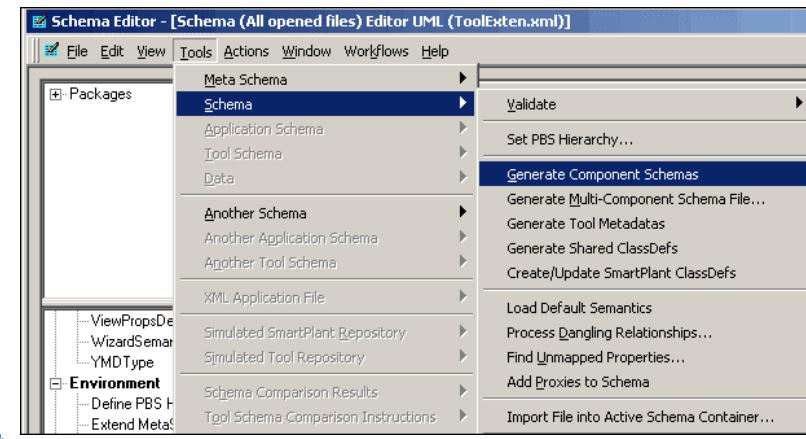
- Remember EFSchema changes needs to be loaded into the SPF server. Go to the SPF server, then locate EFSchemaLoader.exe in your Foundation install directory. Load the schema changes into the SPF database. Also, you need to run **Generate Component Schemas** in Schema editor for the SP3D Design Basis.

With the schema loaded, from the menu bar select **Tools > Schema > Generate Component Schemas** to run the utility to create updated component schemas for each tool. This will be used when we run regenerate Design Basis in the SP3D Project utility. This utility will take several minutes to complete.



SP3D Mapping

- To make sure the component schemas are updated with the schema changes, run the **Generate Component Schemas** utility within the Schema Editor.



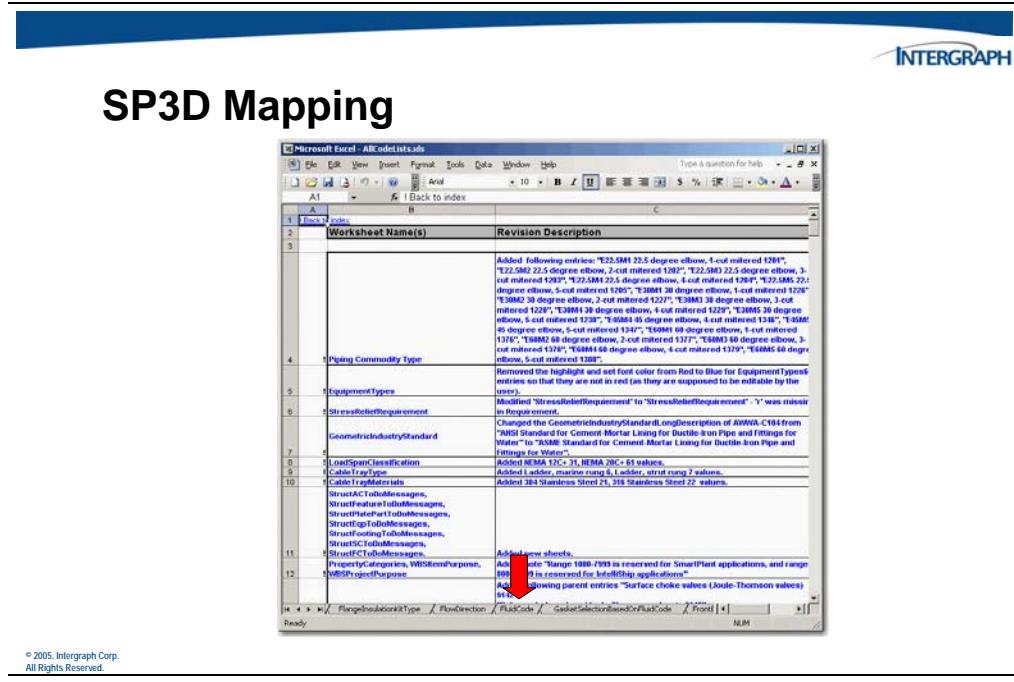
Next, make a copy of the original **AllCodesList.xls**, which is in the directory **E:\SmartPlant\SP3D\CatalogData\BulkLoad\DataFiles** on this server. Edit the copy of **AllCodesList.xls** to add the new Fluid Code object to its worksheet.



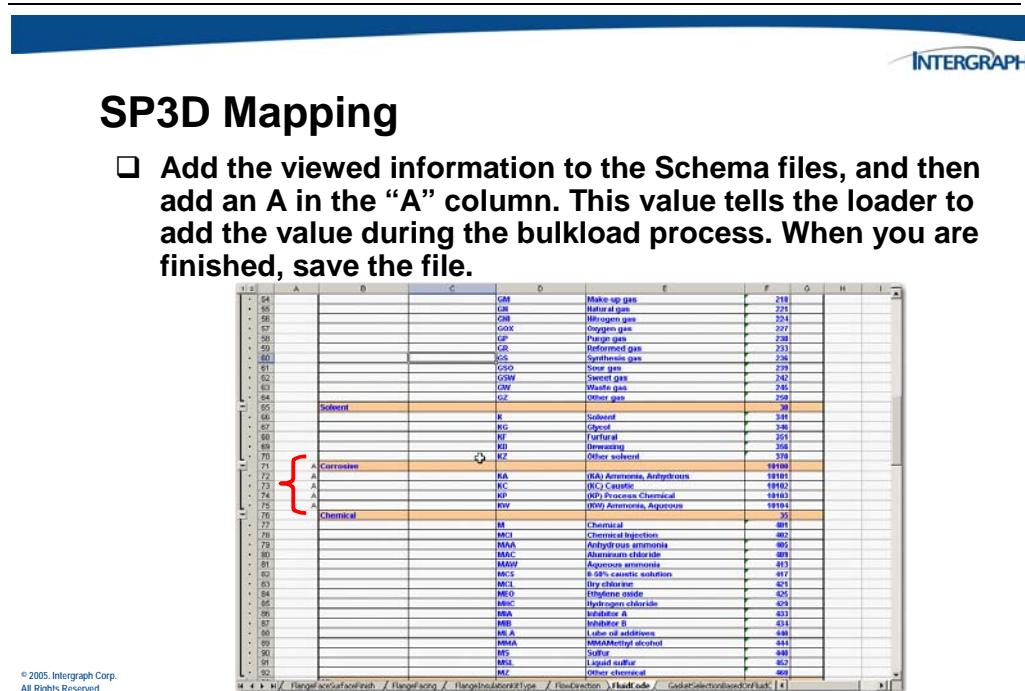
SP3D Mapping

- Make a copy of the original **AllCodesList.xls**, which is in the following directory on this server:
E:\SmartPlant\SP3D\CatalogData\BulkLoad\DataFiles
- Open the spreadsheet, locate the **Fluid Code** tab, locate the **Gas Fluid System**, and then insert a row under this **Fluid System**.

Open the spreadsheet, locate the **Fluid Code** tab, then locate the **Gas Fluid System**. Insert a row under this Fluid System, add the information that was added to the Schema files, then add an “A” to add this value during bulkload. Save the file.



This adds the updated entries for **Corrosive** to the load file for loading.



Next, add two Properties to a Custom Interface XLS file to load them into SP3D.

The file D:\SPF_Training\SP3D\EFCustomProps.xls file will contain the SystemCode and EngineeringSystem properties. Open this file to verify the properties.



SP3D Mapping

- Edit the D:\SPF_Training\SP3D\EFCustomProps.xls file, which contains the SystemCode and EngineeringSystem properties.

A	B	C	D	E	F	G	H
1	Back to Index						
2							
3	Head	InterfaceName	CategoryName	AttributeName	AttributeUserName	Type	UnitsType
4							
5	Start						
6	IUPBSItem	Standard	SystemCode	System Code	Char	0	C
7	IUPBSItem	Standard	EngineeringSystem	Engineering System	Long	0	C
8							
9	End						
10							
11							



SP3D Mapping

	G	H	I	J	K	L	M
1							
2							
3	UnitsType	PrimaryUnits	CodeList	codelisttablenamespace	OnPropertyPage	ReadOnly	SymbolParameter
4							
5							
6	0	0			TRUE	FALSE	
7	0	0	EngineeringSystem	UDP	TRUE	FALSE	
8							
9							
10							
11							
12							

© 2005, Intergraph Corp.
All Rights Reserved.



SP3D Mapping

A	B	C	D	E	F	G	H	I
1								
2	HEAD	ClassName	InterfaceName					
3								
4	Start							
5								
6	!	Example of adding interfaces to virtual classes						
7								
8								
9	!	Adding interfaces to non-virtual classes						
10	a	CPPipelineSystem	IUPBSItem					
11	a	CPMPipeRun	IUPBSItem					
12								
13								
14	End							
15								

© 2005, Intergraph Corp.
All Rights Reserved.

Also in the same directory is the file **EFCustomCodeList.xls**.



SP3D Mapping

- The file **EFCustomCodeList.xls** is in the same directory.

A	B	C	D	E	F
1	Back to Index				
2					
3	!				
4	!				
5	EngineeringSystem HEAD ShortDescription	EngineeringSystem LongDescription	Codelist Number	Sort Order	
6	START				
7	a AA	Steam Gen & Fired Htr, Water/Steam Side		10111	
8	a BA	Steam Generator and Fired Heater, Air/Gas Side		10112	
9	a CA	Ammonia/Urea		10113	
10	a DC	Crude Oil Production		10114	
11	a EA	Auxiliary AC Power Systems		10115	
12	END				
13					
14					

© 2005, Intergraph Corp.
All Rights Reserved.

Once verified close and save the changes.

Next, update the SP3DToEFWClassMap.xml file. This will tell SP3D which object to pass into 3D for the other applications.



SP3D Mapping

- Next, update the **SP3DToEFWClassMap.xml** file.
- Add the XML lines displayed here:

```
<MappedClassName>PidPipingConnector</MappedClassName>
- <InterfaceMap>
  - <InterfaceName>IUPBSItem</InterfaceName>
    - <PropertyMap>
      <PropertyName>SystemCode</PropertyName>
      <MappedInterfaceName>IPBSItem</MappedInterfaceName>
      <MappedPropertyName>SystemCode</MappedPropertyName>
      <Updatable>True</Updatable>
    </PropertyMap>
    - <PropertyMap>
      <PropertyName>EngineeringSystem</PropertyName>
      <MappedInterfaceName>IPBSItem</MappedInterfaceName>
      <MappedPropertyName>EngineeringSys</MappedPropertyName>
      <Updatable>True</Updatable>
    </PropertyMap>
  </InterfaceMap>
- <InterfaceMap>
```

12.11 Loading objects into SP3D

In this section, we will see the steps for loading the custom and modified properties into SP3D. This process has 4 steps:

- Bulkload the data into the Catalog Database
- Create the Design Basis
- Regenerate the views in the Model Database
- Regenerate the Reports Database

Note:

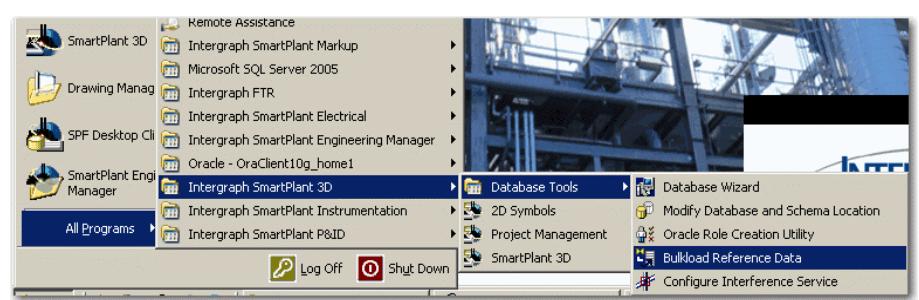
- These steps have been performed for you on the Class VM. This was done because of the time it will take to complete these tasks.



SP3D Mapping

- The SP3D loading process includes 4 steps:**
 - Bulkload the data into the Catalog Database.
 - Generate the Design Basis.
 - Regenerate the views in the Model Database.
 - Regenerate the Reports Database.
- Due to time constraints and the amount of time required to complete these tasks, these steps have been performed for you on the Class VM. However, the following slides illustrate the process.**

The first step is to load the data using the Bulkload utility from SP3D. To do this select **All Programs > Intergraph SmartPlant 3D > Database Tools > Bulkload Reference Data**.

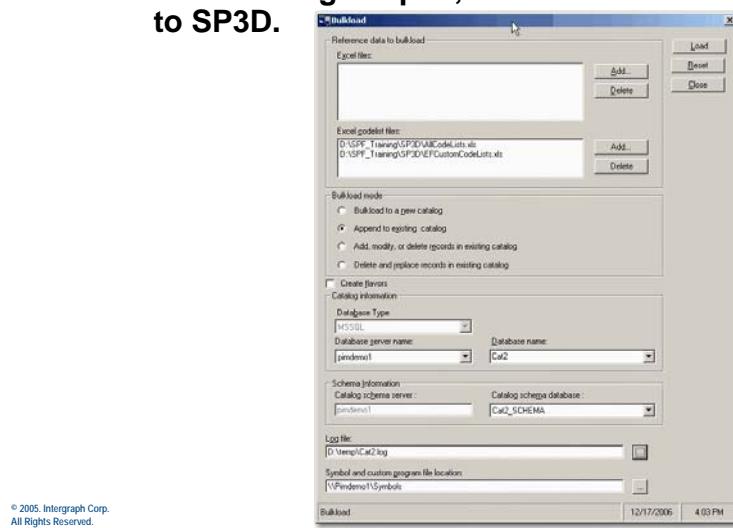


Once the dialog is opened, you will need to select the codelists you wish to input into SP3D. Click the **Add** button next to the codelist section. Browse to the directory where you codelist xls files are located. In this class the xls files are located in the **D:\SPF_Training\SP3D** directory. The file names are **AllCodeLists.xls** and **EFCustomCodeLists.xls**. Under the **Bulkload Mode**, select **Append to existing catalog**. Under the **Catalog Information**, select **pimdemo1** for **Database Server Name**, **Cat2** for the **Database Name**, and **Cat2_Schema** for the **Catalog Schema Database**. You can add a log file path and file name in the next section, we used **D:\temp\cat2.log**. The last section will define where the symbols directory is located. This will be different per your install of SP3D, ours is **\pimdemo1\symbols**.

Select the load button to start the bulkload process. This will take some time to complete.

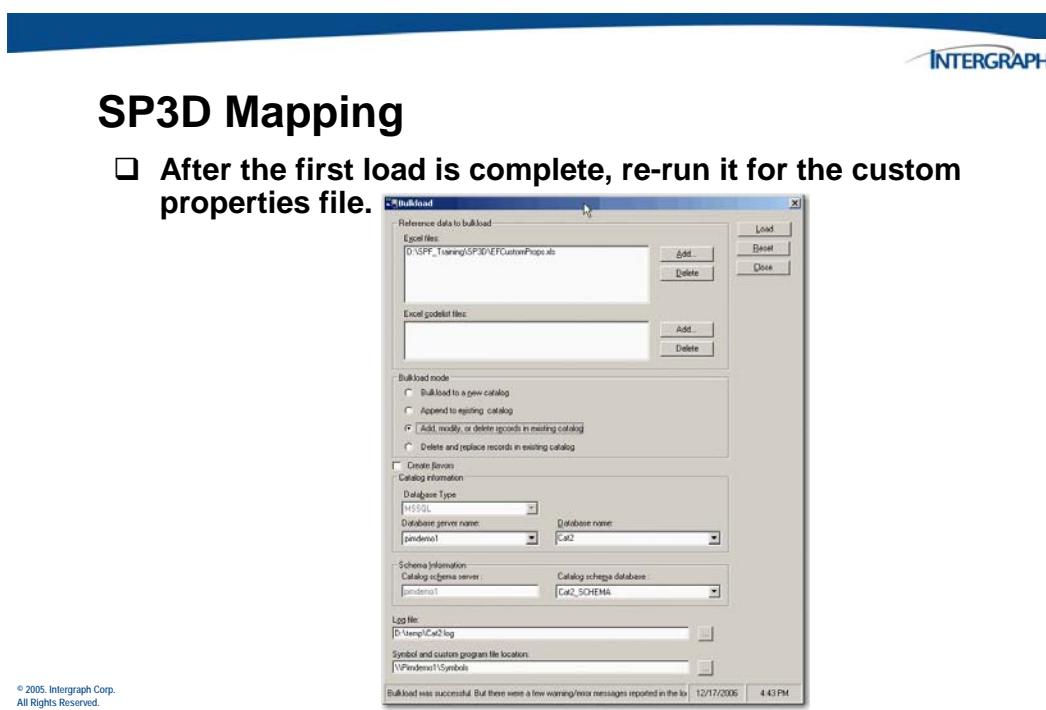
SP3D Mapping

- When dialog is open, select the codelists you want to add to SP3D.



© 2005, Intergraph Corp.
All Rights Reserved.

After the first load is complete, we need to re-run it for the custom properties file. So remove the two codelist files from the list and add the property file to the top section in the dialog. Change the **Append to catalog** to **Add, Modify, delete records in existing catalog**. Again, the file is in the **D:\SPF_Training\SP3D** directory, and its name is **EFCustomProps.xls**. The other setting will be the same as the first load.

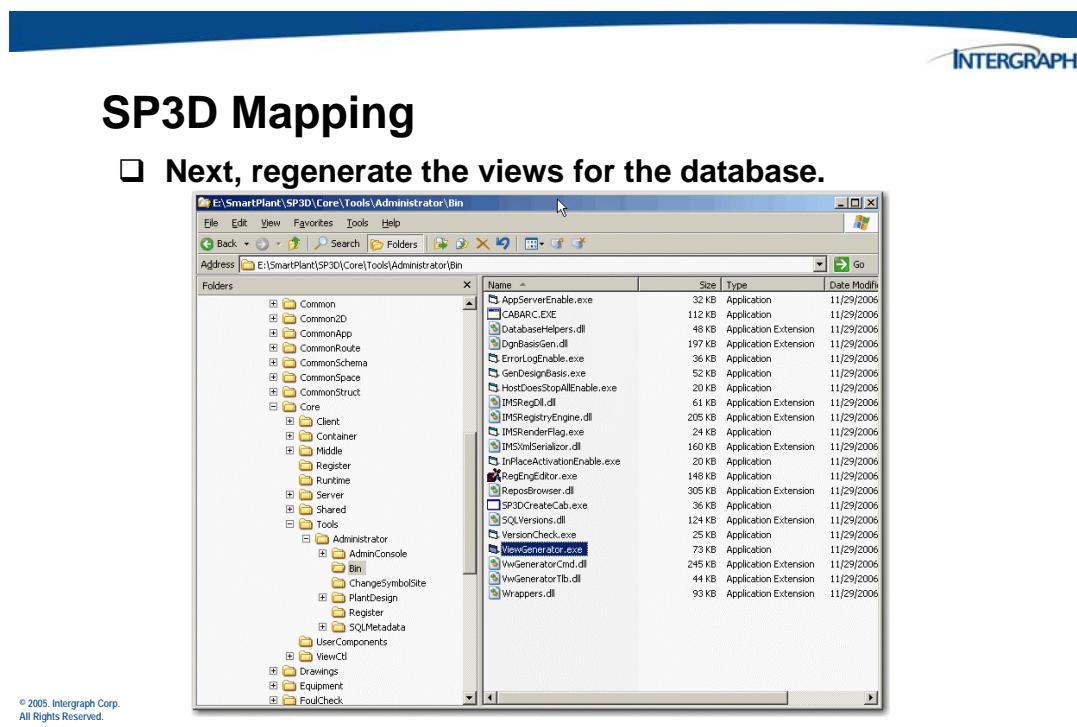


Close the Bulkload utility.

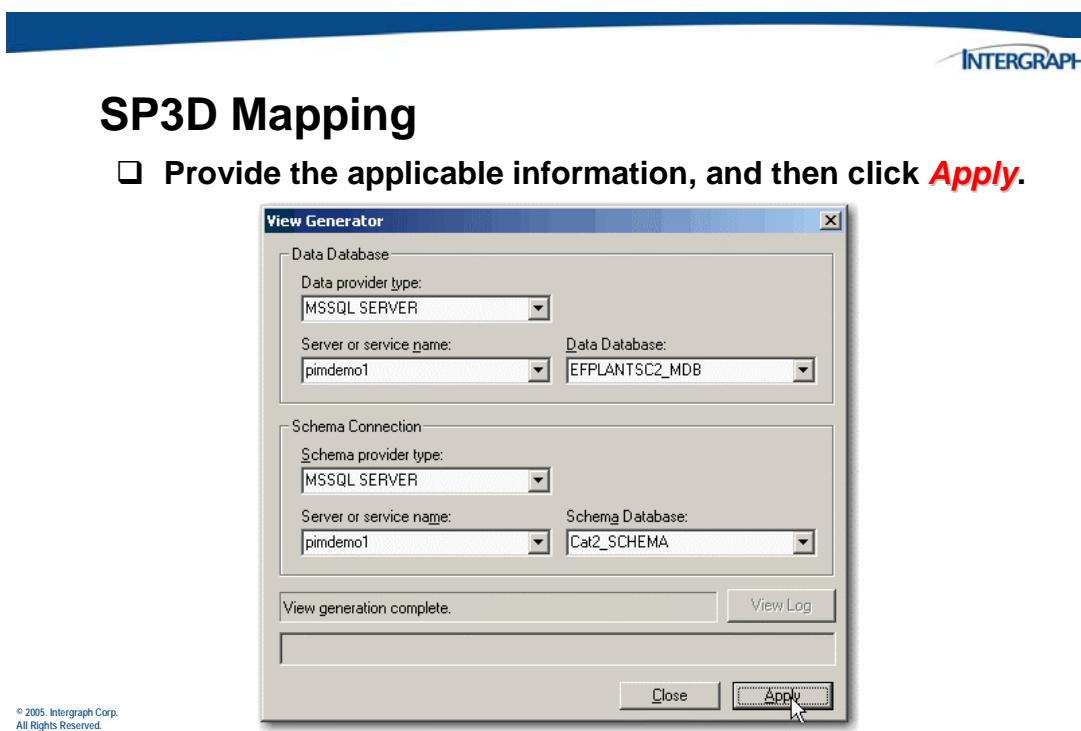
The next step is to regenerate the Views for the Database. To do this you will need to open Windows Explorer and go to the followingn directory:

E:\SmartPlant\SP3D\Core\Tools\Administrator\Bin (the file location will be different on you server depending on you load directory)

Run the file **ViewGenerator.exe**.



In this dialog, enter the Server or Service name of **pimdemo1**, Data Database of **EFPLANTSC2_MDB**, and Schema Database of **Cat2_Schema**. Click **Apply** to run the utility.

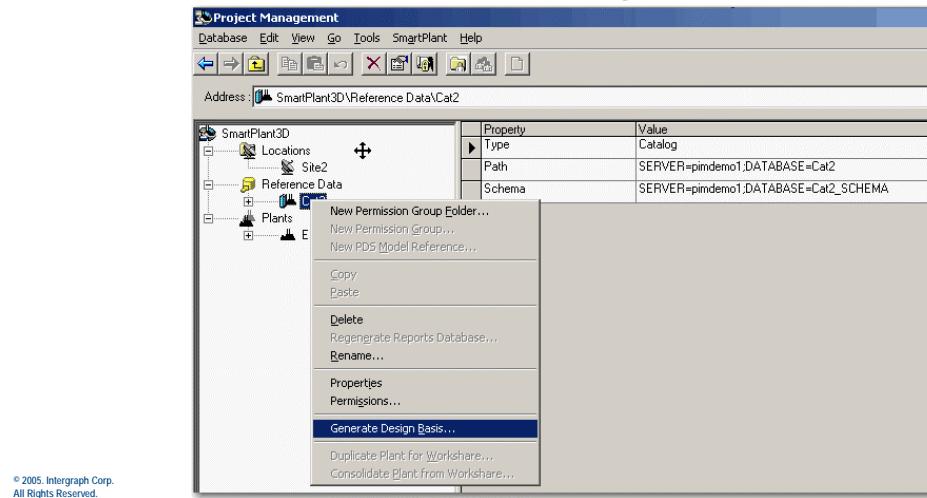


Once both bulkloads and the View Generator have been run, recreate the **Design Basis**. To do this open the Project utility for SP3D. It is located under **All Programs > Intergraph SmartPlant 3D > Project Management**. Under the 3DPlant structure, open the Reference Data section, and right-click on **Cat2**. Click the **Generate Design Basis** command. It will start the process and display a completed dialog when completed.



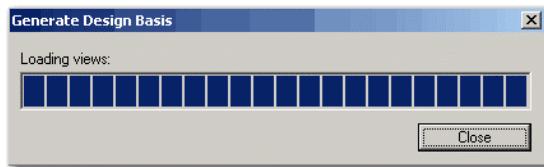
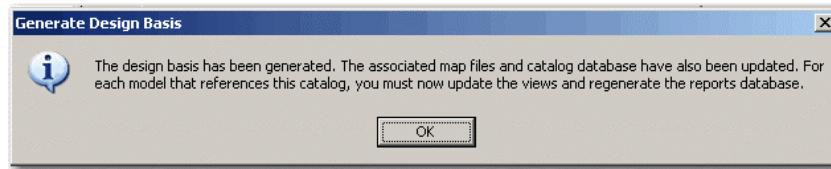
SP3D Mapping

- Once both bulkloads have been run, recreate the Design Basis. To do this, open the *Project* utility for SP3D.

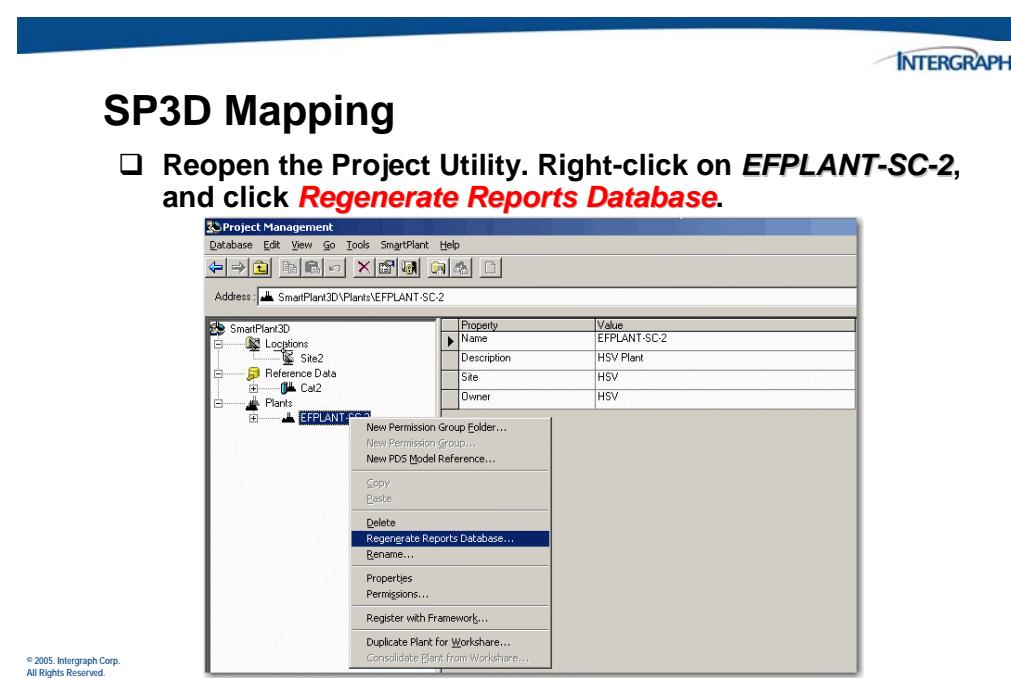


SP3D Mapping

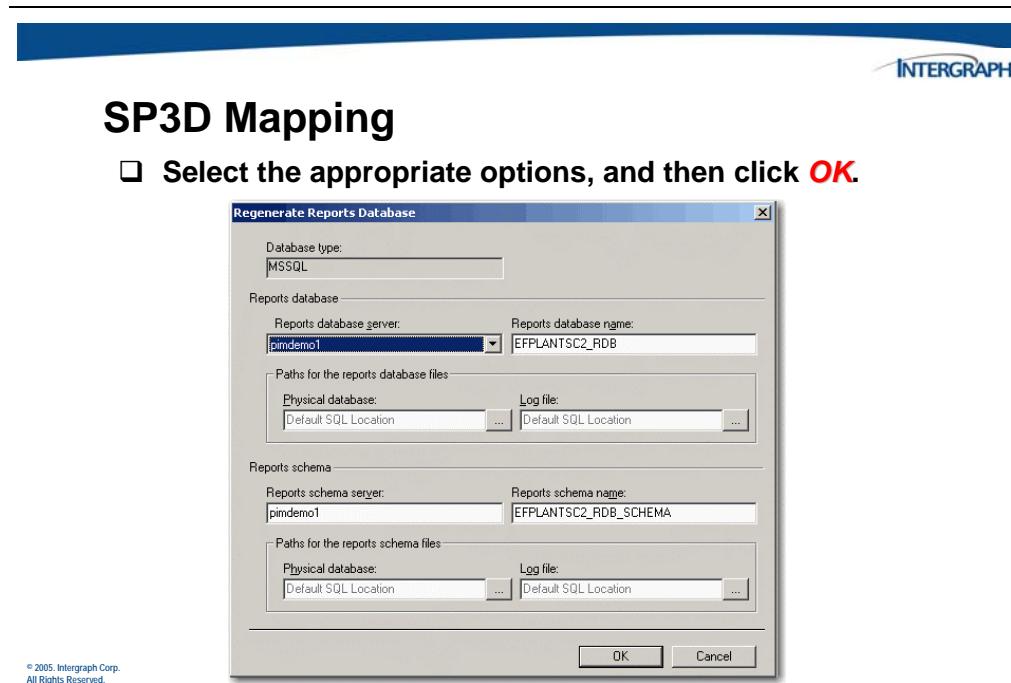
- The following warning box and progress dialog will appear.



In the Project Utility, right-click on **EFPLANT-SC-2**, and click **Regenerate Reports Database**.



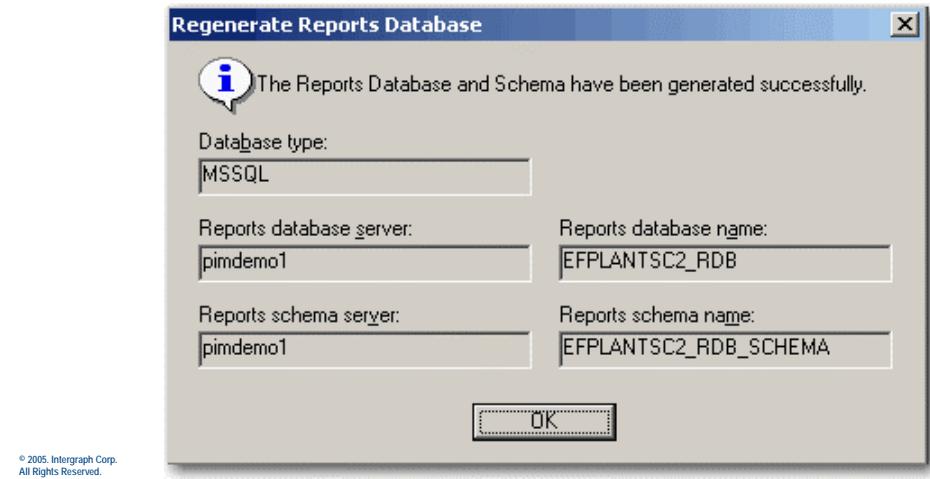
In this dialog, enter the Reports Database Server (**pimdemo1**), The Reports database name (**EFPLANTSC2_RDB**), and the Reports schema name (**EFPLANTSC2_RDB_SCHEMA**), and click **OK**.



The following dialog will appear at the end of the report load.

SP3D Mapping

- When the process is complete, the following message will appear.



Finally, we need to check that the properties were loaded into the SP3D database correctly. Open **SP3D All Programs > Intergraph SmartPlant 3D > SmartPlant 3D**. Cancel the **New** dialog, and select the **Open Session** icon. Select the 128-5001.ses file, and select **Piping Runs** for the objects to work on.



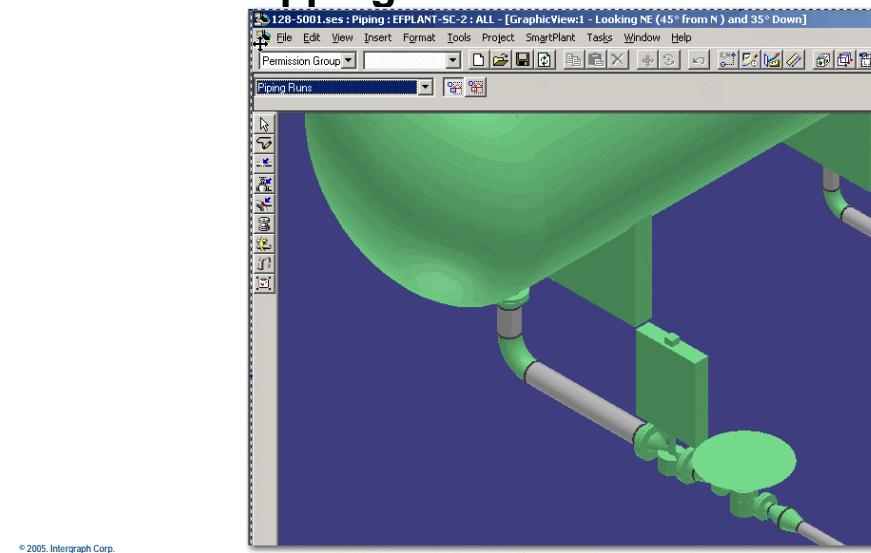
SP3D Mapping

- To check that the properties were loaded into the SP3D database correctly. Open SP3D by clicking **All Programs > Intergraph SmartPlant 3D > SmartPlant 3D**.
- Cancel the **New** dialog, select the **Open Session** icon, and select the **128-5001.ses** file.
- Select **Piping Runs** for the objects to review.

© 2005, Intergraph Corp.
All Rights Reserved.



SP3D Mapping

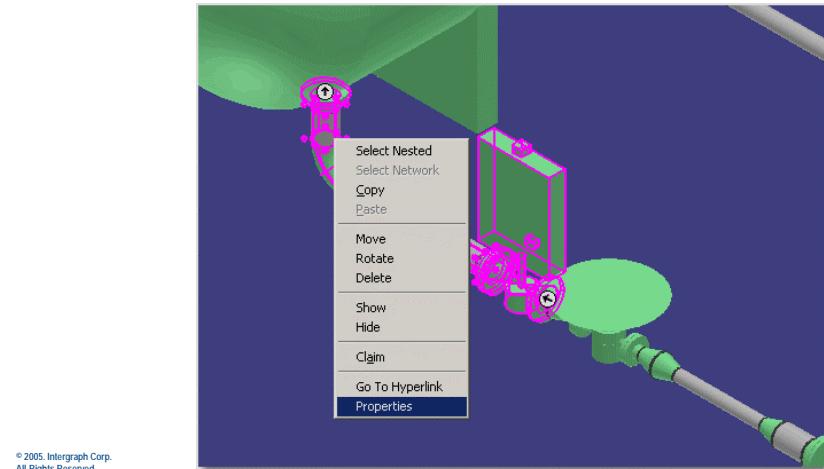


© 2005, Intergraph Corp.
All Rights Reserved.

Find the pipe run shown in the diagram below, right-click for the pop-up menu, and click **Properties**.

SP3D Mapping

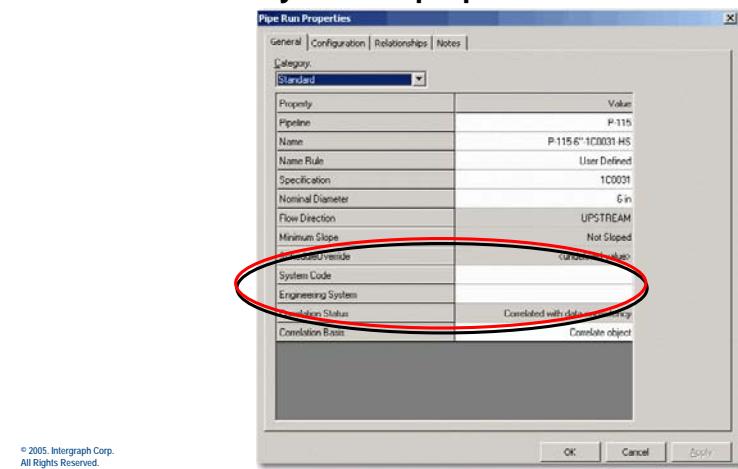
- Find the pipe run shown in the diagram below, right-click for the pop-up menu, and select **Properties**.



In the **Properties** window under the **Standard** category, verify the two properties are in the dialog and that for EngineeringSystem, the code list entries appear.

SP3D Mapping

- In the Properties window, choose the **Standard** category, and verify the two properties.



12.12 Publishing 3D Map Data: An Overview

When you work in an integrated environment with SmartPlant Enterprise, you must publish documents containing the drawing data and relationships before other authoring tools can share this information. You can publish your documents within the Drawings and Reports task.

The **Publish** command is available for the following document types:

- 3D Model Data (SmartPlant Review file type)
- 3D Cable Data (smartPlant Review file type)
- Orthographic Drawings (viewable file with links to data)
- Piping Isometric Drawings (viewable file with links to data)
- Reports (viewable file with links to data)

The viewable files created when you publish drawings and reports provide relationship links to the 3D Model Data. You must also publish the 3D Model Data to provide the navigation between the viewable files and the 3D Model Data.



SP3D Mapping

- Publish is available for the following document types in SP3D:**
 - 3D Model Data (SmartPlant Review file type)
 - 3D Cable Data (SmartPlant Review file type)
 - Orthographic Drawings (viewable file with links to data)
 - Piping Isometric Drawings (viewable file with links to data)
 - Reports (viewable file with links to data)
- The viewable files created when you publish drawings and reports provide relationship links to the 3D Model Data. You must also publish the 3D Model Data to provide the navigation between the viewable files and the 3D Model Data.**

© 2005, Intergraph Corp.
All Rights Reserved.

When you publish documents, the software:

- Publishes a visual representation of the document that you can view without SmartPlant 3D. For drawings, this is an Intergraph proprietary file, called a RAD file (.sha). For reports, the viewable file is a Microsoft Excel workbook. You can review and mark-up the

visual representation of the document using SmartPlant Markup or SmartSketch.

- Places the published XML file and any viewable files in the appropriate SmartPlant Foundation vault. This XML file can be retrieved when users are in other authoring tools.

SmartPlant 3D receives notification when the publish is complete. The software stores the XML file in the appropriate location and loads the data into the database. The XML file can then be retrieved as published data by other tools, such as SmartPlant Electrical.



SP3D Mapping

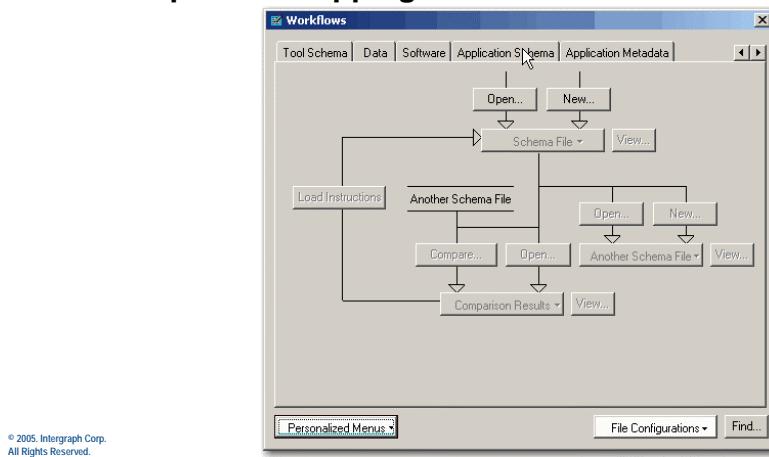
- When you publish documents,
 - The software will publish a visual representation of the document that can be viewed without SmartPlant 3D.
 - Reports are shared as a viewable file in Microsoft Excel workbook format.
 - You can review and mark-up the visual representation of the document using SmartPlant Markup or SmartSketch.
 - The software places the published XML file and any viewable files in the appropriate (registered) SmartPlant Foundation vault.
 - The XML file can then be retrieved as published data by other tools, such as SmartPlant Electrical.

12.13 Mapping Configuration for Publish in SP3D

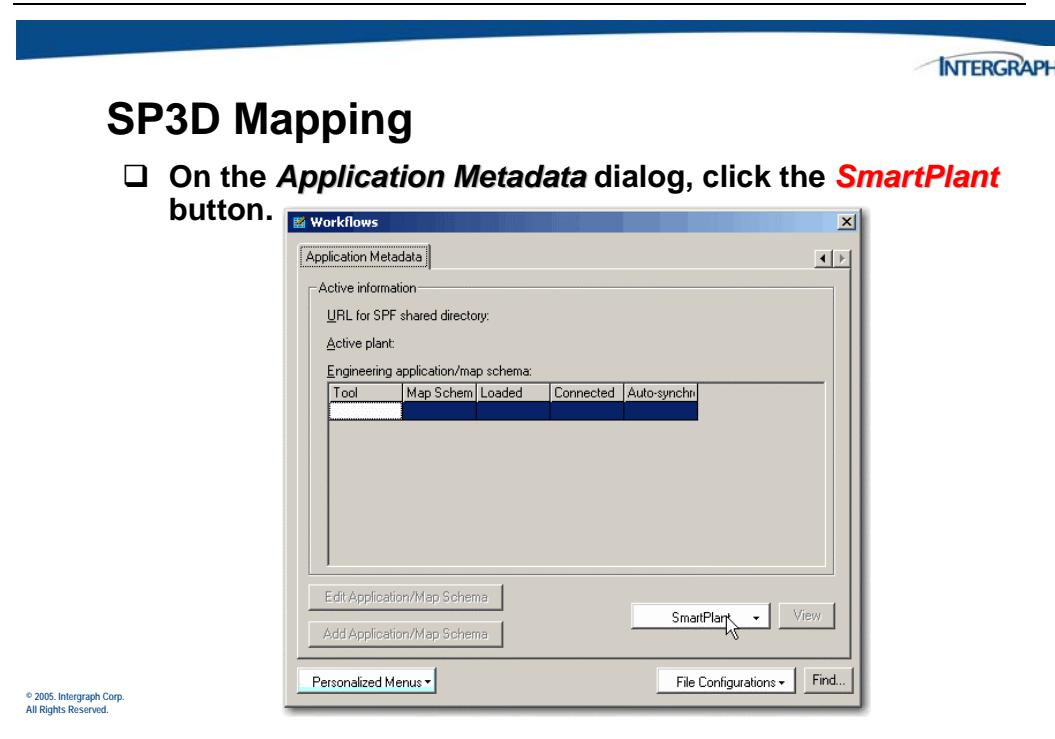
Open the SmartPlant Schema Editor utility, and in the Workflow dialog, select the **Application Metadata** tab to enter the mapping wizard.

SP3D Mapping

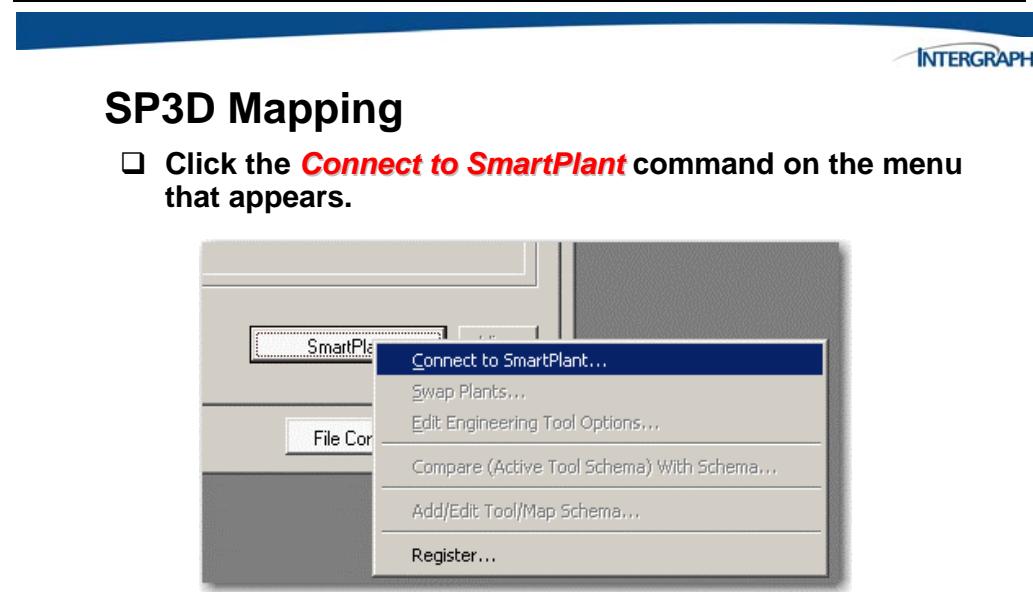
- Open the SmartPlant Schema Editor utility. In the **Workflow** dialog, select the **Application Metadata** tab to open the mapping wizard.



In the **Application Metadata** dialog, click the **SmartPlant** button.



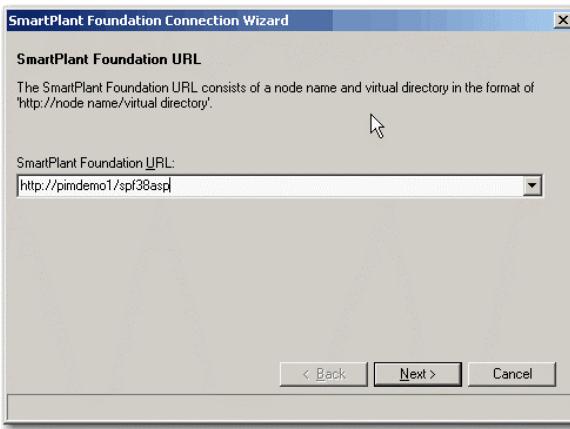
Click the **Connect to SmartPlant** command.



In the SmartPlant Foundation Connection Wizard, enter the SPF URL string for your SPF Server. In the SmartPlant Foundation URL, enter the string <http://pimdemo1/spf38asp>, and click **Next**.

SP3D Mapping

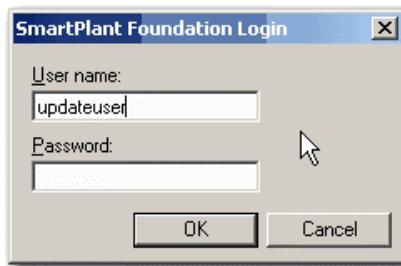
- The standard SmartPlant Foundation Connection Wizard opens. For the SmartPlant Foundation URL, enter the string <http://pimdemo1/spf38asp>, and then click Next.



SmartPlant Foundation will prompt you for a User login. In this lab use updateuser (No Password).

SP3D Mapping

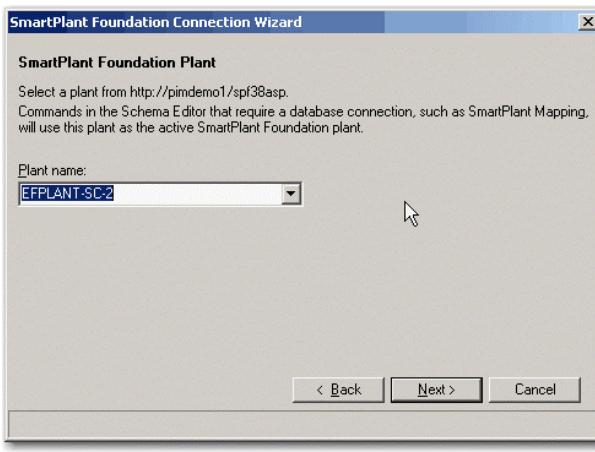
- SmartPlant Foundation prompts you for a User login. In this example, we will use updateuser (no password).



Select the plant to which your SP3D plant is registered. In this class, we will use **EFPLANT-SC-2**

SP3D Mapping

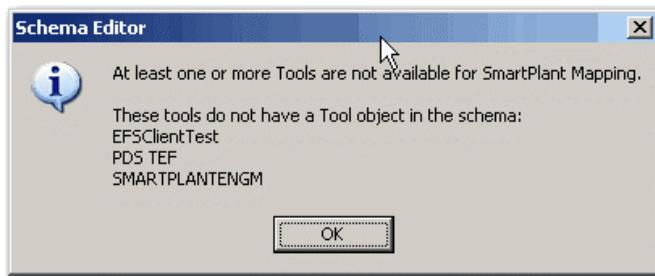
- Select the SPF plant to which your SP3D plant is registered. In this example, we will use **EFPLANT-SC-2**. Select it, and click **Next**.



The Schema Editor will now connect to the SmartPlant Schema file (EFSchema.xml) and any registered application to SPF. The following dialog will open when the connection is completed. These are applications which do not have Tool Schemas. Click **OK**.

SP3D Mapping

- Schema Editor will now connect to the SmartPlant Schema file (EFSchema.xml) and any registered application to SPF**
- The following dialog will open when the connection is completed.**
- It lists applications that do not have Tool Schemas. Click **OK**.**

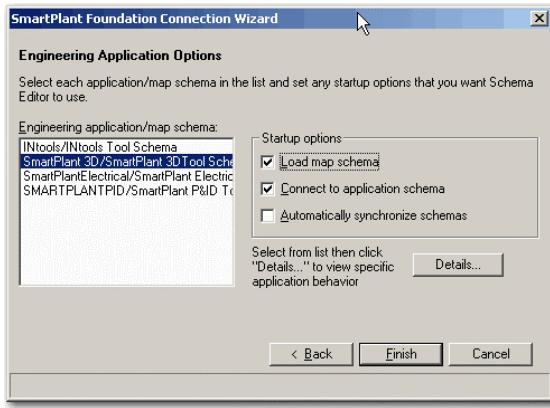


© 2005. Intergraph Corp.
All Rights Reserved.

The connection wizard will now display the Tool Schema list. Select **SmartPlant 3D** from the list, and check the **Load map schema** check box. The **Connect to application schema** will be checked automatically. Click the **Finish** button.

SP3D Mapping

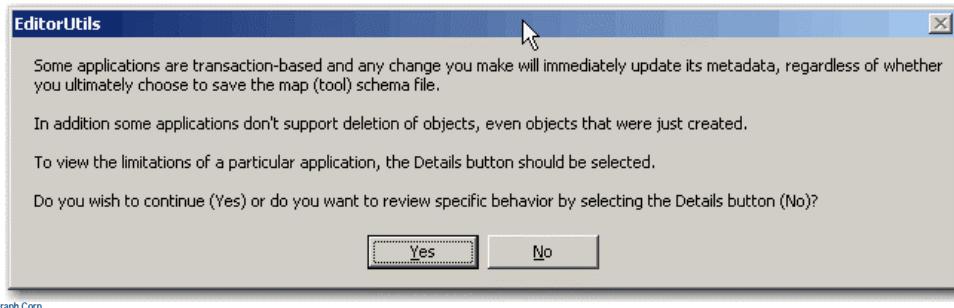
- Select **SmartPlant 3D** from the list, and check the **Load map schema** check box.
- The **Connect to application schema** check box will be checked automatically. Click **Finish**.



A warning dialog will appear. This dialog warns you that some tools (such as PID or SP3D) will store the data in the Tool database. Making changes to the schema file will also update the database, and in these cases cannot be removed from the database. Make your changes carefully; also we recommend you back up your tool databases so you can recover it if needed.

SP3D Mapping

- The following warning dialog will appear.**
- This dialog warns you that some tools (PID, SP3D,etc) will store the data in the Tool database.**
- Make your changes carefully. Additionally, we recommend that you back up your tool databases so you can recover it later, if needed.**



The system will now compare the mapping file for the tool and the tools database. The next dialog displays all the properties that are different and on which side the object is missing. In this example, the class setup shows IUPBSItem is missing in the SP3D map file along with the other objects listed. Click **OK** to synchronize the map file with the tools database.



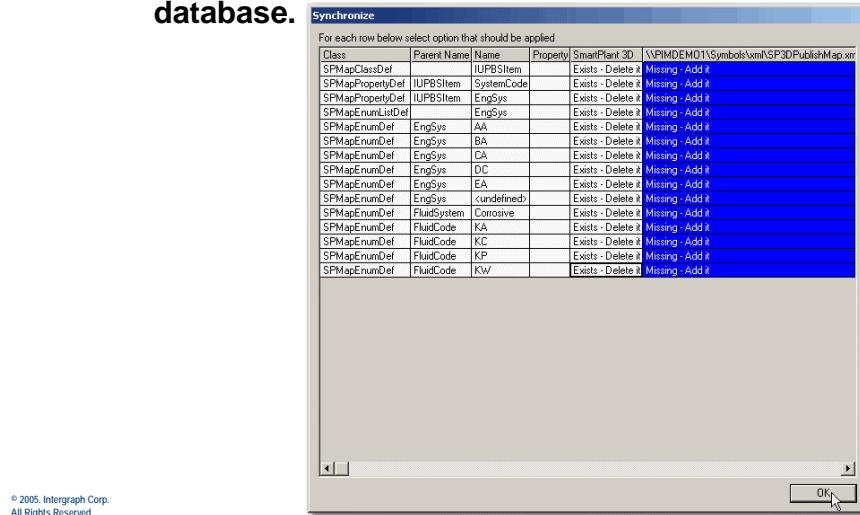
SP3D Mapping

- The system will now compare the mapping file for the tool and the tools database.**
- In this example, the class setup shows IUPBSItem is missing in the SP3D map file along with the other objects listed.**



SP3D Mapping

- Click **OK** to synchronize the map file with the tools database.

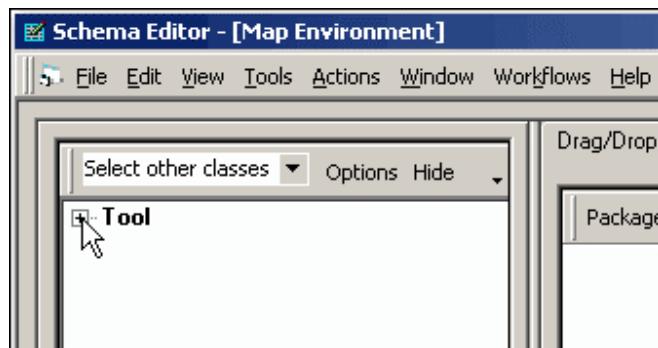


The Schema Editor Edit window will open. Drill down to open the object in the SP3D map configuration. The next slides show this operation.



SP3D Mapping

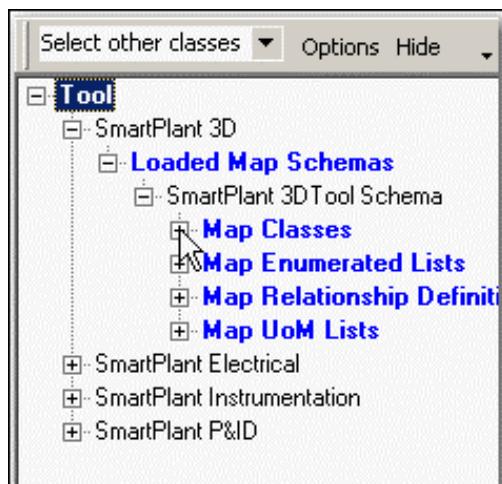
- Drilldown to open the object in the SP3D map configuration.
- Select the (+) to expand the Tool entry.





SP3D Mapping

- Under Tools expand **SmartPlant 3D**, then **Loaded Map Schemas**, then **SmartPlant 3D Tool Schema**, and then **Map Classes**.

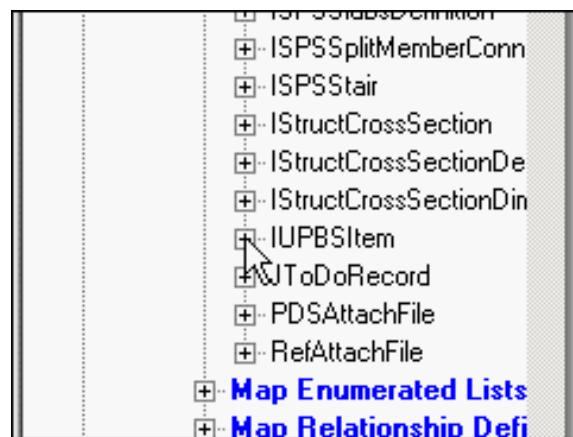


Select (+) for IUPBSItem.



SP3D Mapping

- Under **Map Classes**, expand **IUPBSItem**.

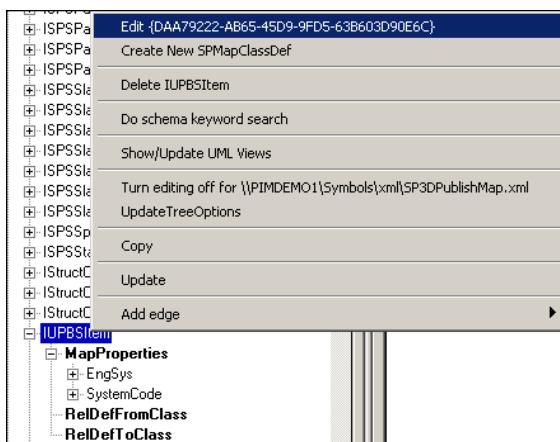


Right click on **IUPBSItem**, and select **Edit**.



SP3D Mapping

- Right-Click on IUPBSItem, and click **Edit**.



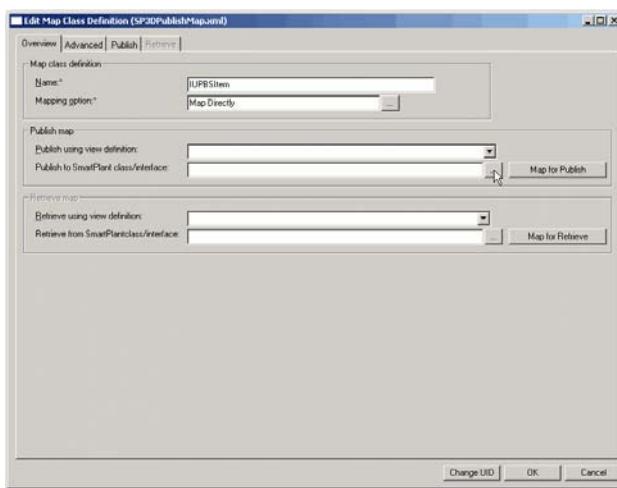
© 2005. Intergraph Corp.
All Rights Reserved.

The **Edit Map Class Definition** dialog appears. Click the Browse button for the **Publish to SmartPlant class/Interface**.



SP3D Mapping

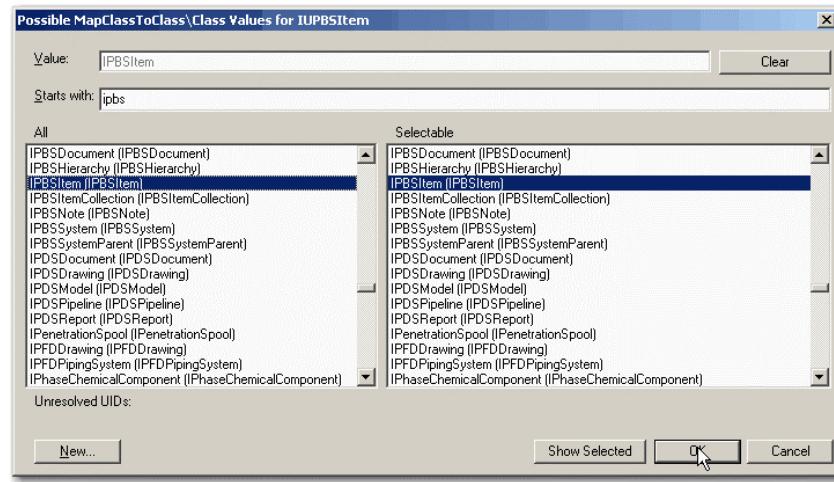
- The *Edit Map Class Definition* dialog appears. Click the browse button for **Publish to SmartPlant class/Interface**.



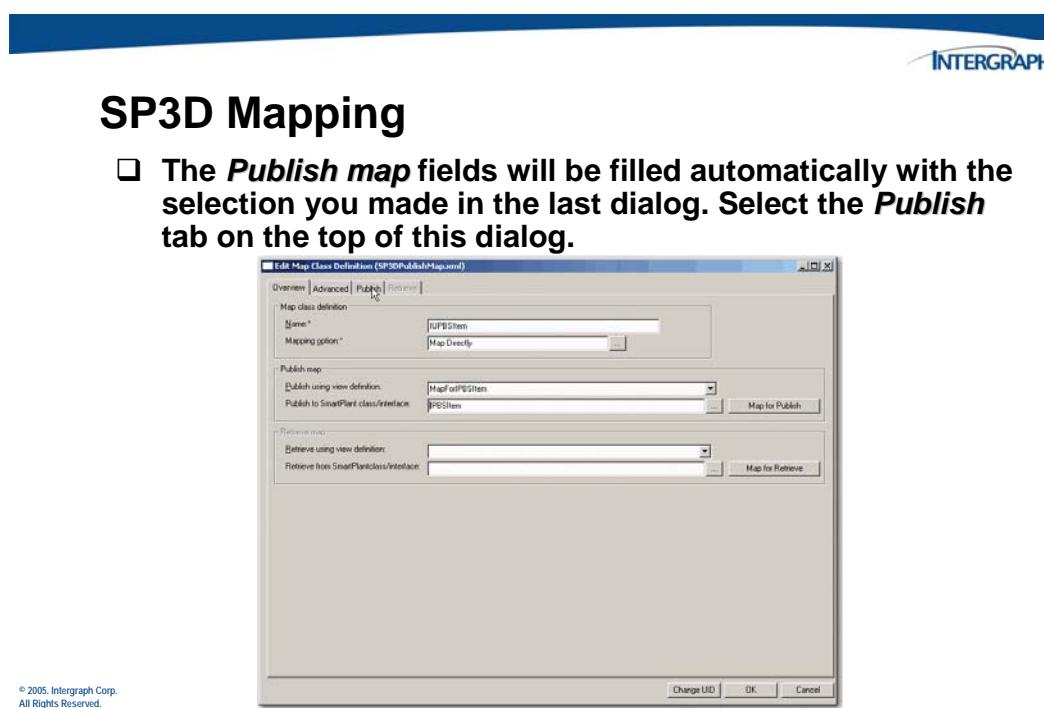
Select (highlight) **IPBSItem** (EFSchema object), and click **OK**.

SP3D Mapping

- Select **IPBSItem** (EFSchema object) in the **Selectable** list, and click **OK**.



The **Publish map** fields will be filled automatically with the selection you made in the last dialog. Select the **Publish** tab on the top of this dialog.



The **Edit Map Class Definition** dialog appears. The left side is the SP3D map data, and the right side is the SPF configuration. Select **EngineeringSystem** on the Tool side and **EngineeringSys** on the SPF side. In the center of this dialog, click the **Map** button to map these selected properties.

SP3D Mapping

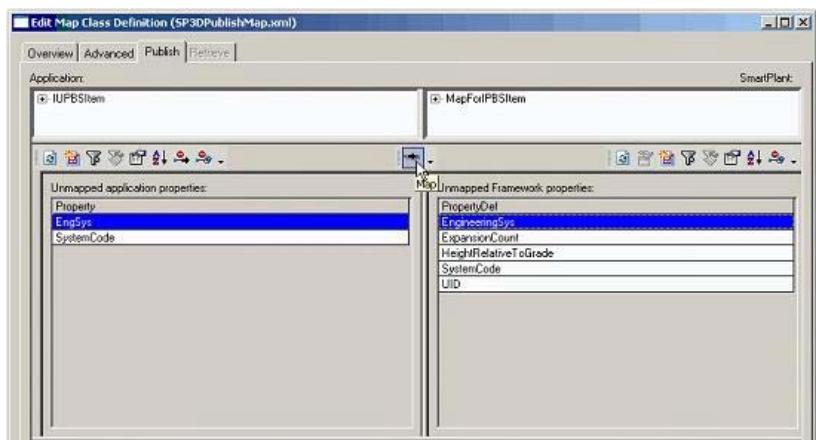
- The Edit Map Class Definition dialog appears.
- The left side is the SP3D map data.
- The right side is the SPF configuration.

© 2005. Intergraph Corp.
All Rights Reserved.

SP3D Mapping

- Select **EngSys** on the Tool side and **EngineeringSys** on the SPF side. In the center of this dialog, click the **Map** button to map these selected properties.

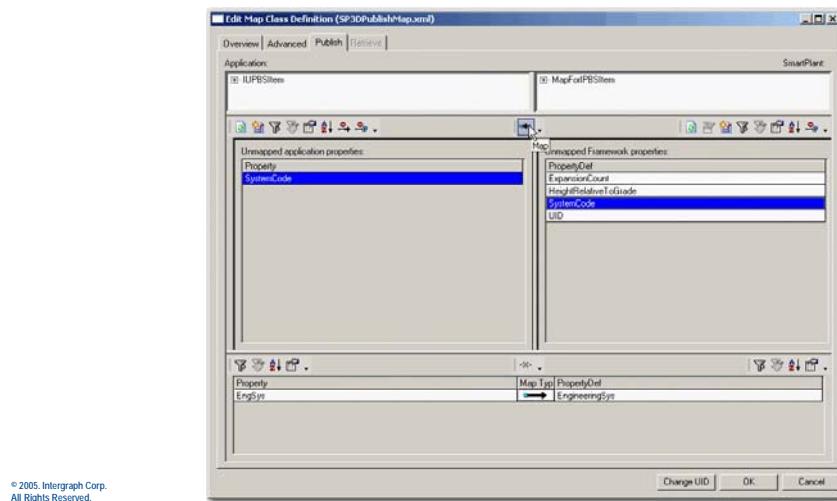
© 2005. Intergraph Corp.
All Rights Reserved.



Perform the same operation for the **SystemCode** / **SystemCode** properties. Click the **Map** button to map these properties.

SP3D Mapping

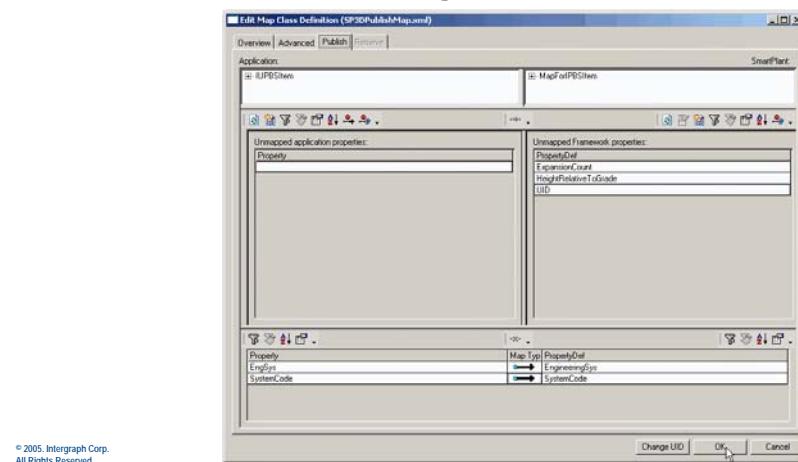
- Perform the same step for the **SystemCode** properties.



Verify both properties are mapped in the bottom section on this dialog. If so, click **OK**.

SP3D Mapping

- Verify that both properties are mapped in the bottom section on this dialog. Click **OK**.

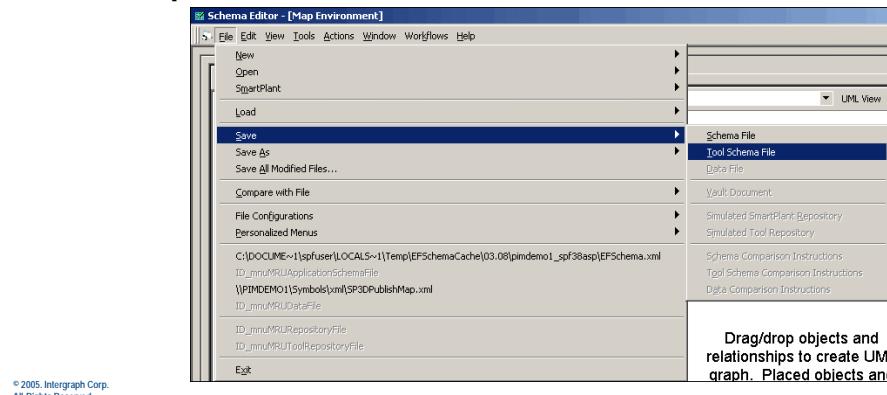


NOTE – Perform the same steps we just did for the Publish Mapping on the Retrieve Mapping. Then perform the next step.

Click **File > Save > Tool Schema File**. Remember, this will update the Tool Schema and application Database (if you added properties in this utility which are not in the Tool Schema or Database).

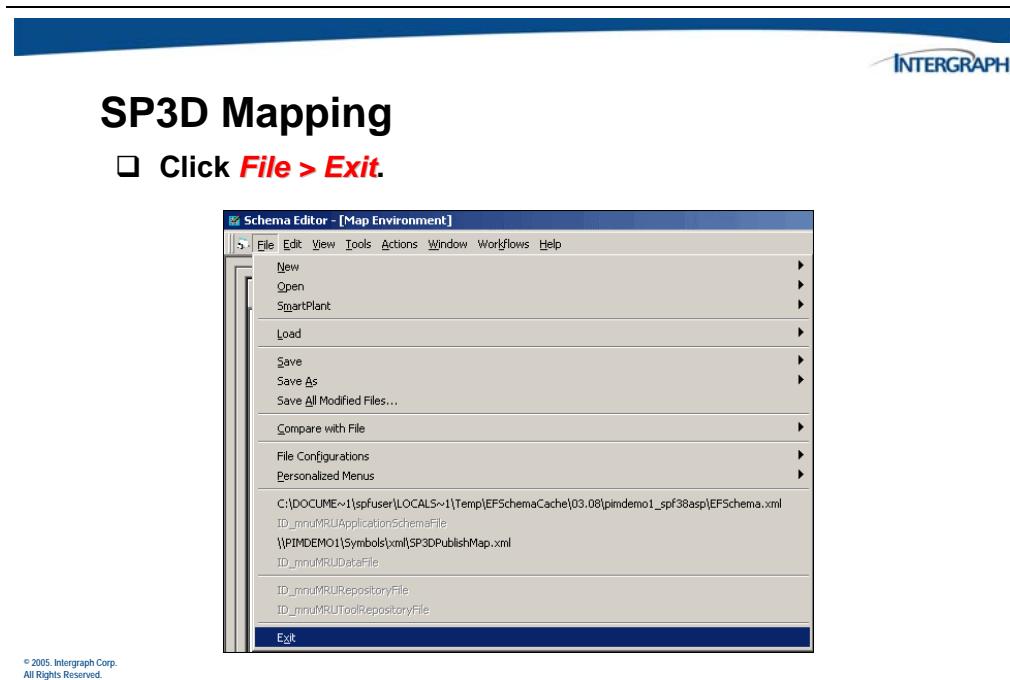
SP3D Mapping

- Click **File > Save > Tool Schema File**.
- Remember, if you added properties in this utility that were not previously in the Tool Schema or Database, this update will add them.



© 2005, Intergraph Corp.
All Rights Reserved.

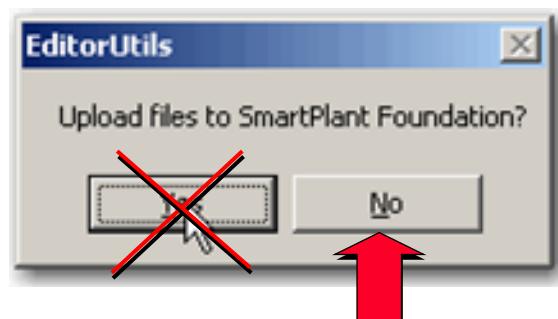
Click **File > Exit.**



When exiting, the Schema Editor will prompt you to Upload the files to SPF. Click **Yes**. This completes the mapping of the properties to SPF Schema.

SP3D Mapping

- When exiting, the Schema Editor will prompt you to Upload the files to SmartPlant Foundation. Click **No**.

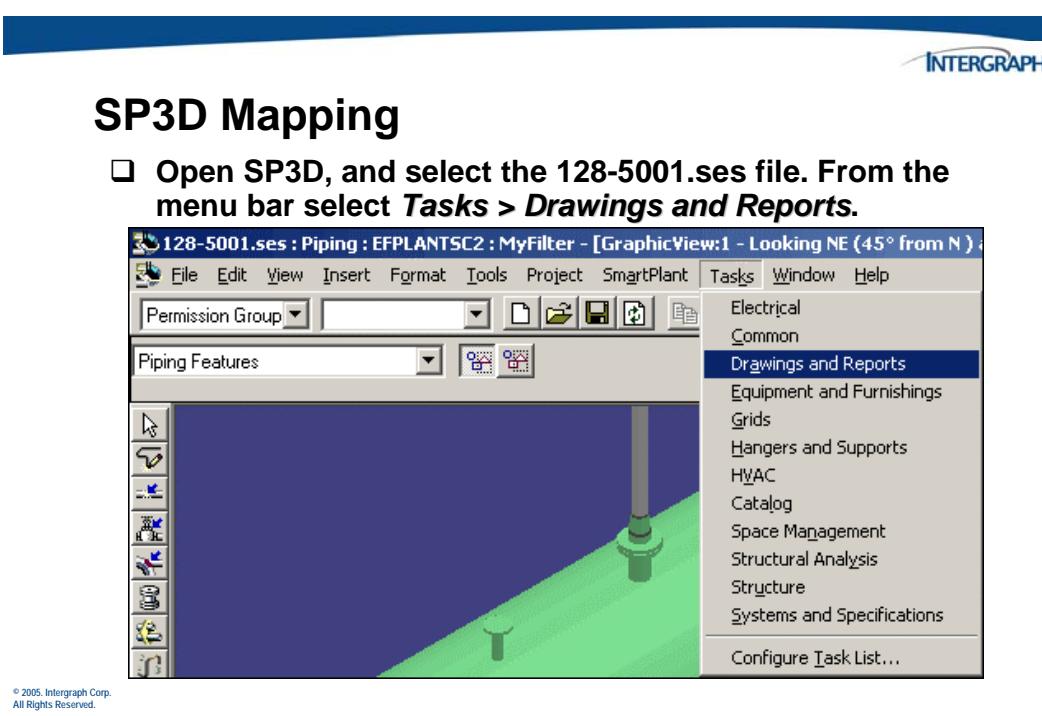


Note

- Before you verify the mapping, the EFSchema change must be loaded into SPF using the SchemaLoader utility.

12.14 Verify Mapped data from SP3D

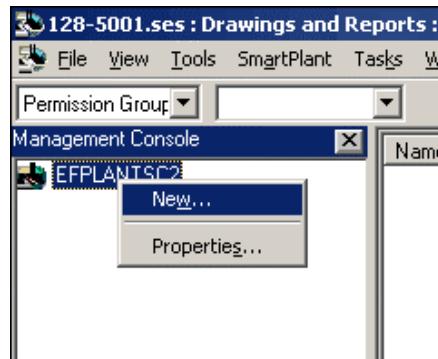
Open SP3D, and select the 128-5001.ses file. From the menu bar, click **Tasks > Drawings and Reports** task.



In the Management Console, right click on the plant **EFPLANTSC2**, and click **New**.

SP3D Mapping

- In the Management Console, right-click on the plant **EFPLANTSC2**, and click **New**.

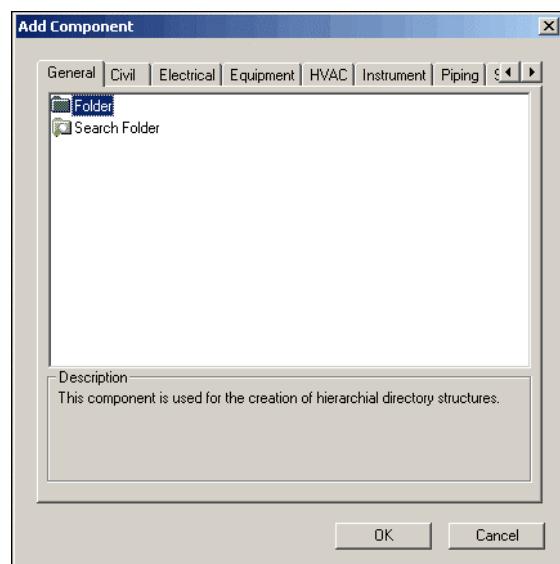


© 2005. Intergraph Corp.
All Rights Reserved.

On the **Add Component** dialog, highlight **Folder** under the general tab, and click **OK**.

SP3D Mapping

- On the **Add Component** dialog, select the **Folder** option on the **General** tab, and click **OK**.

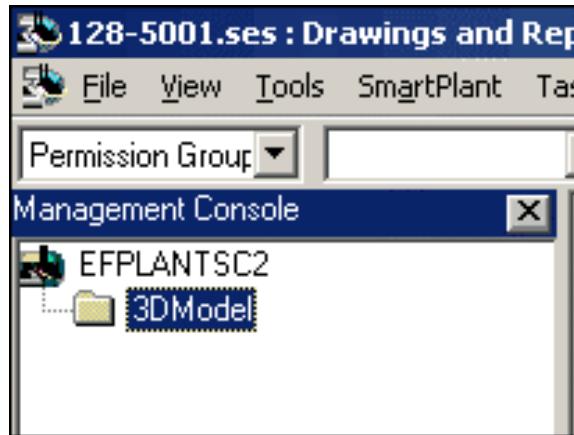


© 2005. Intergraph Corp.
All Rights Reserved.

Rename the folder to reflect a name that describes the folder. We will use **3Dmodel**.

SP3D Mapping

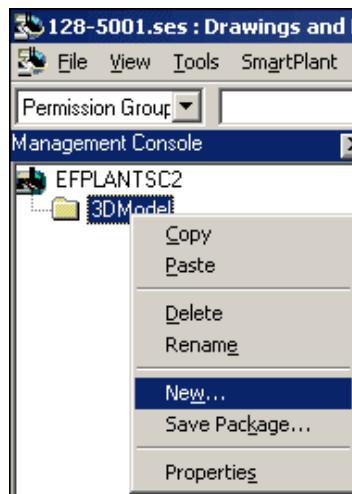
- Right-click on the new folder, and click **Rename** to change its name to something reflective of its purpose. In our example, we will use the name **3DModel**.



Select the **3DModel** folder, right-click, and click **New**.

SP3D Mapping

- Select the **3DModel** folder, right click, and click **New**.

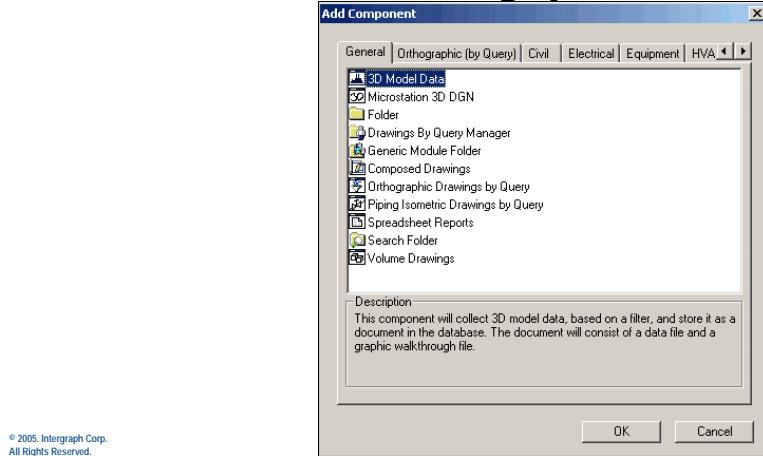


© 2005. Intergraph Corp.
All Rights Reserved.

On the **Add Component** dialog, highlight **3D Model Data**. This is the object type you want to publish. Click **OK** to add 3D Model Data category to this folder.

SP3D Mapping

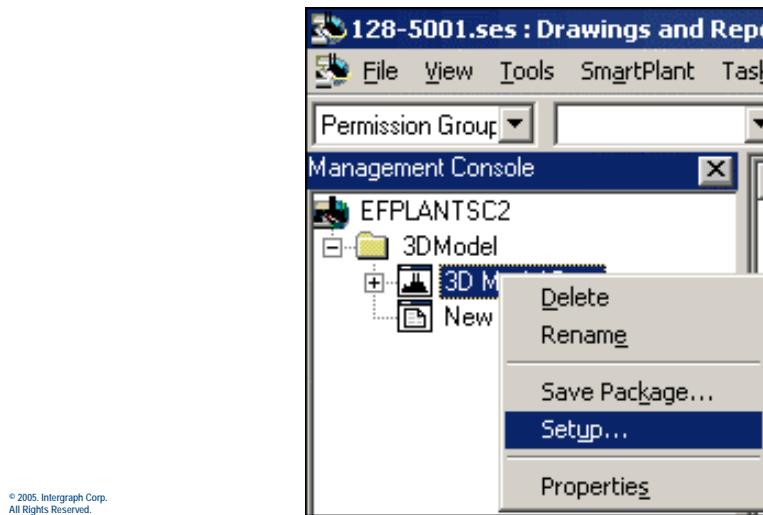
- On the *Add Component* dialog, highlight **3D Model Data**. This is the object type you want to publish. Click **OK** to add 3D Model Data category.**



Right-click the category **3D Model Data**, and click **Setup**.

SP3D Mapping

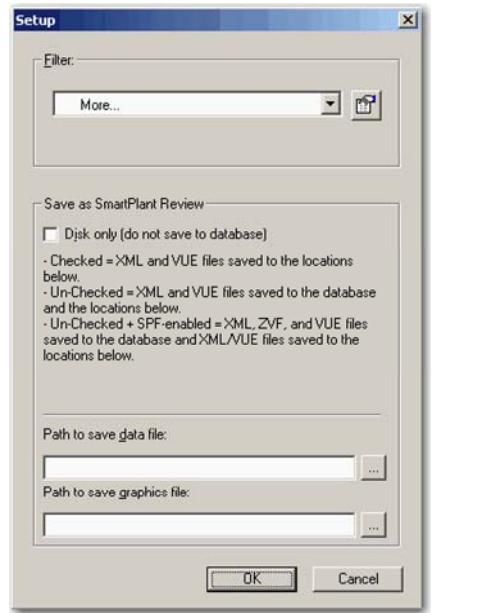
- Right-click the category **3D Model Data**, and click **Setup**.**



This opens the **Setup** dialog. Under the **Filter** section, select **More...** to select which filter you will use.

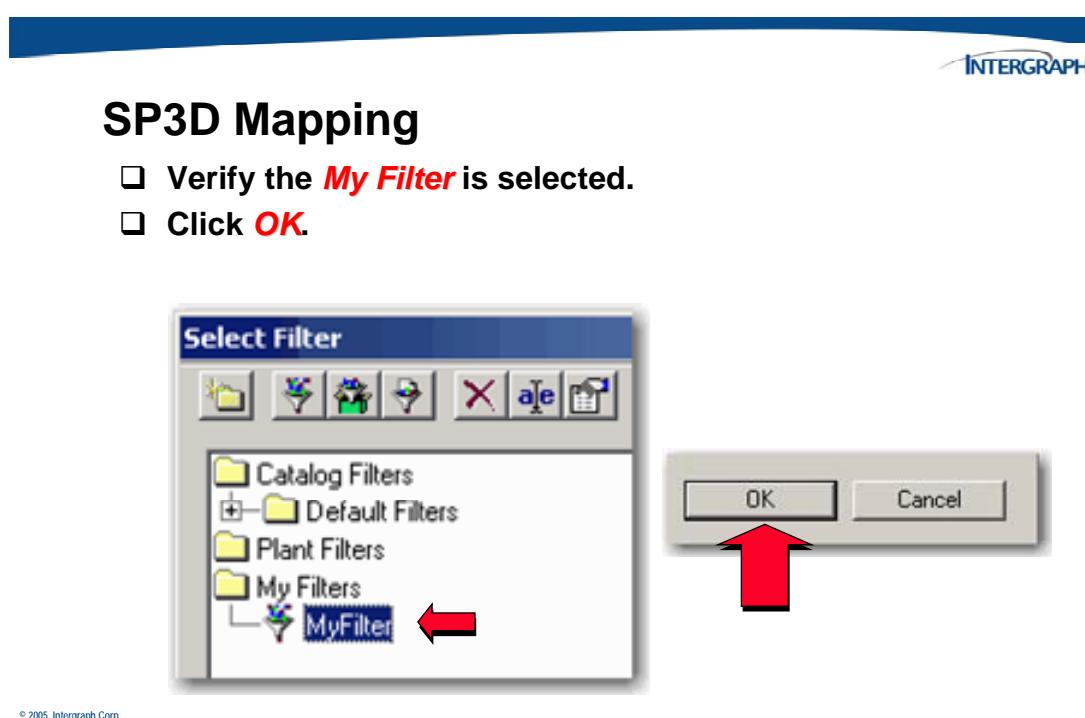
SP3D Mapping

- Under Filter, select **More...** For the next dialog.



© 2005, Intergraph Corp.
All Rights Reserved.

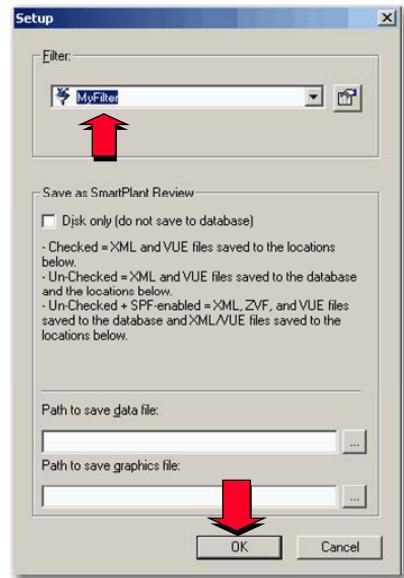
The **Select Filter** dialog will appear. Select the filter we just created (**My Filter**).
Click **OK**



The **Setup** dialog will open. Verify the **My Filter** is selected. Click **OK** to continue.

SP3D Mapping

- Verify the My Filter is selected**
- Click OK to continue**

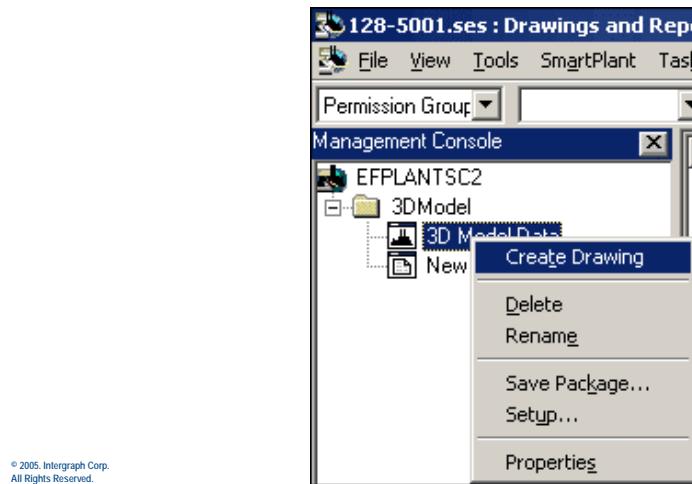


© 2005, Intergraph Corp.
All Rights Reserved.

Next, create a publish drawing object by right-clicking the **3D Model Data** and clicking **Create Drawing**.

SP3D Mapping

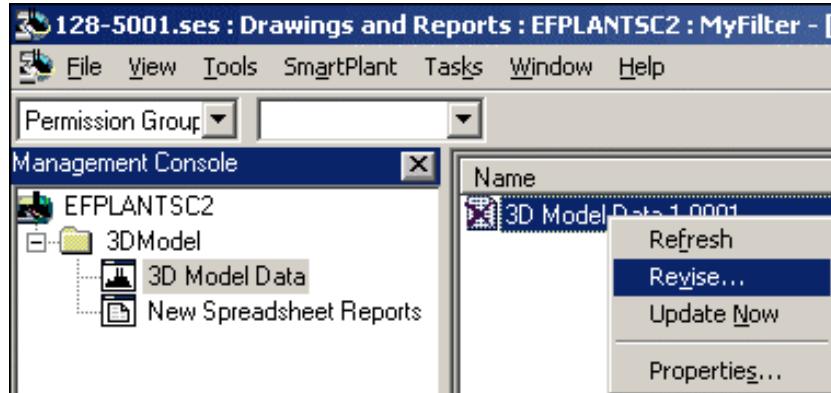
- To create a publish drawing object, right-click the **3D Model Data**, and click **Create Drawing**.



An empty drawing template is created. Select it, right-click it, and click **Revise**. All published drawing must be revised for input into SPF.

SP3D Mapping

- An empty drawing template is created.
- Right-click it, and click **Revise**.

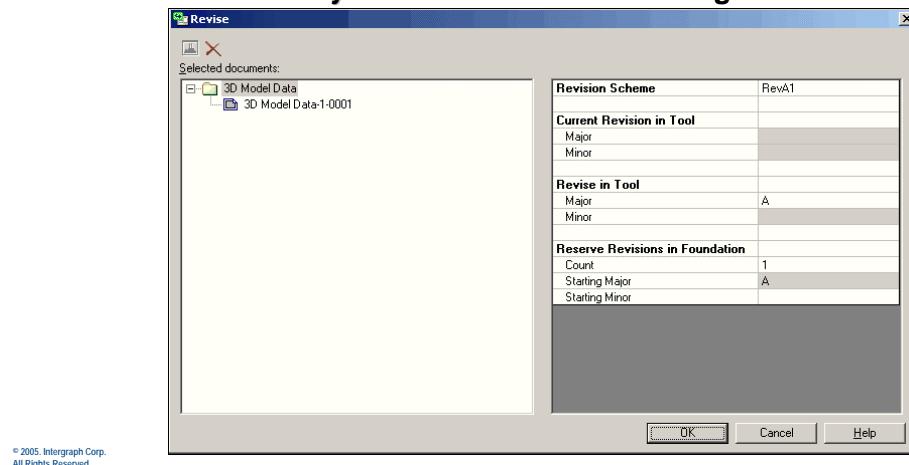


© 2005, Intergraph Corp.
All Rights Reserved.

The **Revise** dialog will open. Select the **Revision Scheme** (we used RevA1) you want to use. Then select the Major Revision for this drawing. Select **First**, then select **OK**

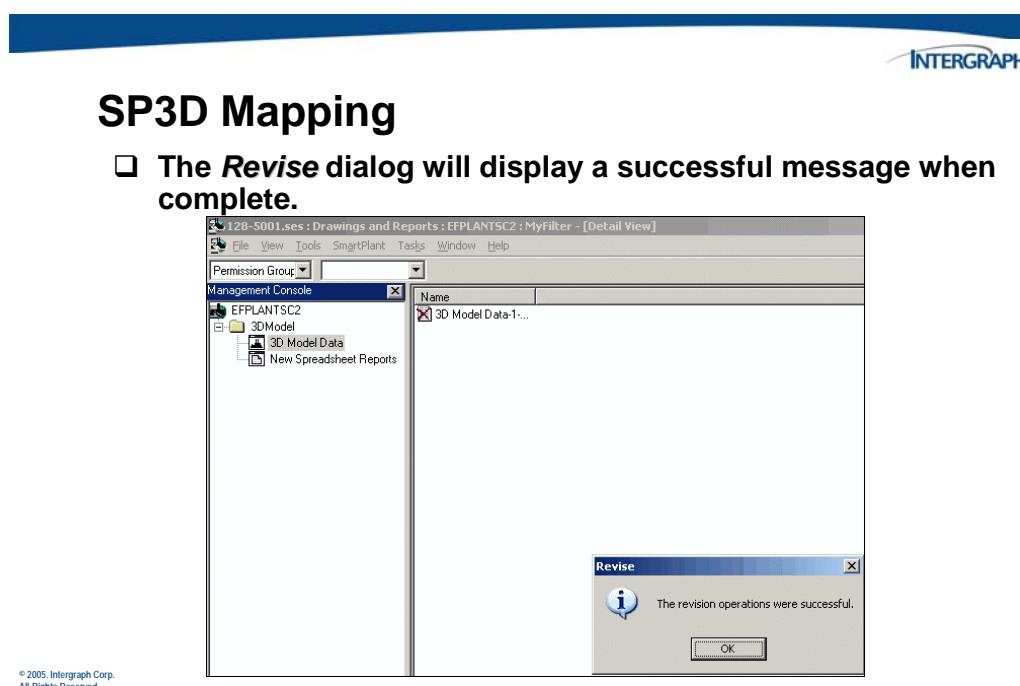
SP3D Mapping

- Choose a **Revision Scheme** (RevA1) you want to use. Then select the **Major Revision** for this drawing. Click **OK**

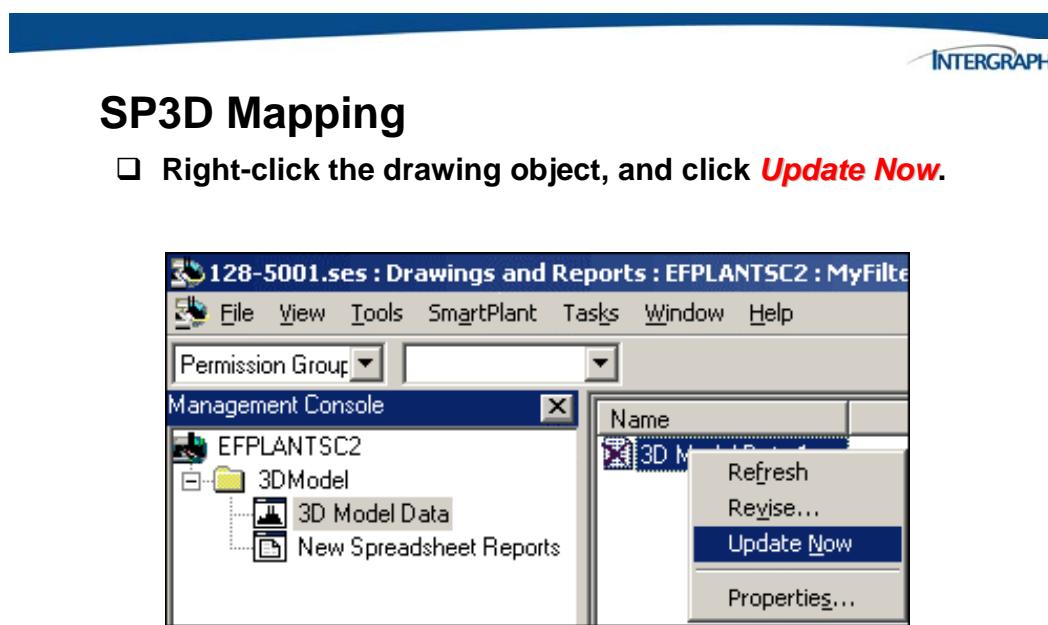


© 2005, Intergraph Corp.
All Rights Reserved.

The **Revise** dialog will display a successful message when complete.



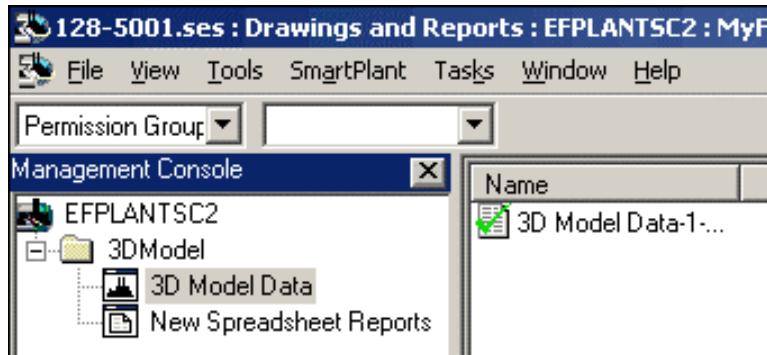
Right-click the drawing object, and click **Update Now**.



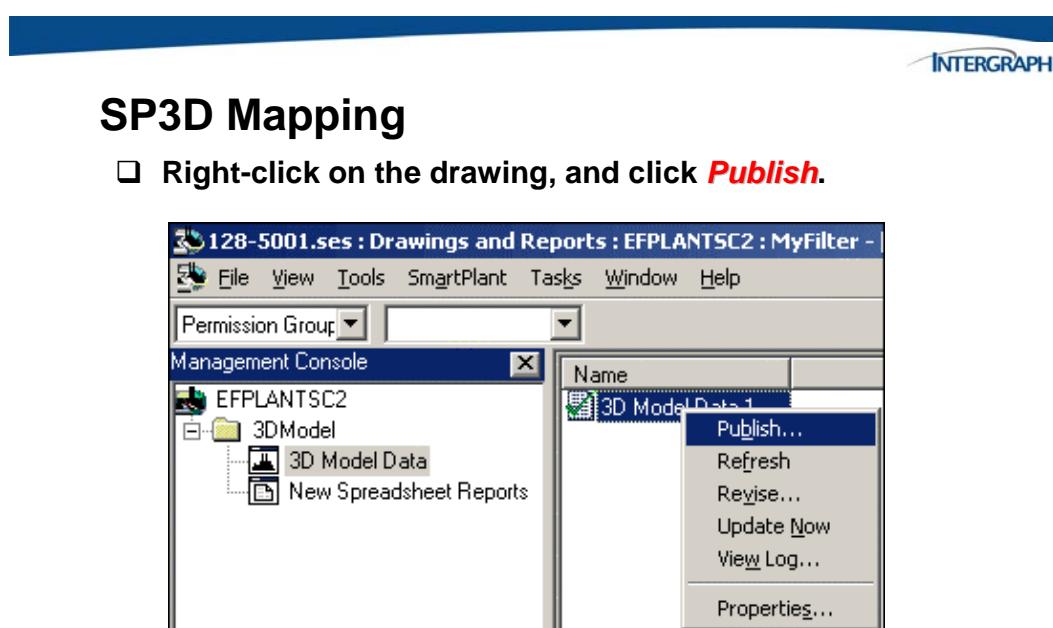
When complete, a green check will appear on the icon for the drawing.

SP3D Mapping

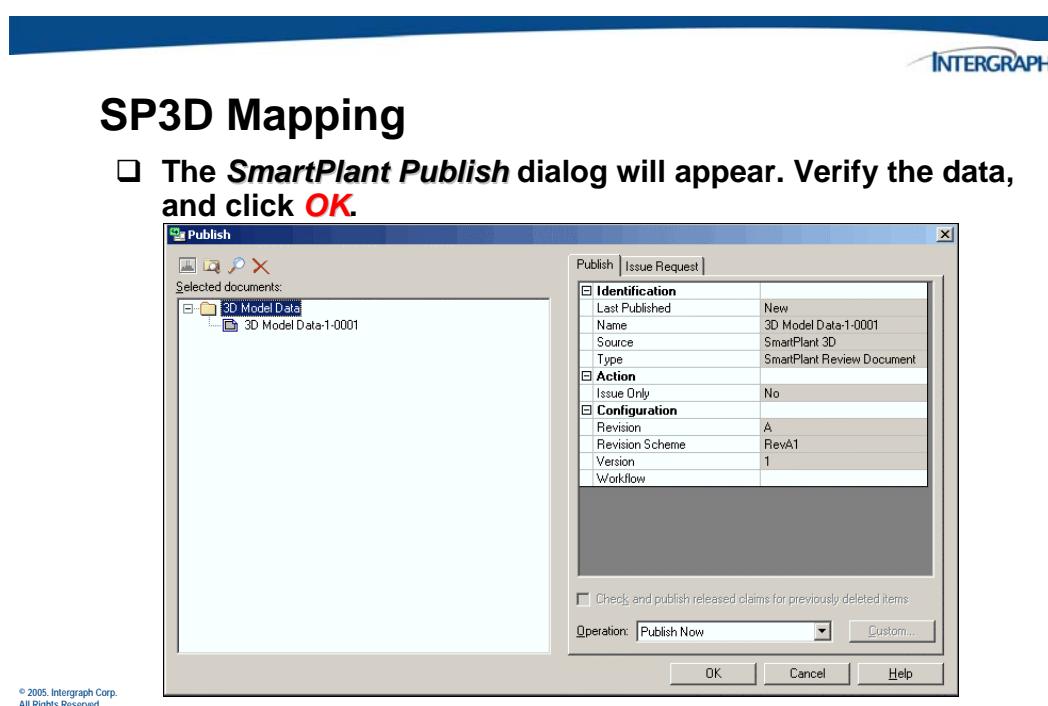
- When the update is complete, a green check will appear on the icon for the drawing.



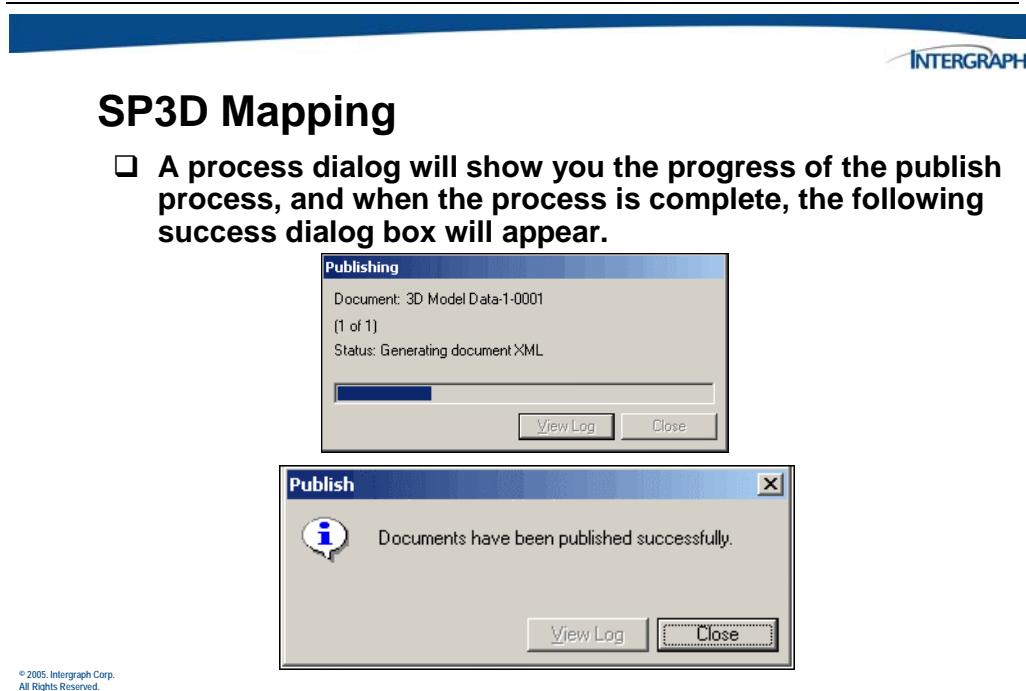
The drawing object is setup to be published. Right-click on the drawing, and click **Publish**.



The **SmartPlant Publish** dialog will appear. Verify the data, and click **OK**.



A process dialog will appear, showing you the publish progress, and when completed, a success dialog appears.



To verify the new properties published properly, open the published metadata xml file in the SPF Vault. Under the D:\Smartplant 2007\FTP Vaults\EFPLANT-SC-2_Vault\SP3D\{UID ID NUMBER}.xml. You may need to check the Time/Date of the files to verify which files were published last. Also, the metadata file will be the largest file in the publish group. For the published file, the vault will contain one View file (3D Model) and three metadata files.



SP3D Mapping

- To verify that the new properties were published properly, open the published metadata xml file in the SPF Vault and find the new properties.

```

</Rel>
- <P3DPipeRun>
  - <Object UID="AAER:model{5c8a0144-801d-4495-84e9-7aa26bb96bc0}: @a=0027!!80013# #29703255#465829715">
    Name="P-115-6"-1C0031-HS"
    <PipeRun />
    <PipingConnector PipingConnectorType="" />
    <IProcessPointCollection />
    <PlannedFacility />
    <3DDrawingItem />
    <Connector FlowDirection="#1" />
    <3DRange Range1X="-3.12196177232423 m" Range1Y="-2.8950881966565 m" Range1Z="-2.85551780665112 m" />
    Range2X="3.40336177232423 m" Range2Y="-0.679455773579396 m" Range2Z="1.76131780665112 m" />
    <3DObject SP3D_DateCreated="12/20/2006 1:27:11 PM" SP3D_DateLastModified="12/21/2006 12:58:25 PM" />
    SP3D_UserCreated="PIMDEMO1\spfuser" SP3D_UserLastModified="PIMDEMO1\spfuser" SP3D_ApprovalStatus="#1" />
    <PipeCrossSectionItem NominalDiameter="6 in" />
    <CoatedItem_Interior_Interior_SurfaceTreatmentRequirement="#11" Interior_SurfaceTreatmentType="#20" />
    Interior_CoatingRequirement="#3" Interior_LocType="#3" />
    <CoatedItem_Exterior_SurfaceTreatmentRequirement="#11" Exterior_SurfaceTreatmentType="#20" />
    Exterior_CoatingRequirement="#5" Exterior_LocType="#3" />
    <InsulatedItem_InsulationType="#5" InsulCompType="#27" TotalInsulThick="0.0381 m" InsulTemp="279.2611111111 K" />
    <PBSItem SystemCode="1234" EngineeringSys="#10113" />
    <PipingSpecifiedItem PipingMaterialsClass="1C0031" />
    <NonDrawingItem />
    <PathRun />
    <DistributionConnectionOwner />
  </P3DPipeRun>
- <P3DProcessPoint>

```

© 2005, Intergraph Corp.
All Rights Reserved.

This completes the verification section.

12.15 Activity – Mapping Properties

Review the Retrieve tasks in this chapter. Next, step though the Publish tasks to map the **SystemCode** and **EngineeringSystem** properties using the SmartPlant Schema. Follow the procedure outlined in this chapter to map the two properties into SPF. Change some property entries and Publish. Verify the data publishes properly.

Note:

- Remember enums in SP3D are not mapped in Schema Editor. Enum lists use the index number defined in the enum definition to map one enum to the one in EFSchema.

A P P E N D I X

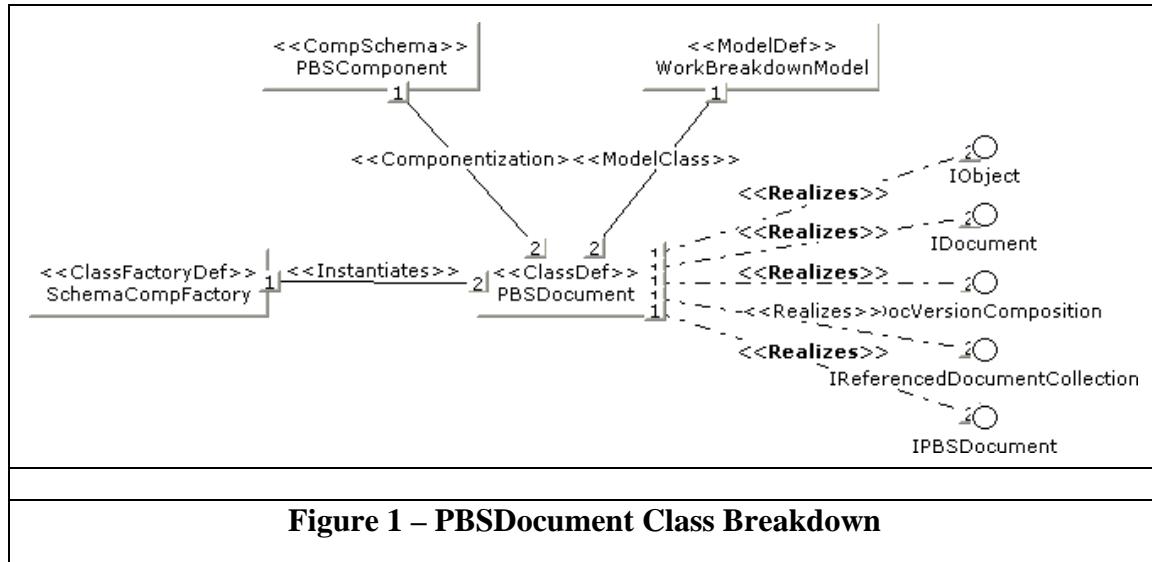
A

SmartPlant Schema Overview

A. SmartPlant Schema Overview

A.1 PBS Component Schema

PBS stands for Plant Breakdown Structure. Plant breakdown is created in SPF and retrieved into the tools to create an identical structure. An administrator can invoke a method displayed as **Publish PBS Document** on the RHM of any Plant. If there have been any changes to the PBS since the last publish, a new document with a ClassDef of PBSDocument is published and entered into the design basis for all tools.



This document is published with a simple HTML view file. Because the document does not realize the INonDrawingItemCollection interface, no relationships are published to the PBS items.

Three types of objects are currently published in the PBS document: ClassDefs of Plant, FunctionalArea, and FunctionalUnit.

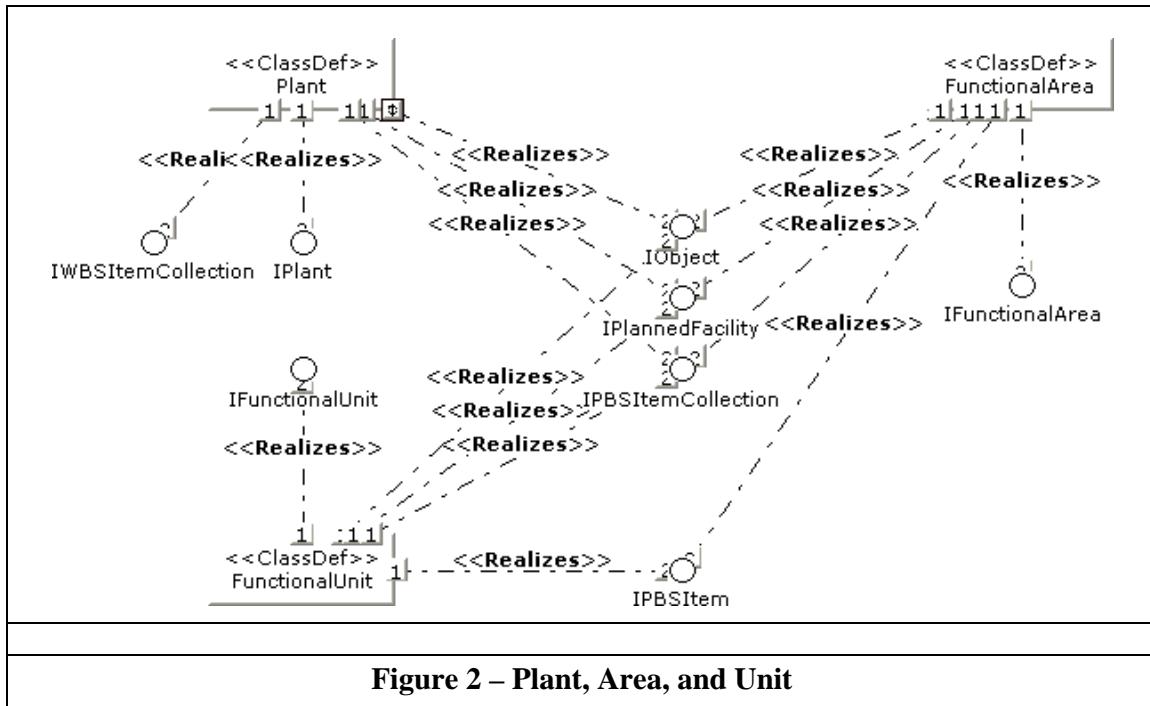


Figure 2 shows simplified class diagrams for Plant, FunctionalArea, and FunctionalUnit. You can see that most of the InterfaceDefs for each are shared among the three ClassDefs. The most important things to notice are that all three ClassDefs realize IPBSItemCollection and that FunctionalArea and FunctionalUnit realize IPBSItem. This creates the structure of the PBS: Plants are a collection of PBSItems, and therefore the Plant is a collection of FunctionalAreas. In the same manner, FunctionalArea is a collection of FunctionalUnits.

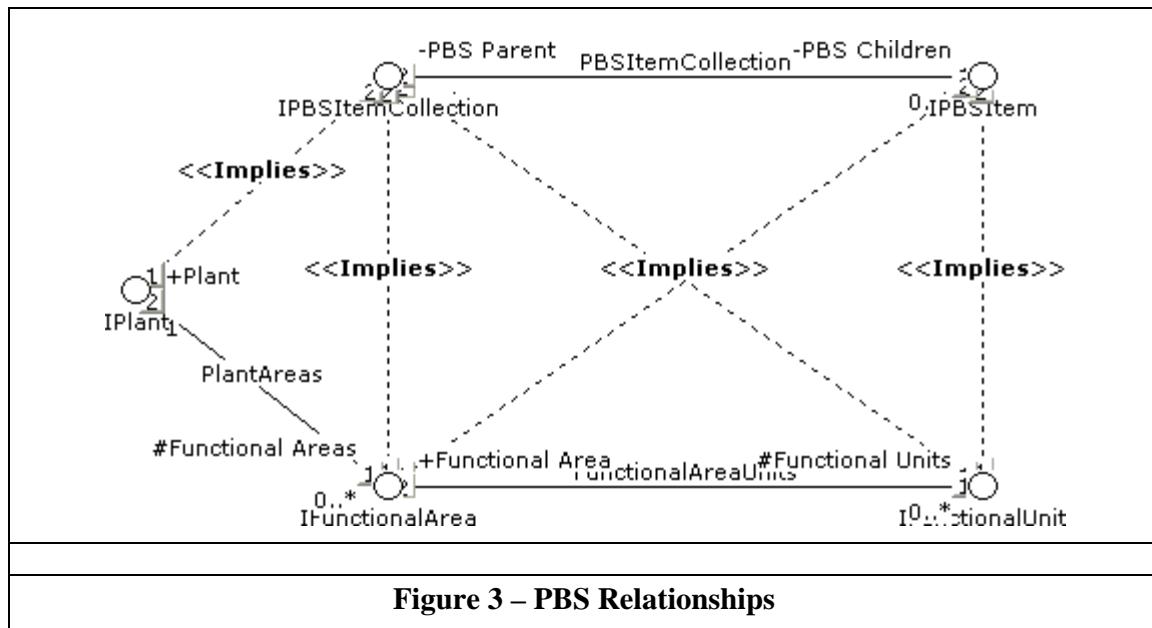
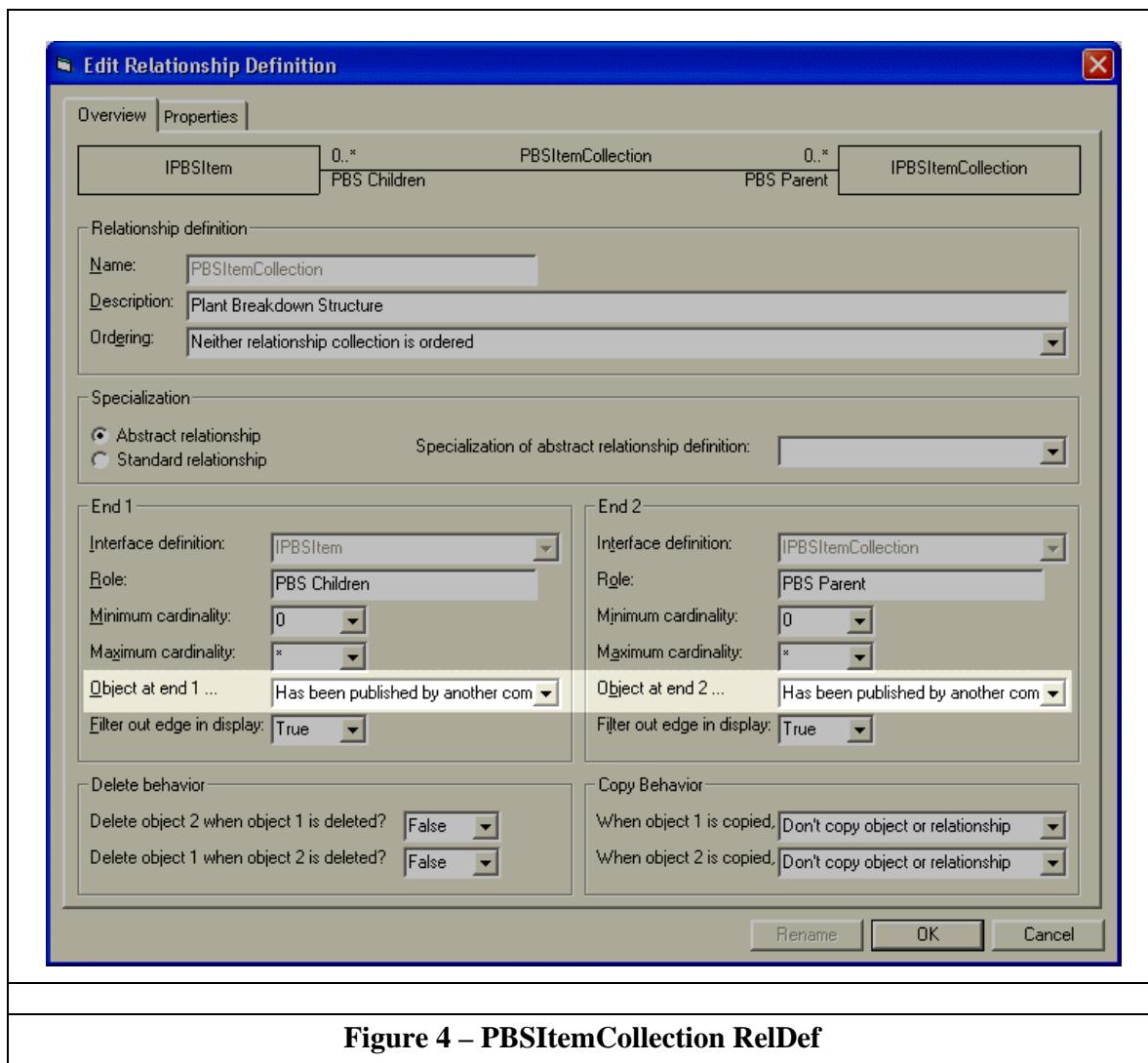


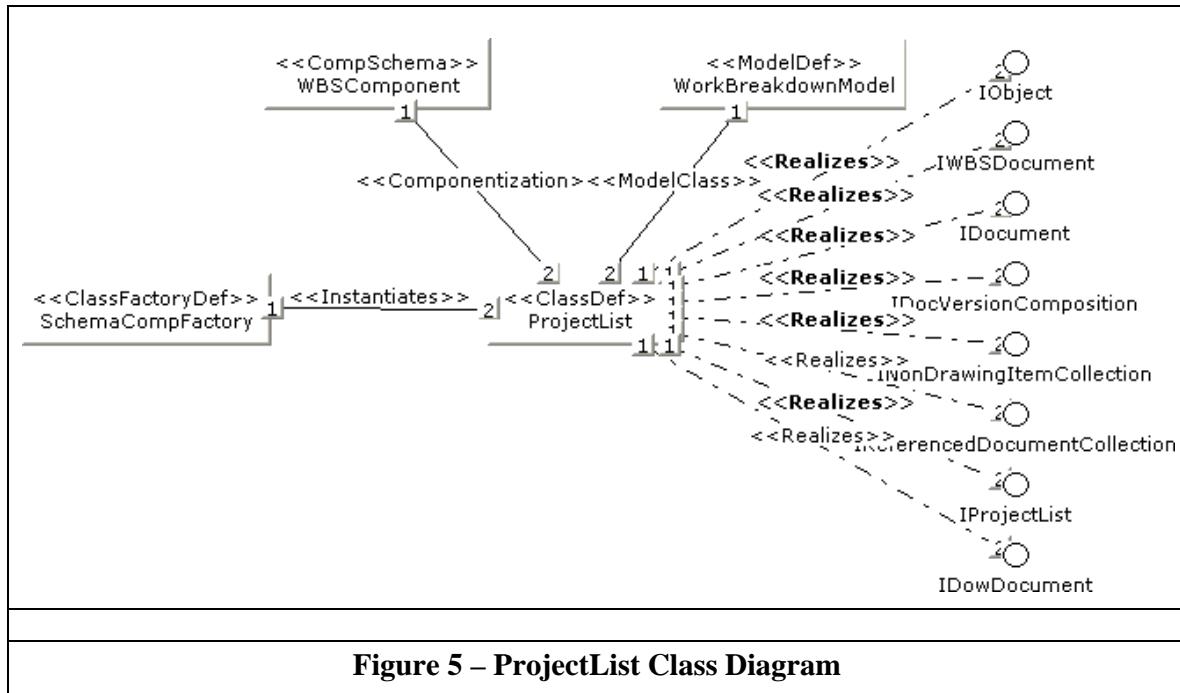
Figure 3 shows the relationships defined between the PBS interfaces. There is a generic RelDef named PBSItemCollection between IPBSItemCollection and IPBSItem. Because the PBS ClassDefs realize those two InterfaceDefs, PBSItemCollection is a valid RelDef for the PBS structure. However, RelDefs are defined between the primary InterfaceDefs, which are specializations of the PBSItemCollection RelDef. The PlantAreas RelDef is defined between IPlant and IFunctionalArea. Likewise, the FunctionalAreaUnits RelDef is defined between IFunctionalArea and IFunctionalUnit.

When the authoring tools publish a document with data, the top-level objects (at least) are usually related to the PBS structure. This is accomplished with a PBSItemCollection relationship between the FunctionalUnit and the PBSItem. Because the tool does not actually publish the FunctionalUnit object, the relationship is dangling for a tool document publish. The PBSItemCollection RelDef is defined to allow the objects at either end of the relationship to be published by other applications; therefore, the objects will not necessarily appear in the same container as the published data. (See Figure 4.)

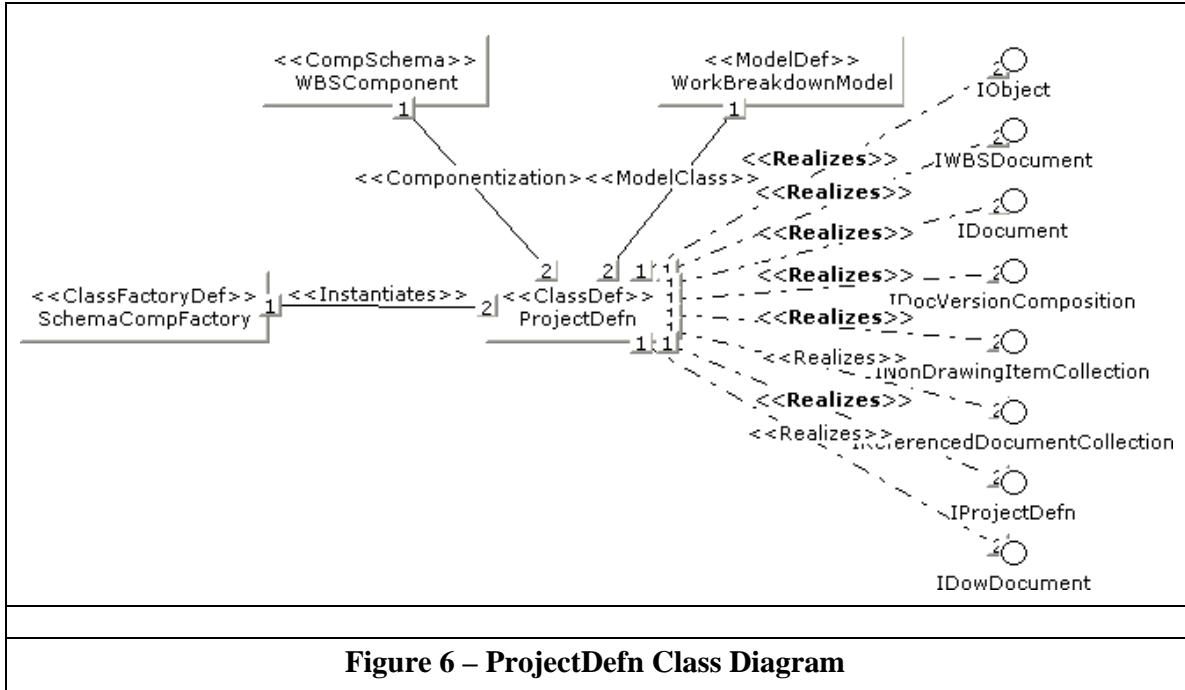


A.2 WBS Component Schema

WBS stands for Work Breakdown Structure, and like PBS, the objects are created and published in SPF and retrieved by the tools. However, there are actually several WBS documents published by SPF.



The ProjectList document is published by an administrator using the **Publish Project List** RHM method on the Plant. This document is also automatically published whenever the project state is changed using the **Project Manager** method. It is very important to realize that *there is one project list published for each plant in the as-built configuration*. This document contains the Plant object and all Projects created within that plant. Some tools, such as SmartPlant 3D, prefer to create the projects in batch mode so they will retrieve the ProjectList and get all projects at once.



An Administrator publishes the ProjectDefn document using a RHM method on Project displayed as **Publish Project Definition**. It is also automatically published whenever the project state is changed using the **Project Manager** method. The project definition document publishes the same structure (Plant->Project) as the project list, but does so for *one project only*. It is also *published in the as-built configuration*. This document is retrieved by tools like SmartPlant Instrumentation or SmartPlant P&ID in order to create projects.

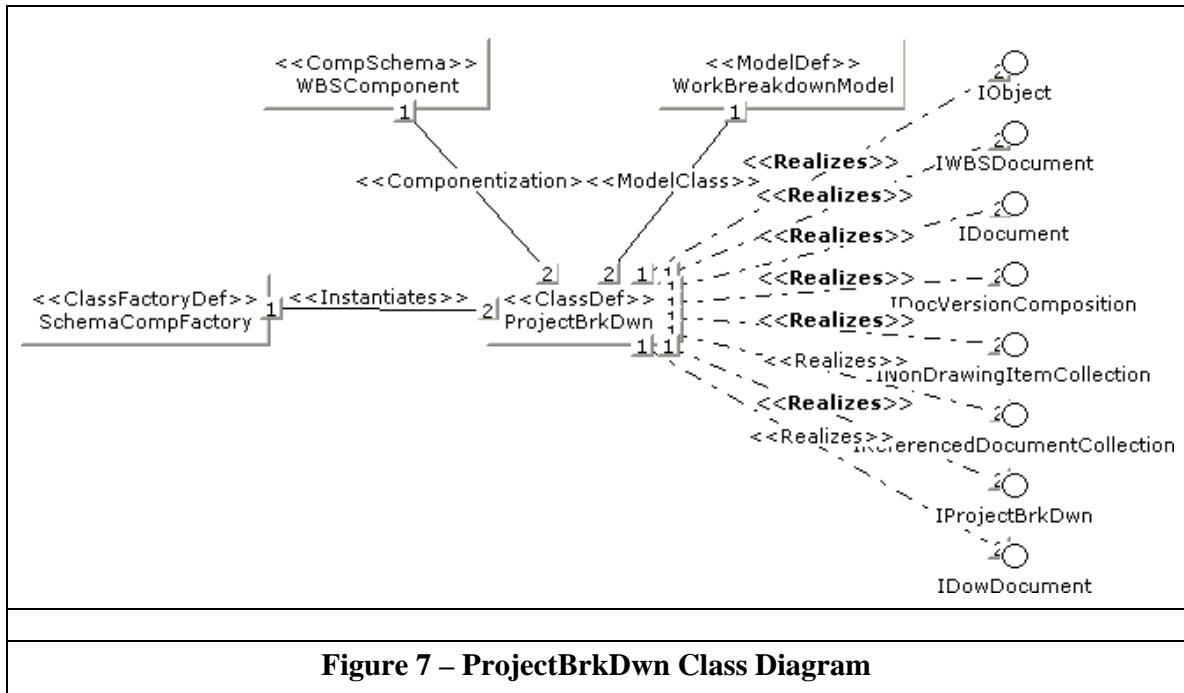


Figure 7 – ProjectBrkDwn Class Diagram

An Administrator publishes the Project Breakdown using the RHM on the Project displayed as **Publish Project Breakdown**. It is also automatically published whenever the project state is changed using the **Project Manager** method. The project breakdown object includes the full WBS structure (Plant->Project->Contract) for one project only and is published in the project configuration. Tools must connect to a project to retrieve this document.

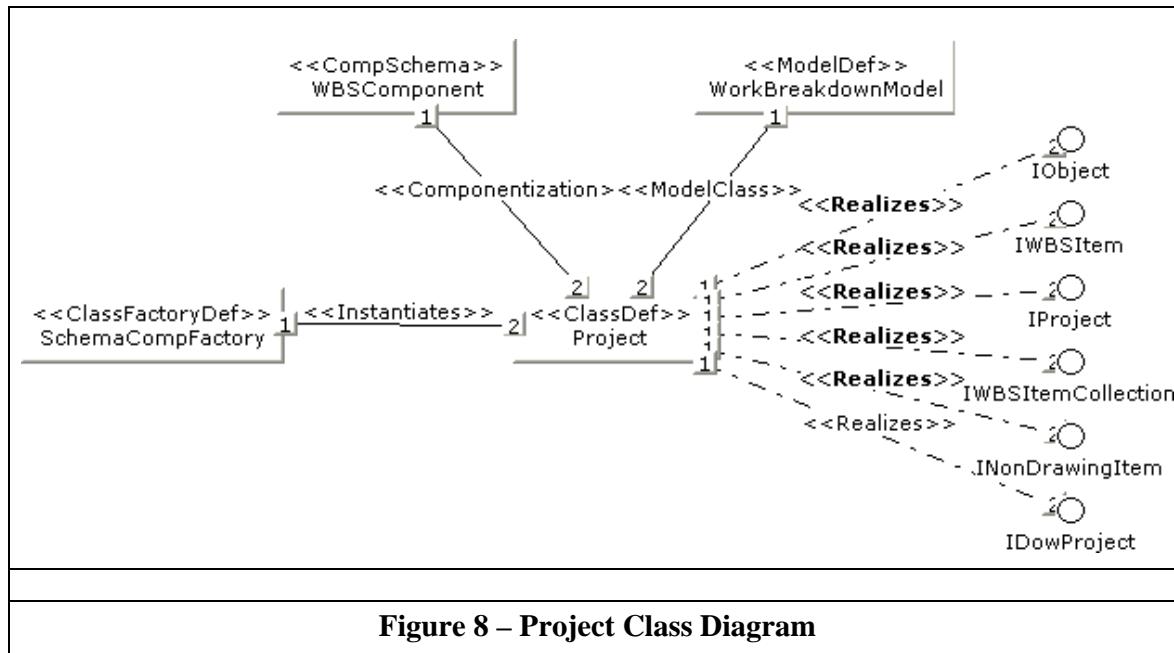
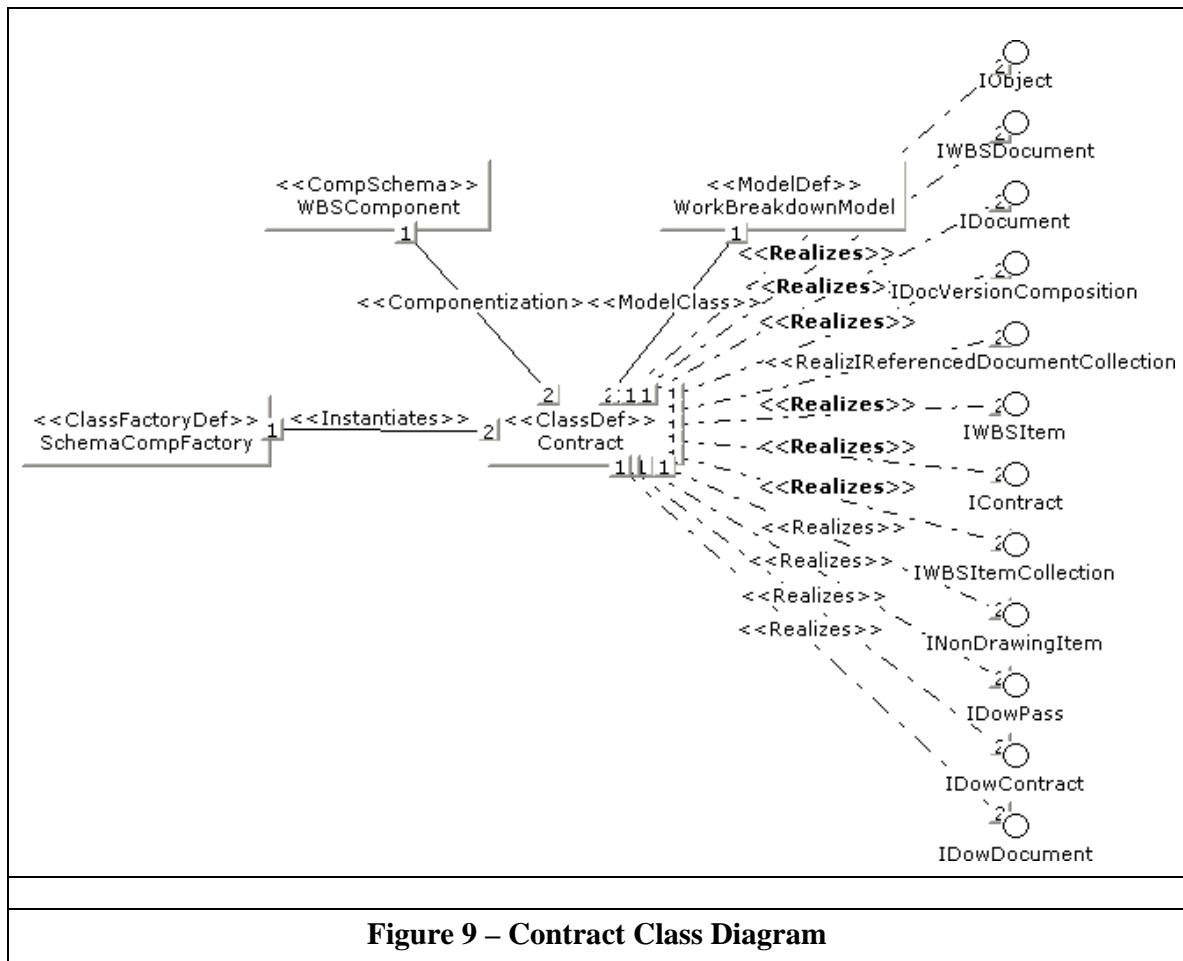


Figure 8 – Project Class Diagram

Figure 8 shows the class diagram for Project. Notice that it realizes IWBSItemCollection and IWBSItem. This structure is identical to the PBS structure.



Contract is different from the other WBS items in that it is actually a document. The contracts are created using SPF's document management functionality. Files may be attached, checked in, checked out, signed off, and revised like any other document. At the same time, the contract is a WBS item.

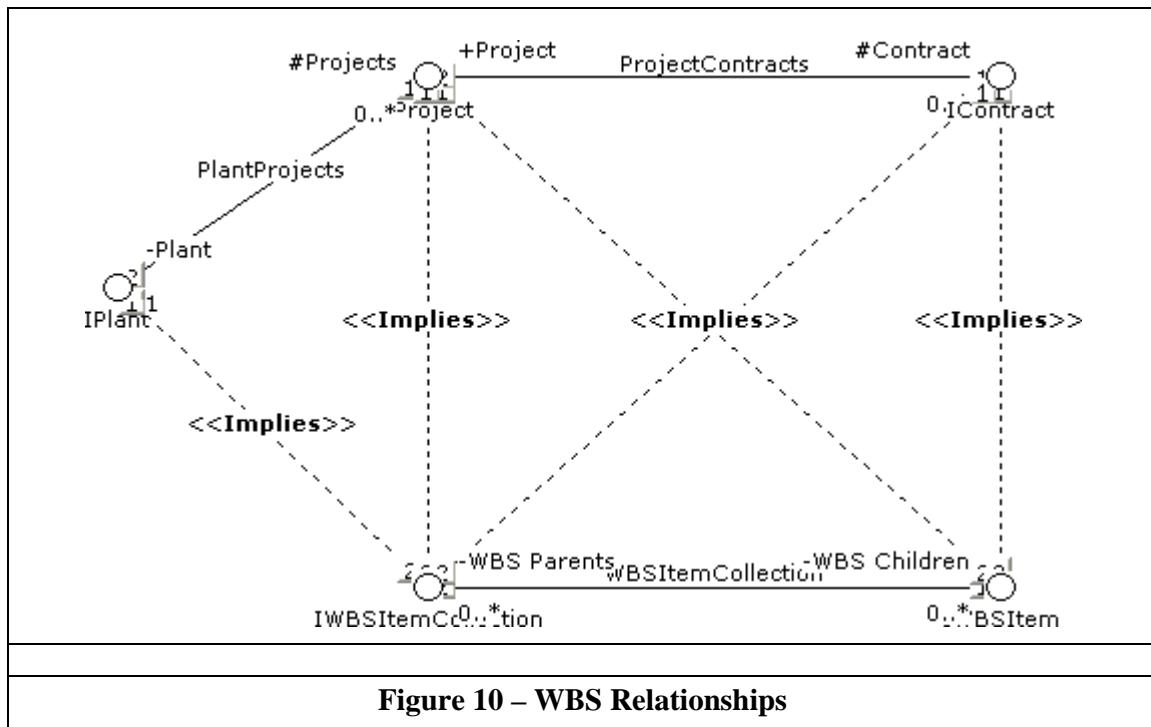


Figure 10 shows the relationships involved in the WBS, which has a similar structure to PBS. Plant, Project, and Contract are all related through the generic WBSItemCollection RelDef, but there are also RelDefs that are specializations, including PlantProjects between IPlant and IPProject and ProjectContracts between IPProject and IContract.

A.3 Generic Document Component Schema

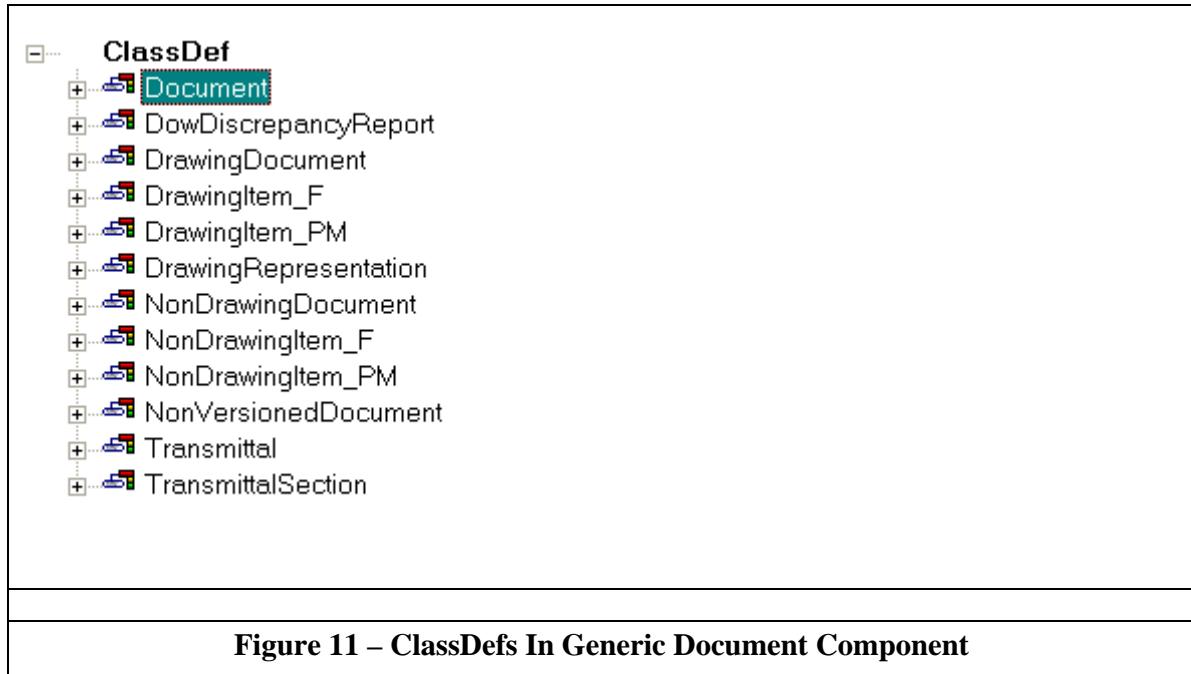


Figure 11 shows all the ClassDefs contained in the generic document component schema. Almost everything in the generic document component schema will soon change drastically. The only ClassDef of consequence right now is Document.

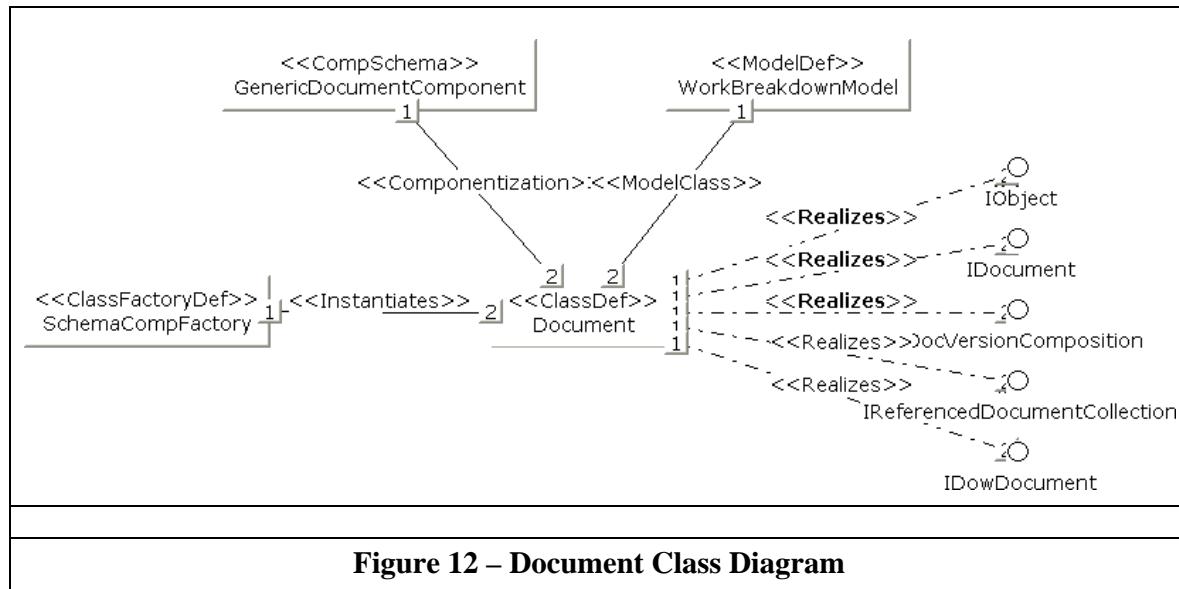


Figure 12 shows the class diagram for Document. As the name implies, the Document class represents a very generic document. It serves two very useful purposes right now. First, most documents that are created interactively in SPF are of the Document ClassDef. The create methods delivered with the EF Model create documents from this ClassDef.

The second use of the Document ClassDef is during the publish process. When the SmartPlant Client queries the tool adapter for the documents to be published, it passes in a schema container for the results. So that the adapter does not need to load the entire component schema, the documents are created using the Document ClassDef. Later, the SmartPlant Client polls the adapter to publish the data for each document one-by-one. Before the Client polls the adapter for the data, it asks the adapter the component schema and ClassDef for each. Each document begins its existence as ClassDef Document and is converted to the specific ClassDefs published by each of the tools.

A.4 Document Versioning Component Schema

The document versioning component schema is all that is required to define the data that appears in the metadata container for publish and retrieve.

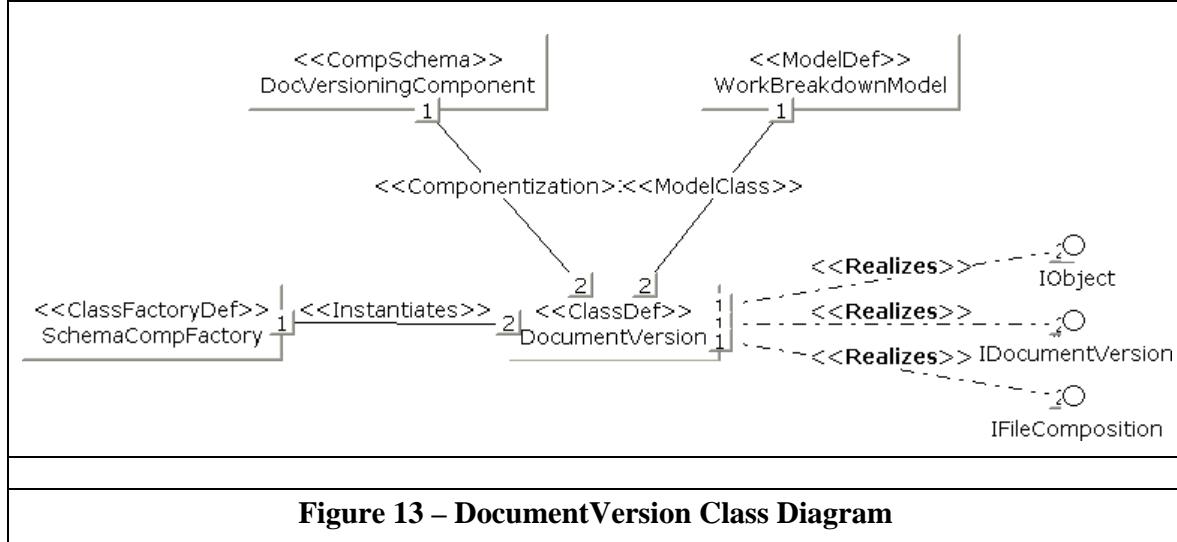


Figure 13 shows the class diagram for DocumentVersion. The IDocumentVersion InterfaceDef exposes the RevisionScheme, Revision, and Version for the document version. IDocumentVersion has a relationship defined to IDocVersionComposition, which is realized by all versioned documents. IFileComposition defines a relationship to IFile.

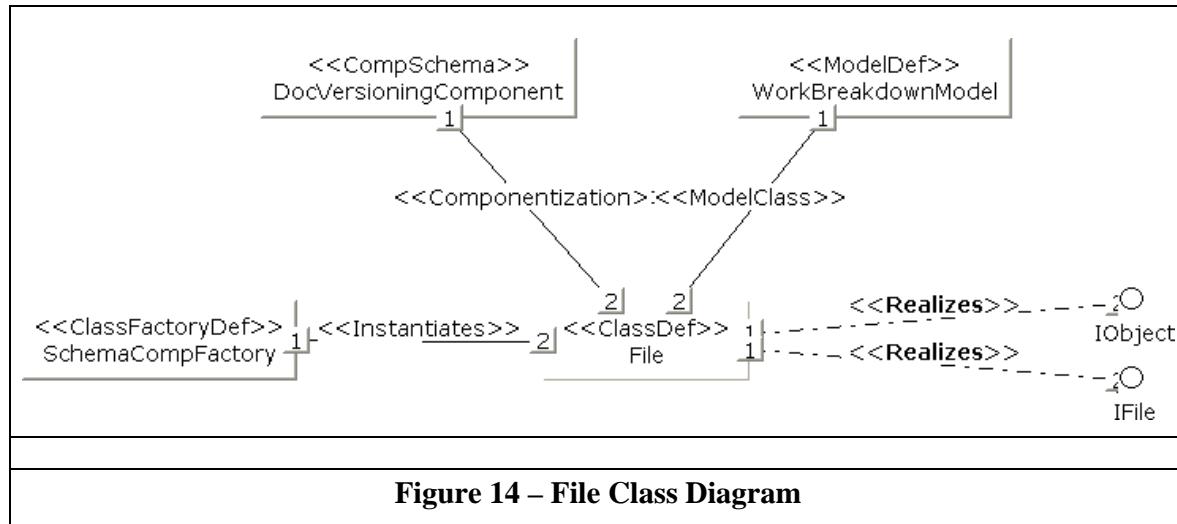


Figure 14 shows the class diagram for File, which is one of the simplest in the SmartPlant Schema. The IFile InterfaceDef exposes a path and file type properties.

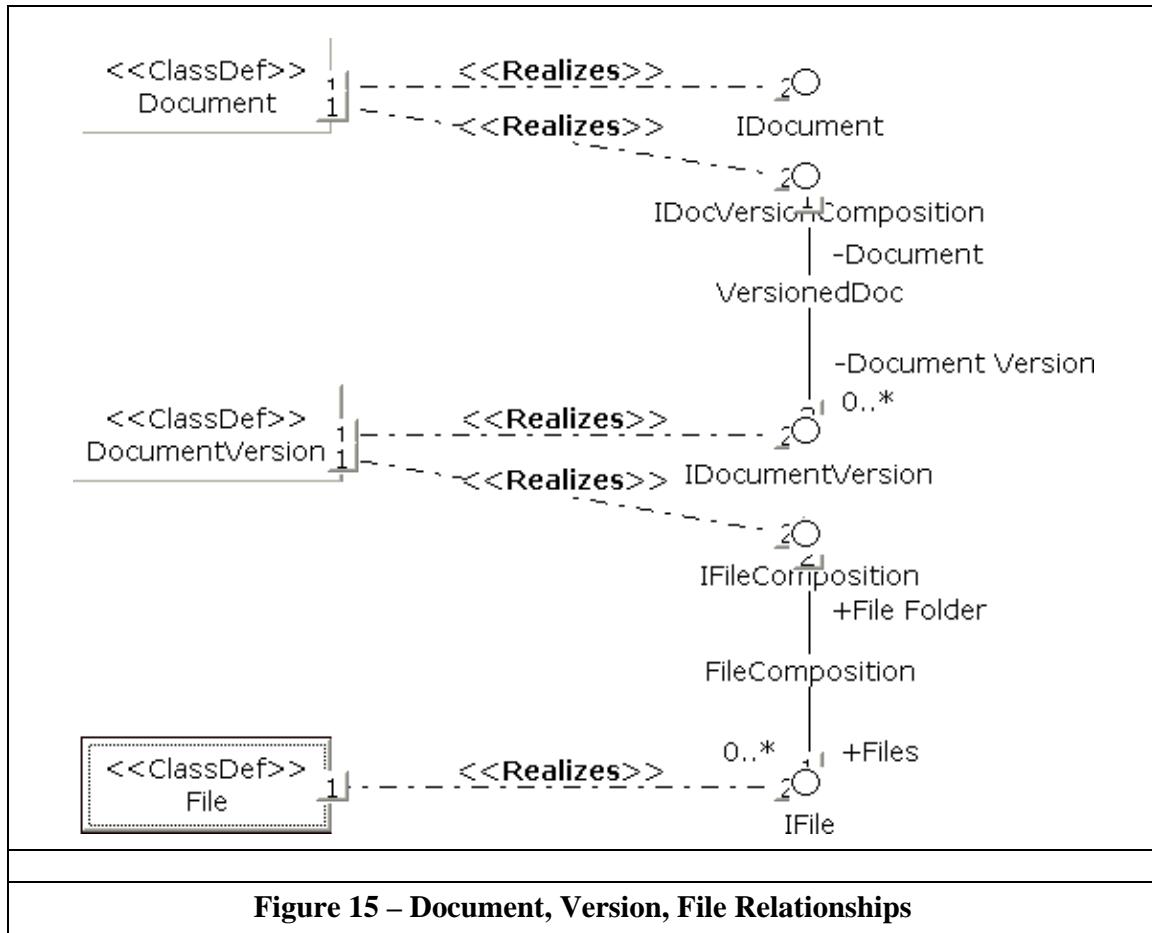


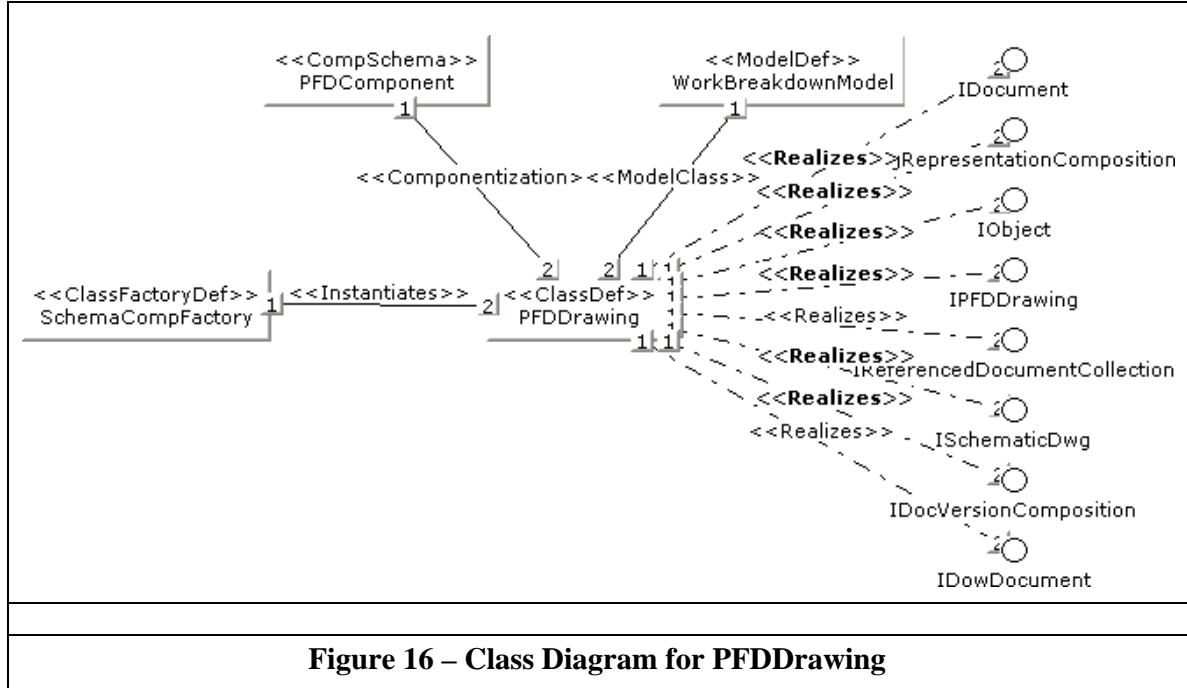
Figure 15 – Document, Version, File Relationships

Figure 15 shows another view of the document, version, and file relationships. Notice that document and document version realize **IDocVersionComposition** and **IFileComposition** respectively. Each of these compositions defines relationships to the component versions and files.

The document versioning component defines four more ClassDefs: **RevisionScheme**, **Workflow**, **Ref2DAttachment**, **Ref3DAttachment**. These ClassDefs are used in communication between the SmartPlant Client and Server.

A.5 PFD Component Schema

The PFD component schema defines the ClassDefs that are published by Zygad with the Process Flow Diagram. The drawing is published as ClassDef PFDDrawing.



The PFD is published with a RAD-based view file. The file extension is .zyq. The items graphically represented on the drawing are published with a relationship to the drawing through the IDwgRepresentationComposition InterfaceDef. This allows normal SPF Graphic Navigation to be supported for PFDs.

Along with the drawing itself, two top-level objects are published: Equipment and Streams.

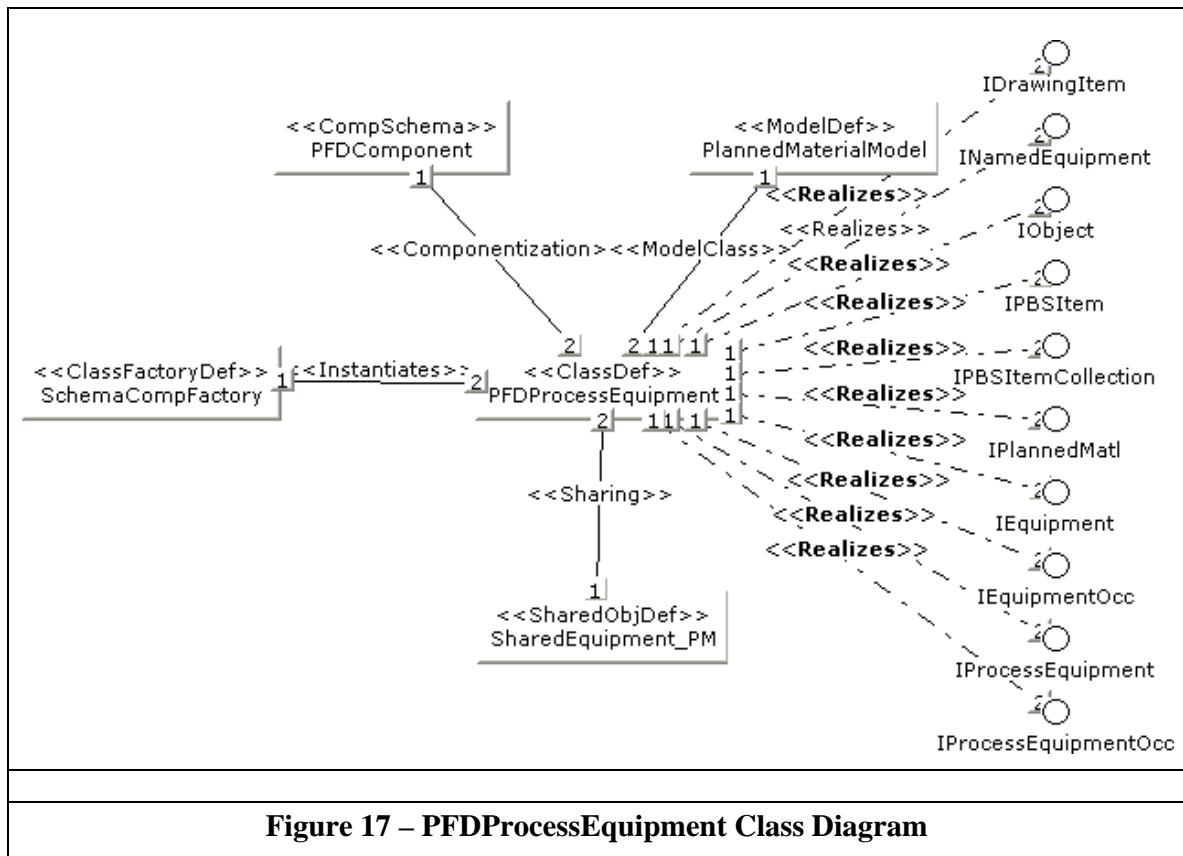
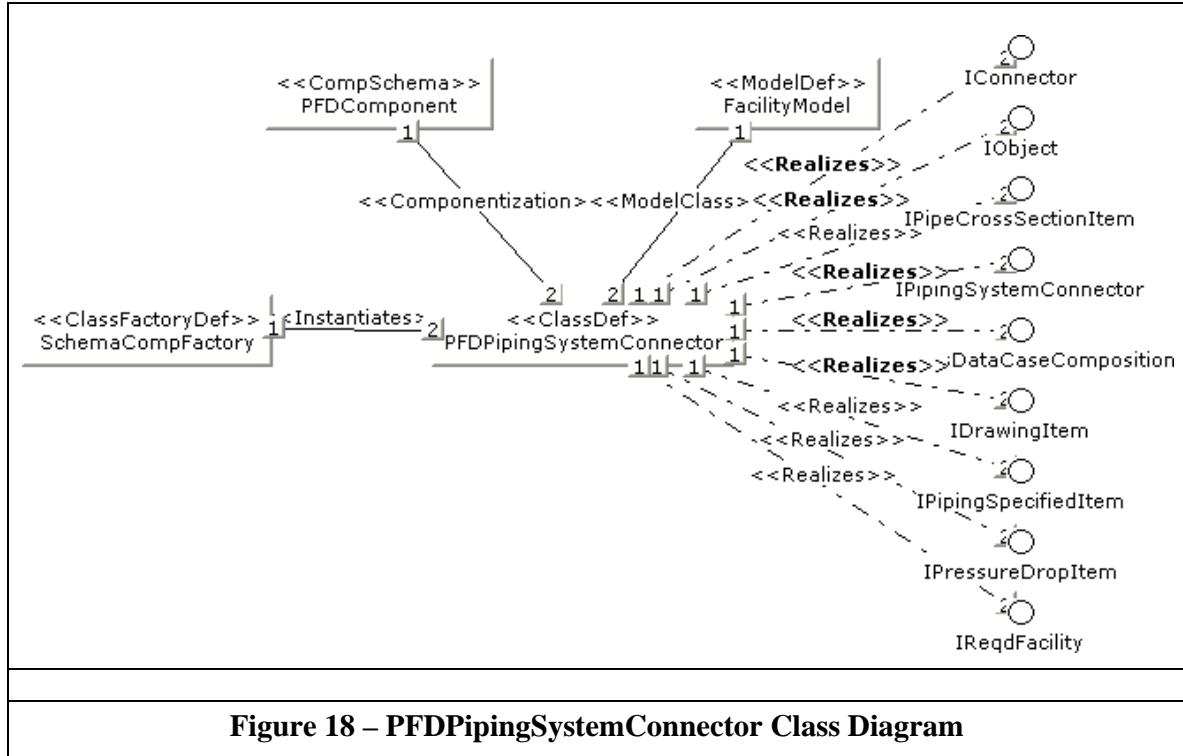


Figure 17 – PFDProcessEquipment Class Diagram

Equipment is published as PFDProcessEquipment. Each object is graphically related to the drawing through the IDrawingItem InterfaceDef. Each object is mapped into the PBS structure through the IPBSItem InterfaceDef. No other connectivity is published for equipment on the PFD. No process data is published for equipment, either.



Streams are published as **PFDPipingSystemConnectors**. Each object is graphically related to the drawing through the **IDrawingItem** InterfaceDef. Process data is published with each stream (see Figure 19).

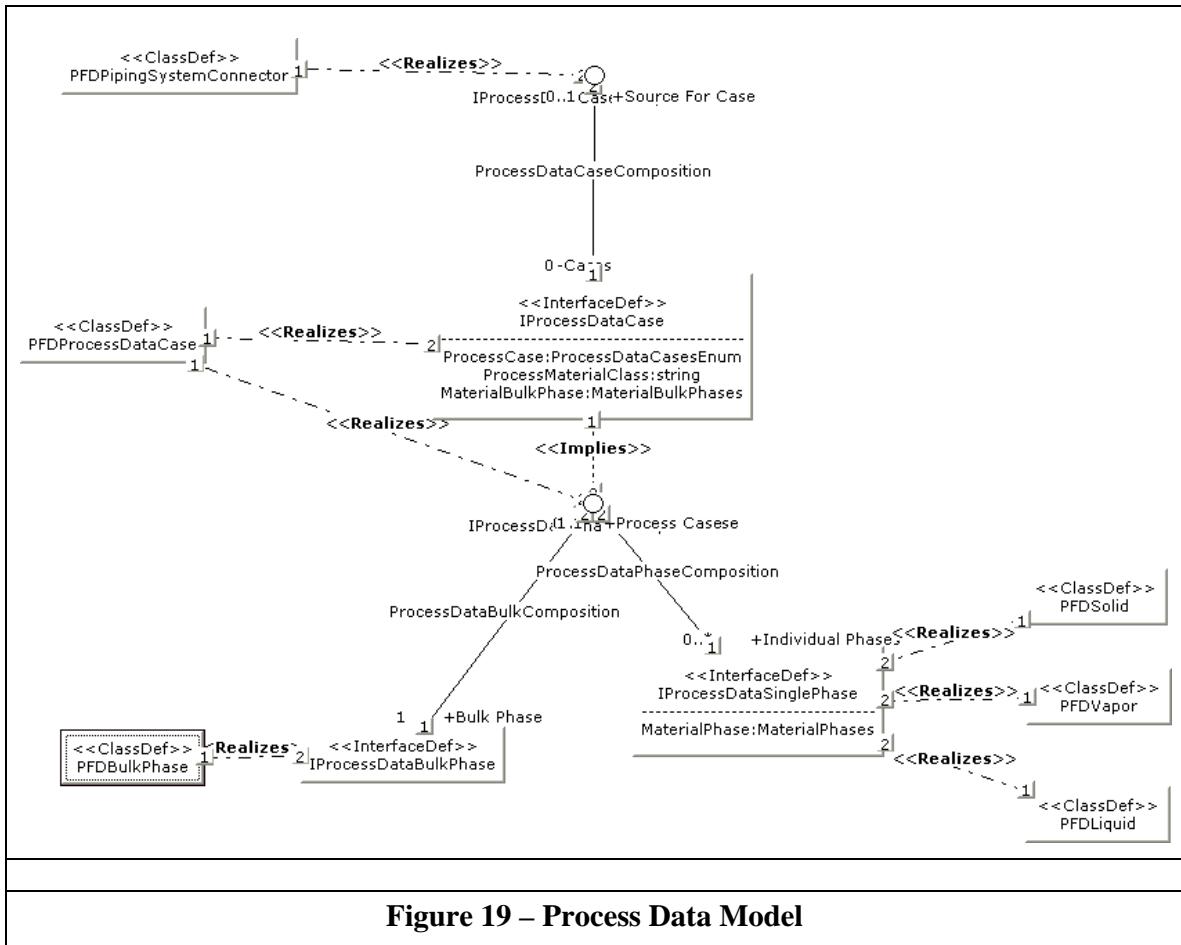


Figure 19 – Process Data Model

The PFDPipingSystemConnector (stream) realizes the IProcessDataCaseComposition InterfaceDef, which defines a relationship named ProcessDataCaseComposition to each of the cases. Three fixed cases are published:

- Min
- Max
- Normal

The ClassDef for the cases is PFDProcessDataCase, and the primary InterfaceDef is IProcessDataCase. (See Table 1 for properties.) Each of the data cases also realizes the IProcessDataPhaseComposition InterfaceDef. The ProcessDataPhaseComposition InterfaceDef has a RelDef named ProcessDataPhaseComposition to the IProcessDataSingPhase InterfaceDef.

Three phases for each case may be published under the PFDSolid, PFDVapor, and PFDLiquid ClassDefs. Combinations of two phases may be published under the PFDBulkPhase ClassDef, which is related to the case by the ProcessDataBulkComposition RelDef. (See Table 2 for properties exposed by the PFDBulkPhase ClassDef. The other phases expose similar properties.)

Table 1 – PFDProcessDataCase Properties

Class	Interface	Property	Type
PFDProcessDataCase	IObject	UID	string128
PFDProcessDataCase	IObject	Name	string128
PFDProcessDataCase	IObject	Description	string
PFDProcessDataCase	IProcessDataCase	ProcessCase	ProcessDataCasesEnum
PFDProcessDataCase	IProcessDataCase	ProcessMaterialClass	string
PFDProcessDataCase	IProcessDataCase	MaterialBulkPhase	MaterialBulkPhases
PFDProcessDataCase	IProcessDataPhaseComposition		

Table 2 – PFDBulkPhase Properties

Class	Interface	Property	Type
PFDBulkPhase	IFluidMixture	BubblePointP	PressureUoM
PFDBulkPhase	IFluidMixture	BubblePointT	TemperatureUoM
PFDBulkPhase	IFluidMixture	DewPointP	PressureUoM
PFDBulkPhase	IFluidMixture	DewPointT	TemperatureUoM
PFDBulkPhase	IFluidMixture	RetrogradeDewPointP	PressureUoM
PFDBulkPhase	IFuel	HighHeatingValue	MassHeatingValueUoM
PFDBulkPhase	IFuel	HydrocarbonRatio	double
PFDBulkPhase	IFuel	LowHeatingValue	MassHeatingValueUoM
PFDBulkPhase	IFuel	SodiumContent	double
PFDBulkPhase	IFuel	SulphurContent	double
PFDBulkPhase	IFuel	VanadiumContent	double
PFDBulkPhase	IFuel	PayoutPeriod	TimeUoM
PFDBulkPhase	IFuel	CostForPayout	double
PFDBulkPhase	IMovingObject	Velocity	VelocityUoM
PFDBulkPhase	IMovingPhase	MassFlowRate	MassFlowUoM
PFDBulkPhase	IMovingPhase	MoleFlowRate	MolarFlowUoM
PFDBulkPhase	IObject	UID	string128
PFDBulkPhase	IObject	Name	string128

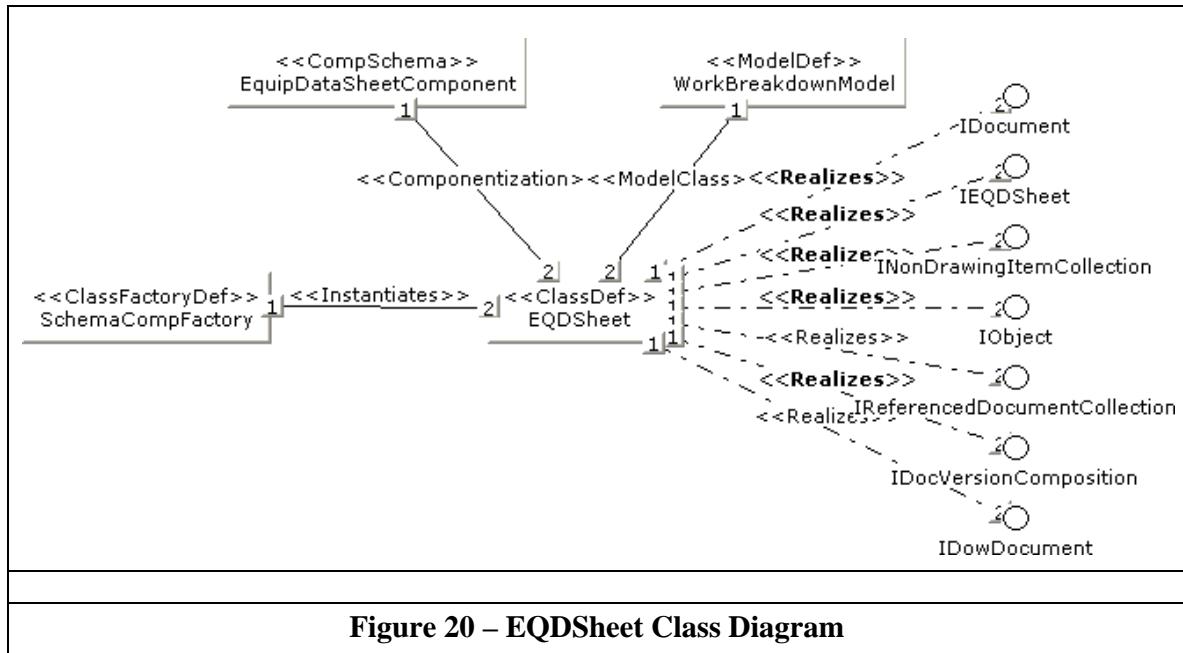
Class	Interface	Property	Type
PFDBulkPhase	IObject	Description	string
PFDBulkPhase	IProcessDataBulkPhase		
PFDBulkPhase	IProcessDataPhase	CriticalPressure	PressureUoM
PFDBulkPhase	IProcessDataPhase	CriticalTemperature	TemperatureUoM
PFDBulkPhase	IProcessDataPhase	CriticalVolume	VolumeUoM
PFDBulkPhase	IProcessDataPhase	DielectricConstant	double
PFDBulkPhase	IProcessDataPhase	MolarDensity	DensityUoM
PFDBulkPhase	IProcessDataPhase	MolarSpecificGravity	double
PFDBulkPhase	IProcessDataPhase	MolarSpecificVolume	SpecificVolumeUoM
PFDBulkPhase	IProcessDataPhase	Prandtl	double
PFDBulkPhase	IProcessDataPhase	PhaseTemperature	TemperatureUoM
PFDBulkPhase	IProcessDataPhase	ThermalConductivityValue	ThermalConductivityUoM
PFDBulkPhase	IProcessDataPhase	Abrasiveness	double
PFDBulkPhase	IProcessDataPhase	MolecularWeight	double
PFDBulkPhase	IProcessDataPhase	MassDensity	DensityUoM
PFDBulkPhase	IProcessDataPhase	MassSpecificGravity	double
PFDBulkPhase	IProcessDataPhase	MassSpecificVolume	SpecificVolumeUoM
PFDBulkPhase	IVaporLiquid	HeatOfVaporization	double
PFDBulkPhase	IVaporLiquid	VaporQuality	double
PFDBulkPhase	IHazardousMaterial	Corrosive	YesNo
PFDBulkPhase	IHazardousMaterial	Erosive	YesNo
PFDBulkPhase	IHazardousMaterial	Flammable	YesNo
PFDBulkPhase	IHazardousMaterial	IDLH	YesNo
PFDBulkPhase	IHazardousMaterial	LethalService	YesNo
PFDBulkPhase	IHazardousMaterial	LTEL	string
PFDBulkPhase	IHazardousMaterial	MAC	string
PFDBulkPhase	IHazardousMaterial	Oxidizing	YesNo
PFDBulkPhase	IHazardousMaterial	ToxicityClassification	ToxicityClassifications
PFDBulkPhase	IHazardousMaterial	Toxic	YesNo

Class	Interface	Property	Type
PFDBulkPhase	IFluid	Reynolds	double
PFDBulkPhase	IFluid	VolumetricFlowRate	VolumetricFlowUoM
PFDBulkPhase	IFluid	ContainsParticles	YesNo
PFDBulkPhase	IFluid	Cp	HeatCapacityUoM
PFDBulkPhase	IFluid	CpCvRatio	double
PFDBulkPhase	IFluid	Cv	HeatCapacityUoM
PFDBulkPhase	IFluid	SuspendedSolids	double
PFDBulkPhase	IFluid	Viscosity	AbsoluteViscosityUoM
PFDBulkPhase	IFluid	Diffusion	double
PFDBulkPhase	IFluid	IndexOfRefraction	double
PFDBulkPhase	IFluid	Permittivity	double
PFDBulkPhase	IFluid	FluidPressure	PressureUoM
PFDBulkPhase	IFluid	FluidCompressibility	double

Two other ClassDefs are defined in the PFDCOMPONENT schema: PFDNozzlePort and PFDPipingSystemPort. These are not currently used.

A.6 EQD Component Schema

The EQD component schema (EquipDatasheetComponent) defines the ClassDefs published by the equipment datasheet functionality in Zygad. Equipment datasheets may be published for each equipment object in Zygad as well as each stream object. The document is published as ClassDef EQDSheet.



The EQDSheet is a non-drawing document, so the items are related to the document through the `INonDrawingItemCollection` InterfaceDef. The EQD is published with an Excel view file.

Unlike the PFDProcessEquipment class, each type of equipment has an EQD* ClassDef defined. These classes are very detailed and property-rich. Figure 21 shows the Class Diagram for EQDAgitator, which is a typical ClassDef for this component schema.

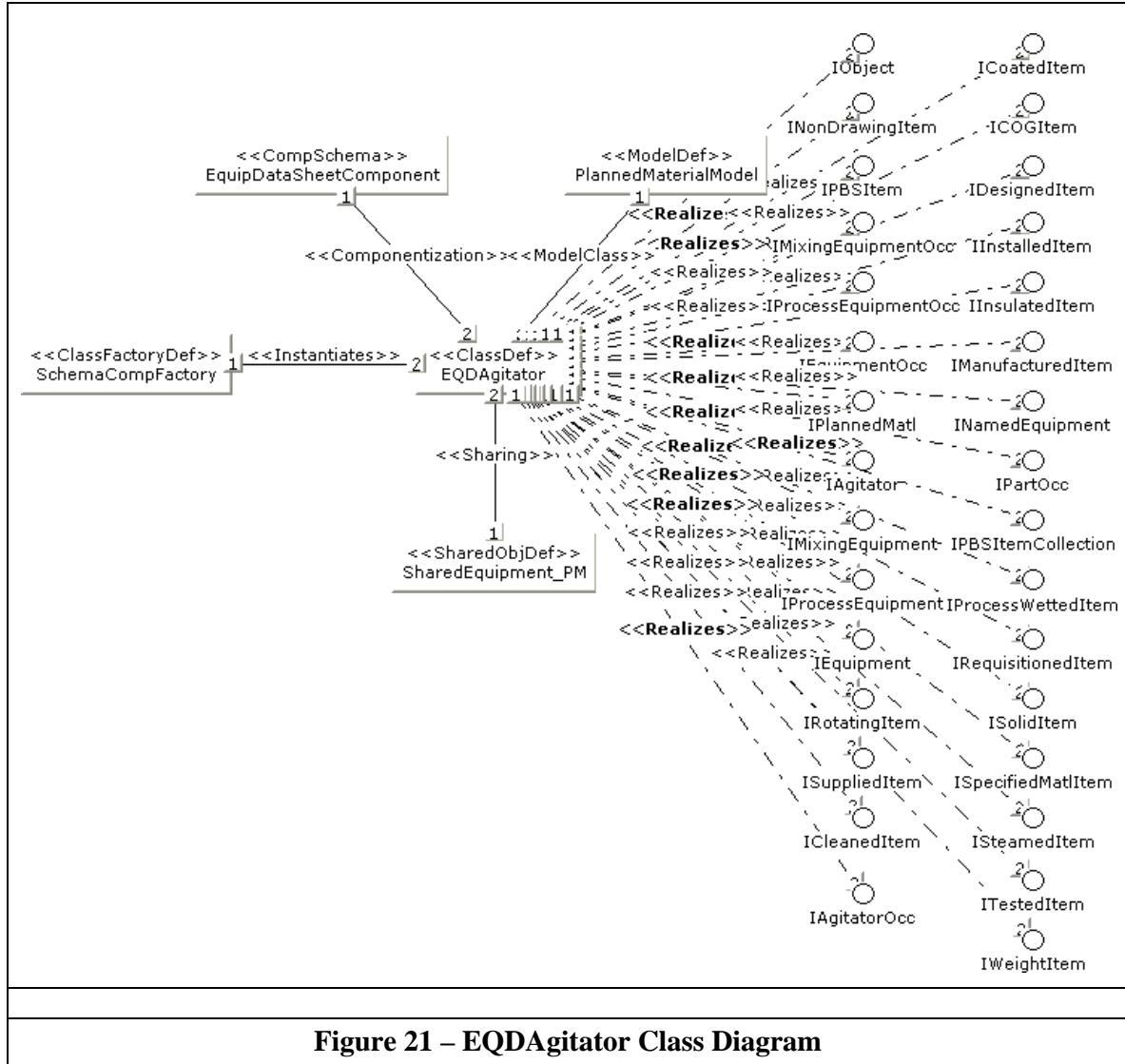


Figure 21 – EQDAgitator Class Diagram

The only relationships published with the EQD objects are the NonDrawingItemCollection (through **INonDrawingItem**) and the PFDItemCollection (through **IPBSItem**).

The stream may also be published on a datasheet as the **EQDPipingSystemConnector** ClassDef.

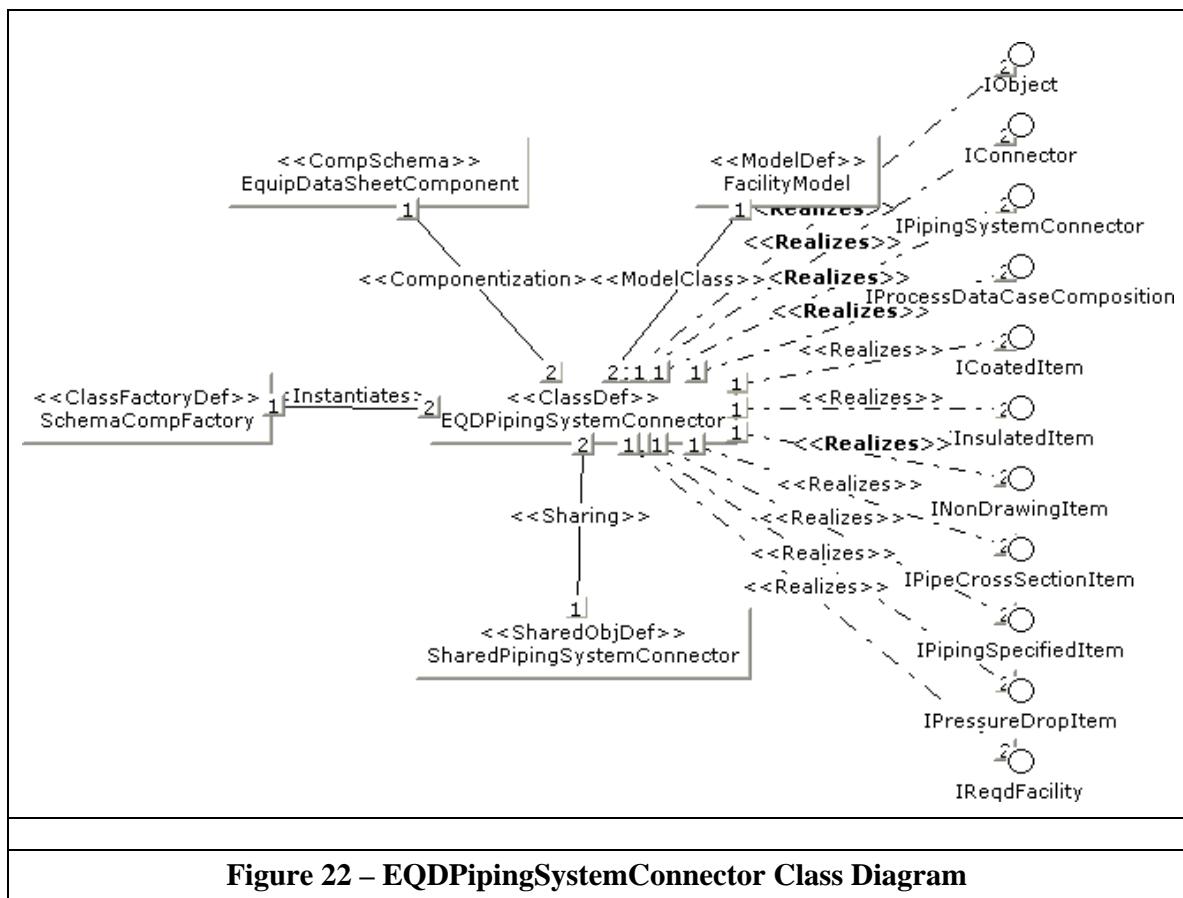
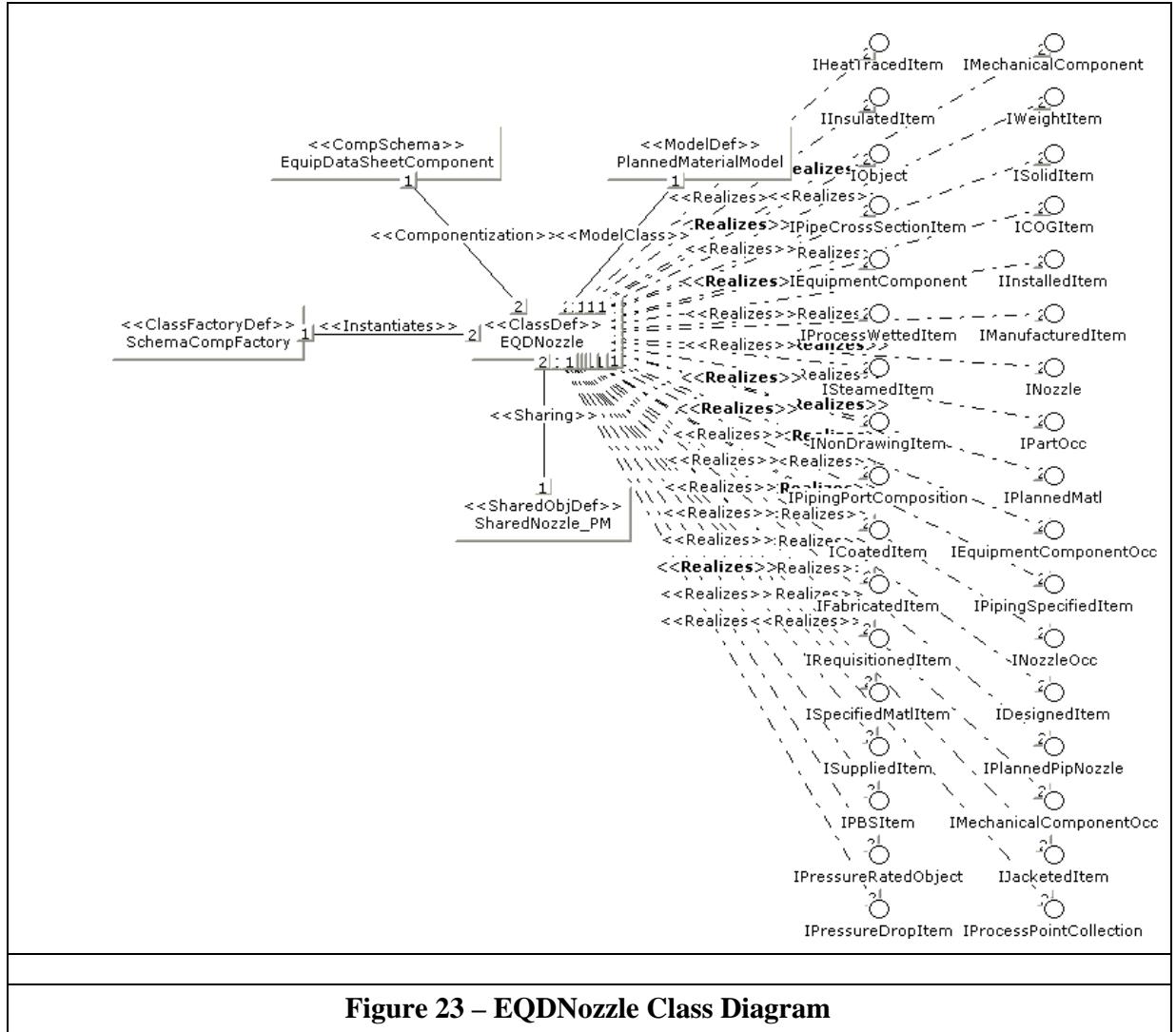


Figure 22 – EQDPipingSystemConnector Class Diagram

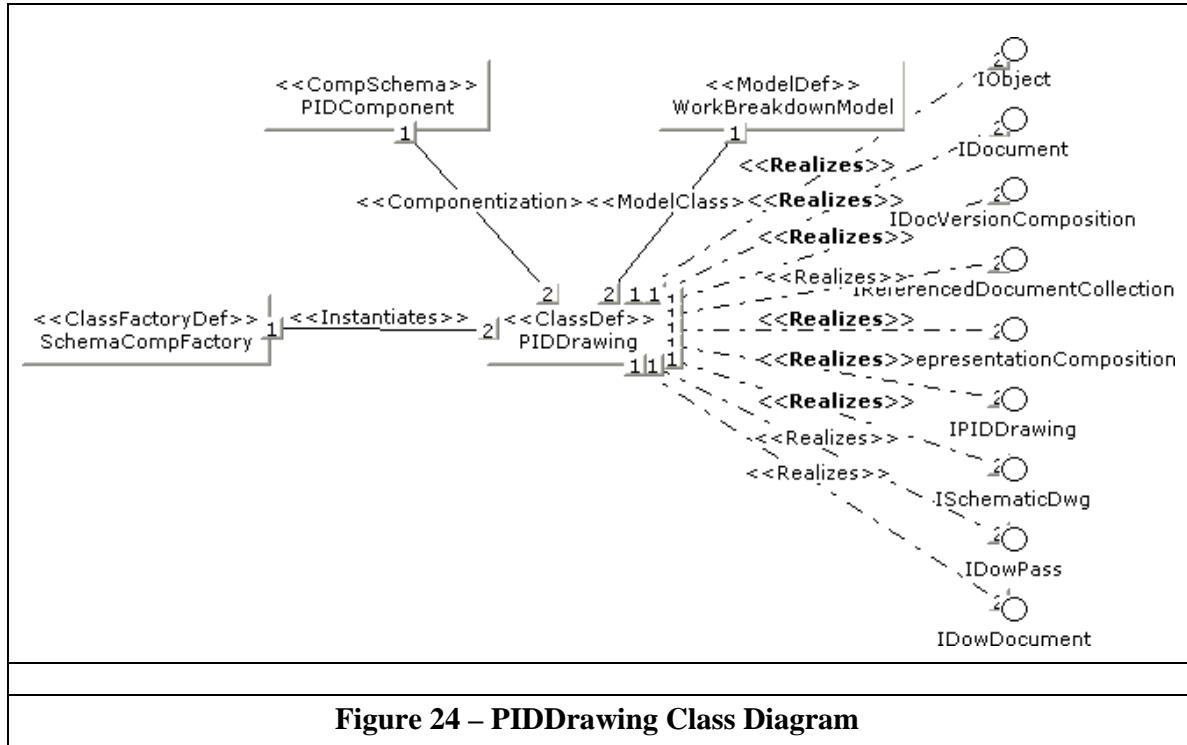
Process data is published on the datasheet in a structure similar to the data published with PFDPipingSystemConnector. The only difference is the names of the ClassDefs: EQDProcessDataCase, EQDProcessDataBulkPhase, EQDVapor, EQDLiquid, and EQDSolid.



The only equipment components being published by the EQD component today are the equipment nozzles (ClassDef EQDNozzle).

A.7 PID Component Schema

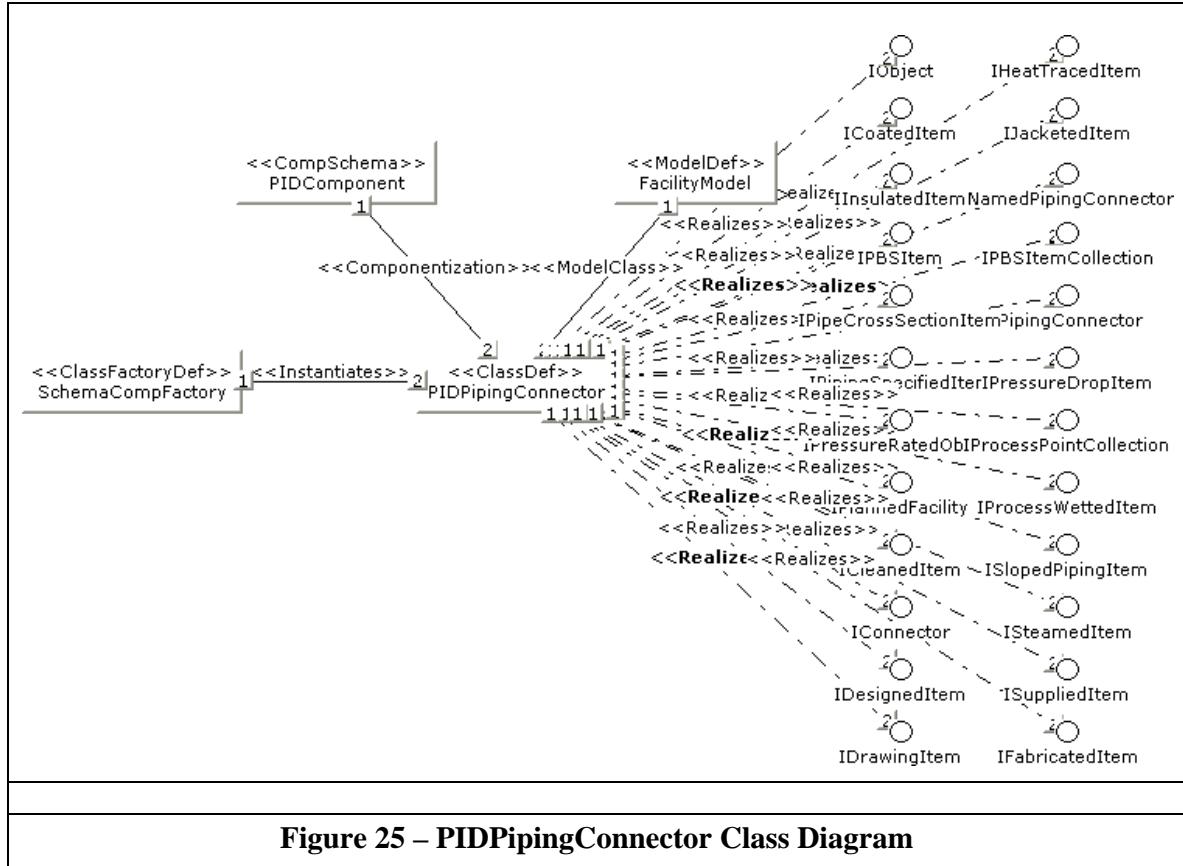
The main document published by SmartPlant P&ID is a P&ID. That document is published using the PIDDrawing ClassDef.



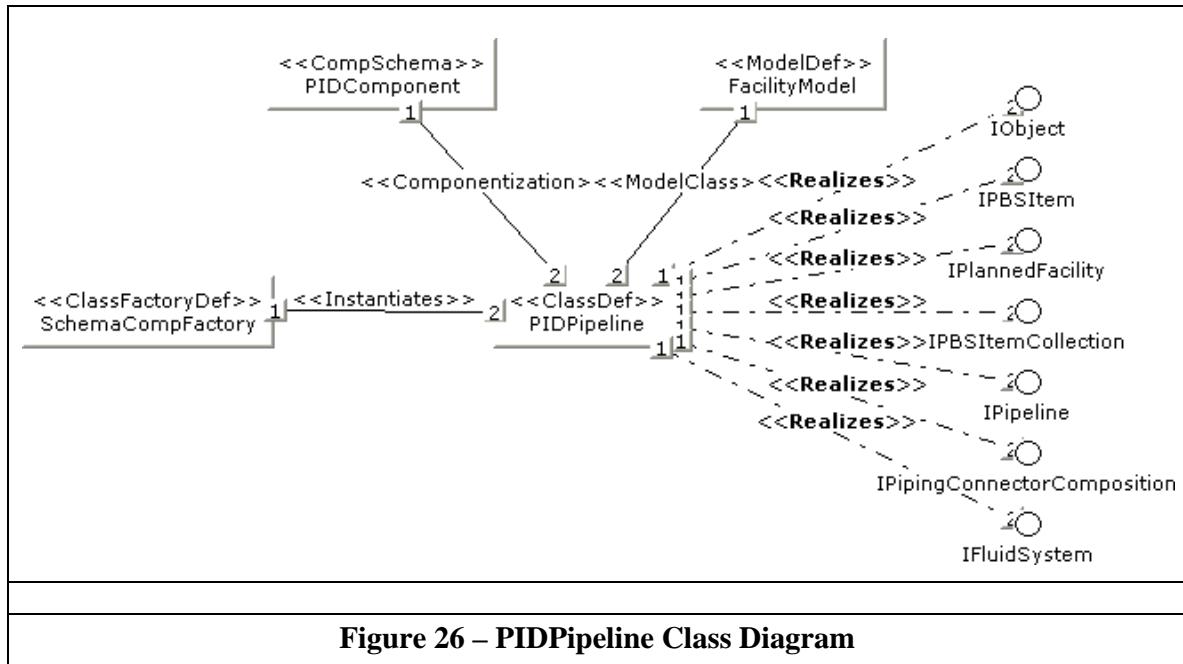
The P&ID is published with a RAD-based view file. The file extension is .pid. The items graphically represented on the drawing are published with a relationship to the drawing through the IDwgRepresentationComposition InterfaceDef, allowing normal SPF Graphic Navigation to be supported. In fact, almost all objects published by SmartPlant P&ID realize the IDrawingItem InterfaceDef. Likewise, almost all objects realize the IPBSItem InterfaceDef so they are related into the PBS structure.

Along with the drawing, numerous top-level objects are published, along with a host of supporting objects.

Streams in Zygad (PFDPipingSystemConnector) are translated during the retrieve into PIDPipingConnectors. These are pipe runs in SmartPlant P&ID and are sometimes called pipe segments.



SmartPlant P&ID does not maintain a Line object internally. It has nothing that relates one-to-one with the stream in Zygad. However, during publish SmartPlant P&ID generates a PIDPipeline object to group the piping connectors.



Notice that the **PIDPipeline** ClassDef does not realize the **IDrawingItem** InterfaceDef. There is no graphic object on the P&I drawing to warrant the drawing representation.

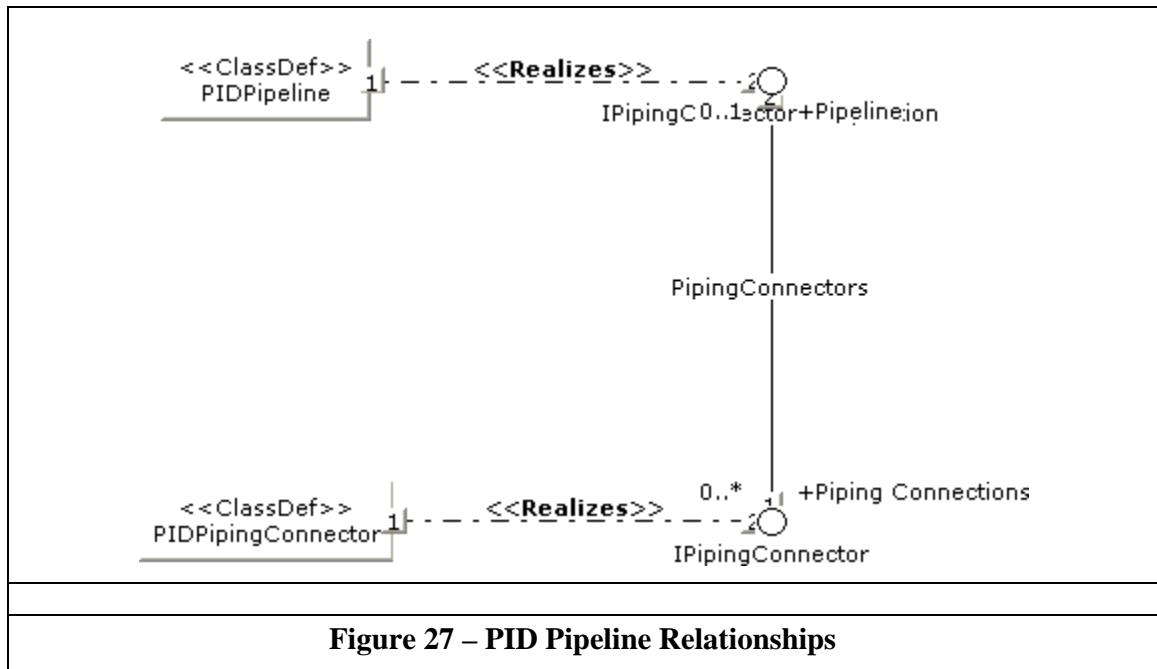


Figure 27 shows the relationship between **PIDPipeline** and **PIDPipingConnector**.

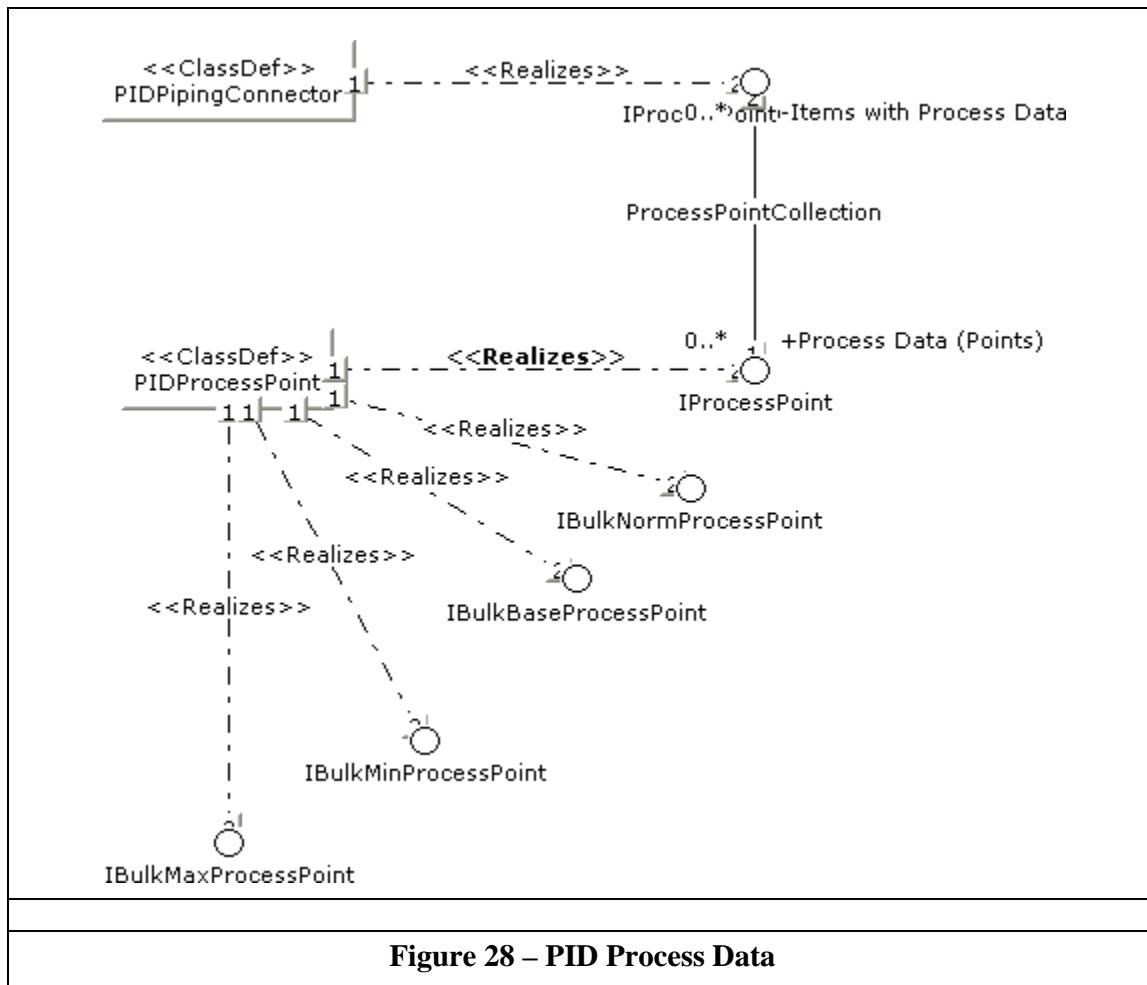
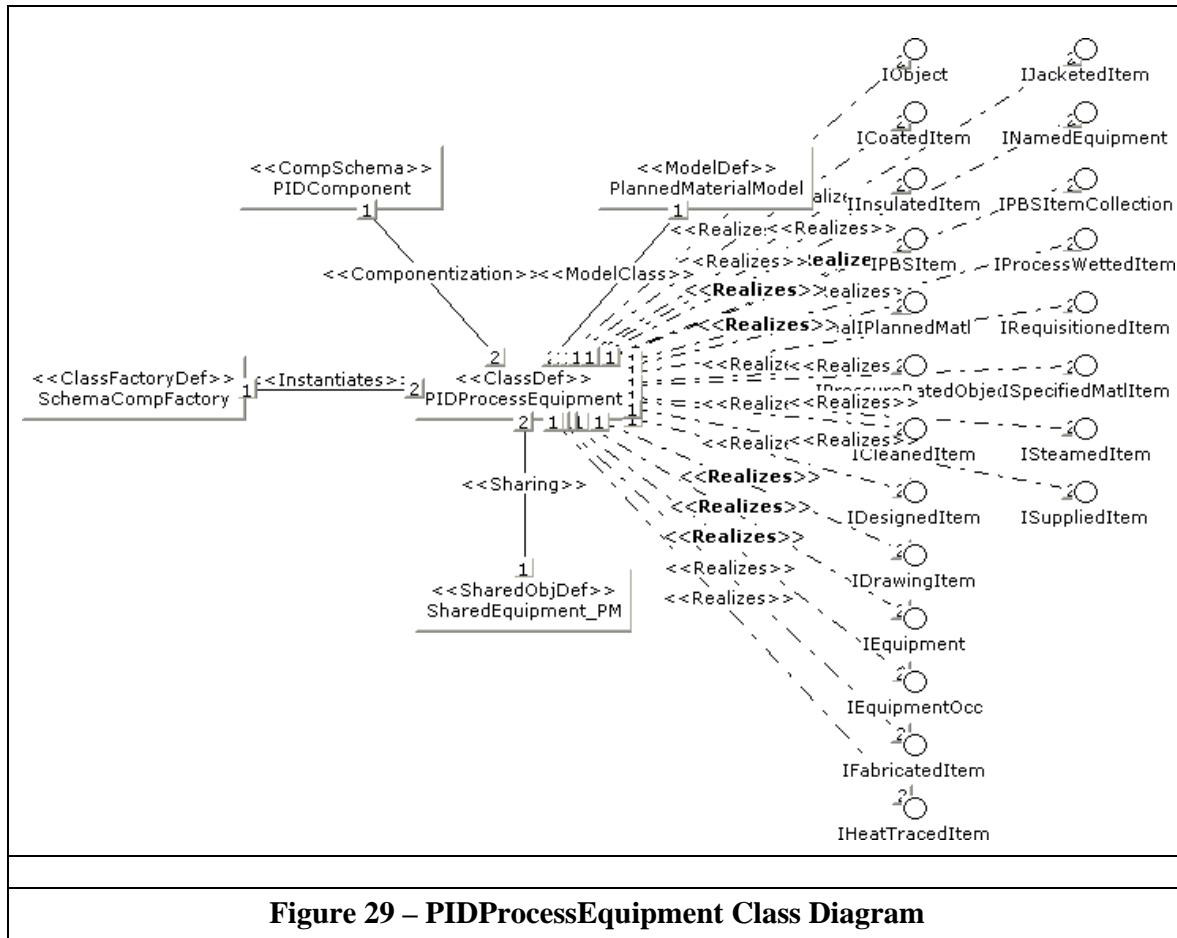


Figure 28 – PID Process Data

SmartPlant P&ID publishes process data on the piping connectors. Figure 28 shows the relationship between the piping connectors and the process information.

As you can see from this diagram, the process data is published as objects of ClassDef **PIDProcessPoint**. These are related to the piping connectors via the **ProcessPointCollection RelDef**.



All equipment is published as PIDProcessEquipment. Just like the PFDProcessEquipment, these objects are types with the properties EqType0 through EqType6 on the IEquipment InterfaceDef. The values for these properties are scoped by the Equipment EnumListType.

As you can see in Figure 29, the PIDProcessEquipment ClassDef realizes many InterfaceDefs. Each object is related to the drawing through the DrawingItemCollection relationship defined on the IDrawingItem InterfaceDef. This supports SPF Graphical Navigation. Each object is also related into the PBS structure through the IPBSItem InterfaceDef.

Equipment components such as pump drivers or vessel components are published as `PIDProcessEquipmentComponent`. This ClassDef realizes many of the same InterfaceDefs as `PIDProcessEquipment`.

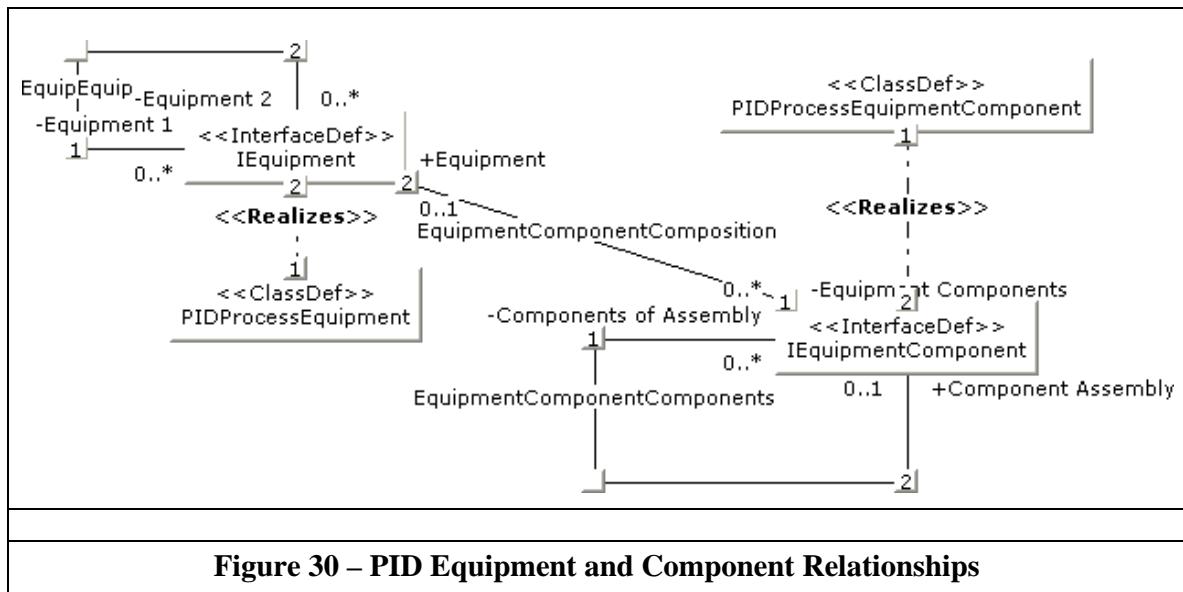


Figure 30 – PID Equipment and Component Relationships

In figure 30, you can see the RelDefs for equipment and components. Notice that the IEquipment InterfaceDef has a RelDef to itself named EquipEquip. This is the type of relationship you would find between a pump and a driver, for instance. Also, notice that the IEquipmentComponent InterfaceDef has a RelDef to itself named EquipmentComponentComponents. This RelDef can be used to create assemblies of equipment components. The RelDef between equipment and components is named EquipmentComponentComposition.

SmartPlant P&ID treats nozzles as a special form of equipment component.

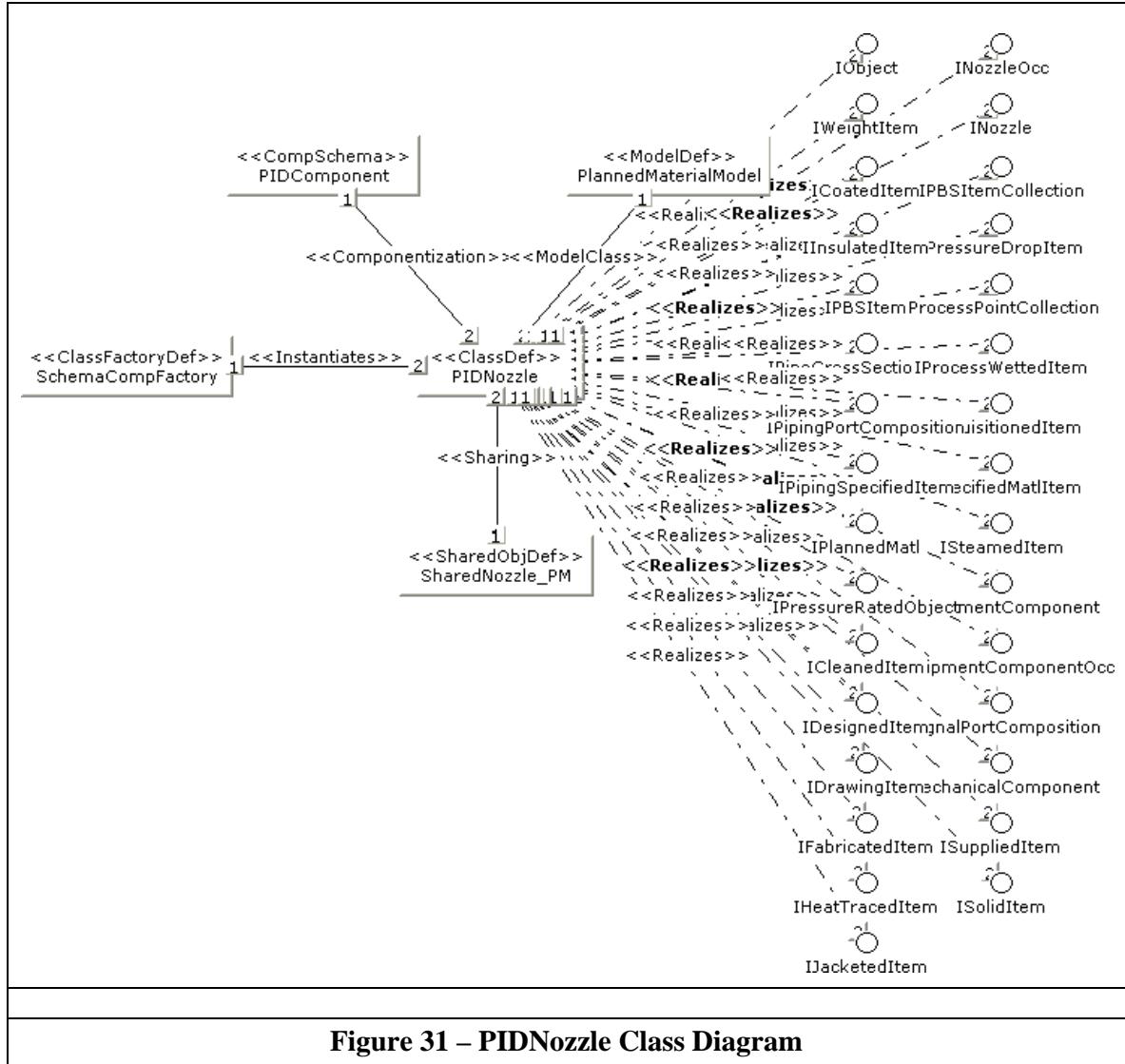
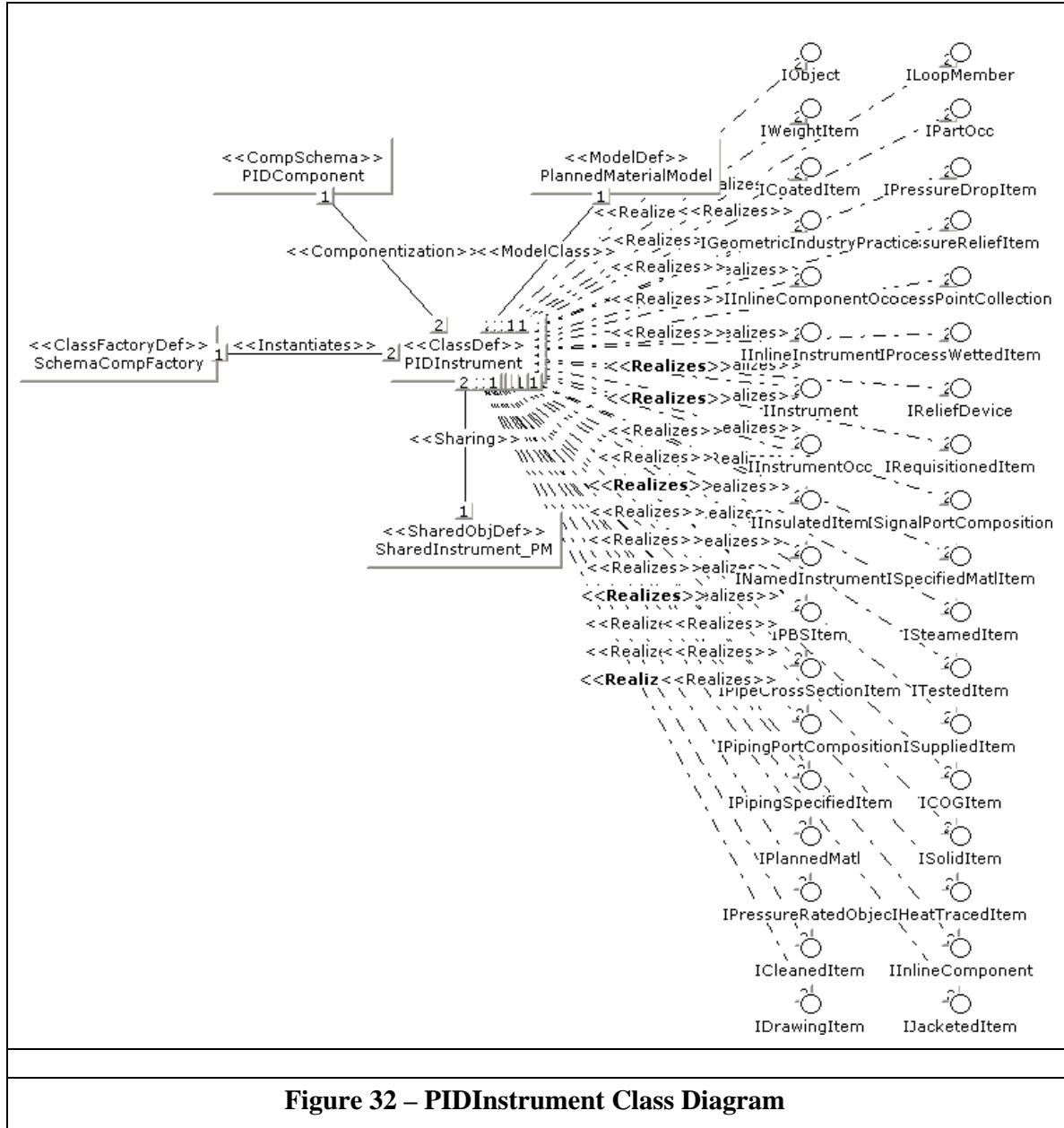


Figure 31 – PIDNozzle Class Diagram

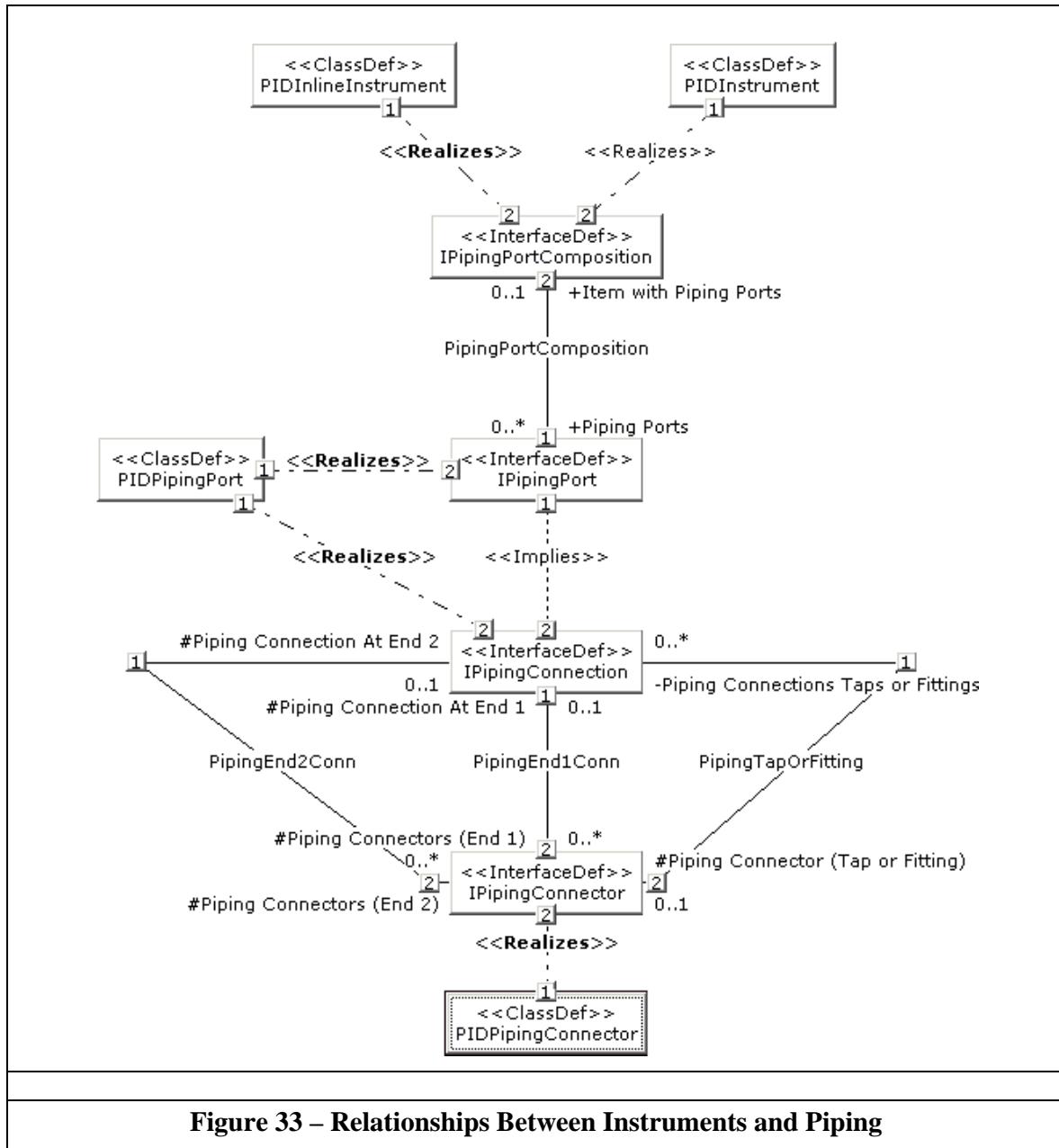
Figure 31 shows the class diagram for **PIDNozzle**, which is a very rich **ClassDef**. Notice that it has basic connectivity through the **IPBSItem** and **IDrawingItem** **InterfaceDefs**. It is also connected to the equipment through the **IEquipmentComponent** **InterfaceDef**.



Another top-level object published by P&ID is the PIDInstrument. This ClassDef represents pressure and temperature instruments, transmitters, gauges, switches, and so on.

Once again, basic connectivity is achieved through IPBSItem and IDrawingItem. The exact type of instrument is determined by the ProcFunc, InstrumentType, and InstrumentType3 PropertyDefs, which are scoped by the InstrumentProcessFunctions EnumListType.

Other types of instruments such as control valves, flow elements, and safety relief devices are published as PIDInlineInstrument. This ClassDef is similar to the PIDInstrument ClassDef. The main difference is that inline instruments have connectivity to PIDPipingConnectors through the IPipingPortComposition InterfaceDef. See Figure 33 for more information on IPipingPortComposition.



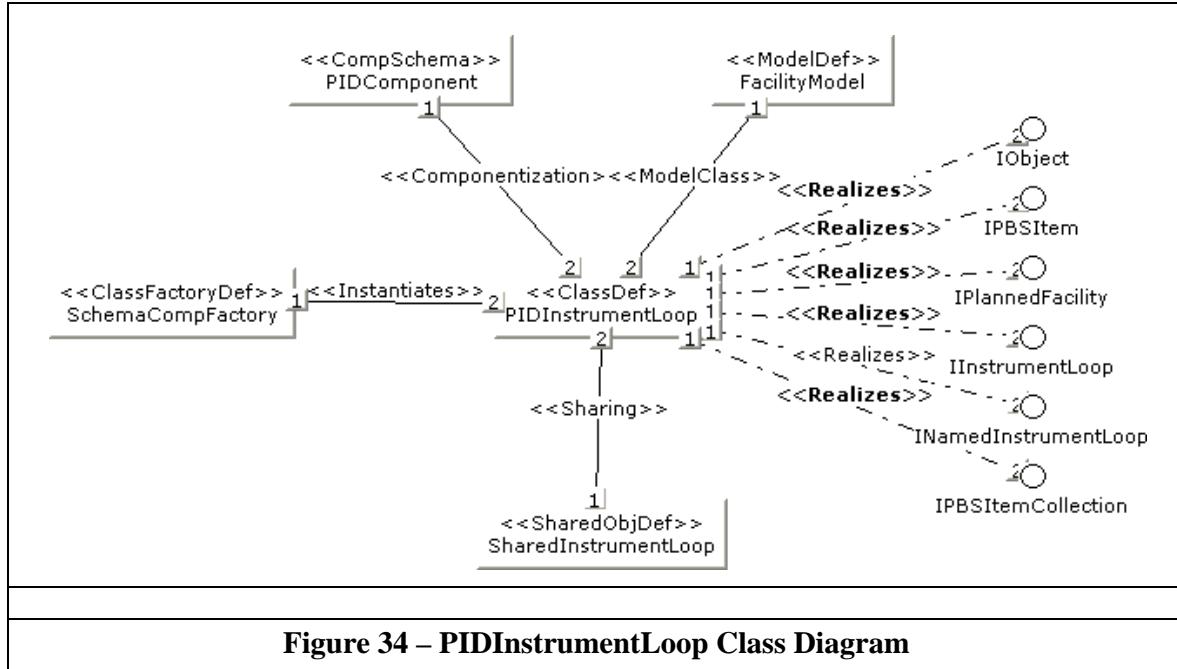


Figure 34 – PIDInstrumentLoop Class Diagram

SmartPlant P&ID also publishes instrument loops, which are retrieved by SmartPlant Instrumentation (see Figure 34).

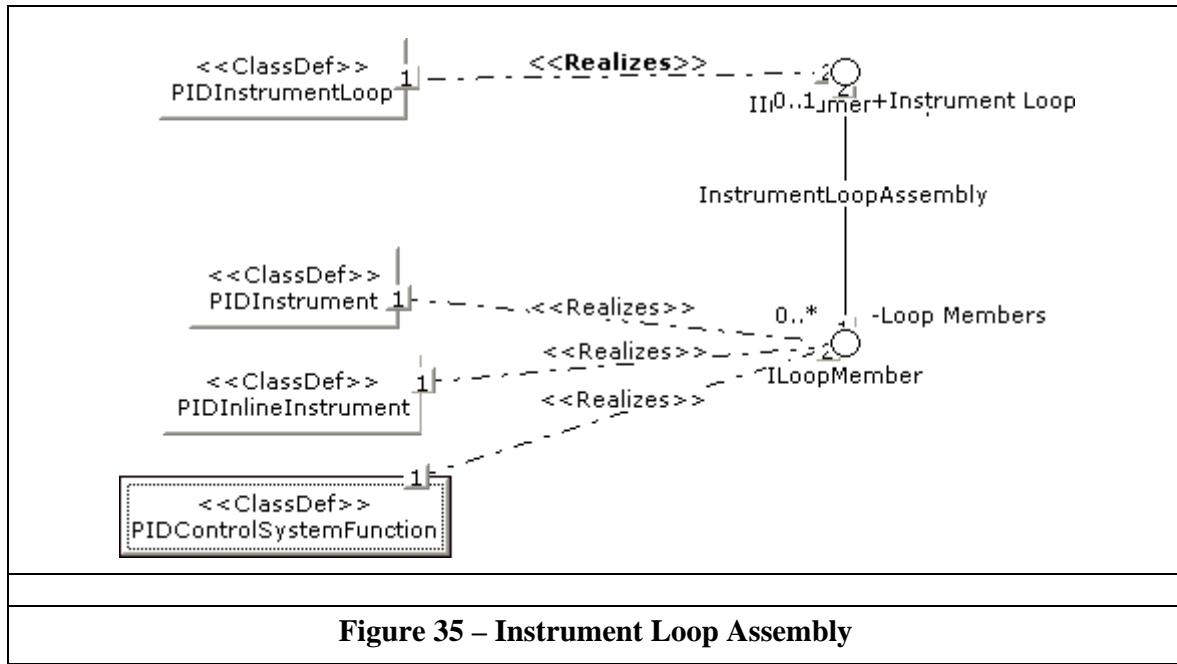
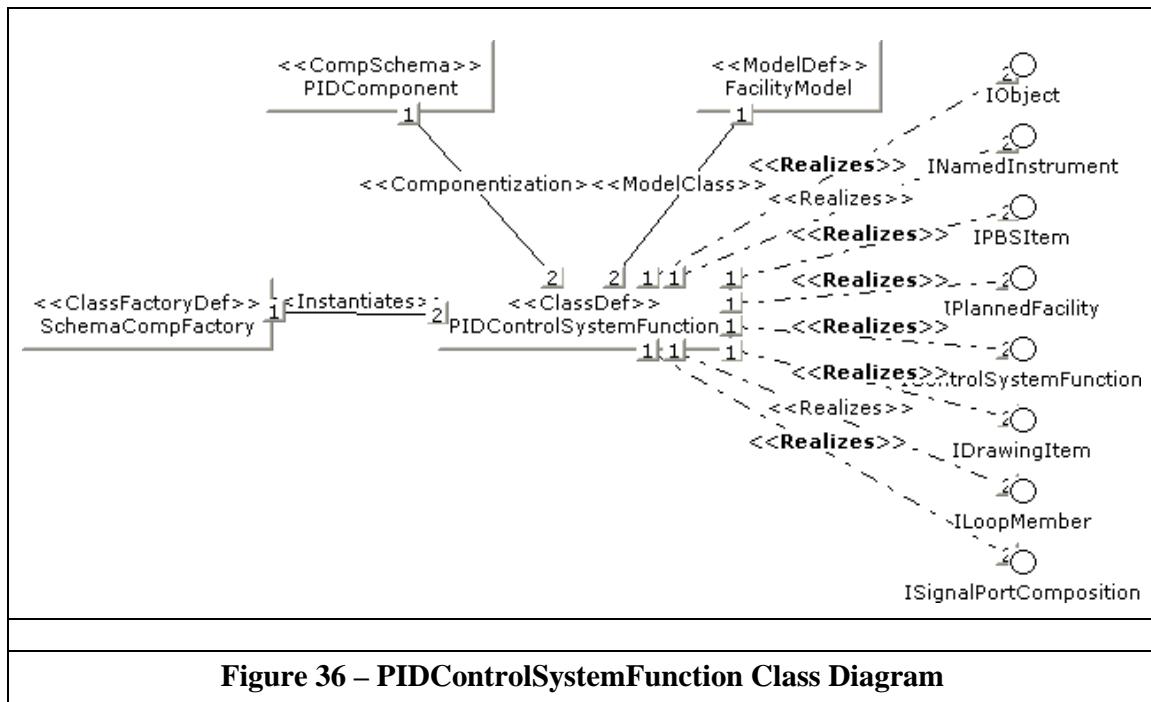
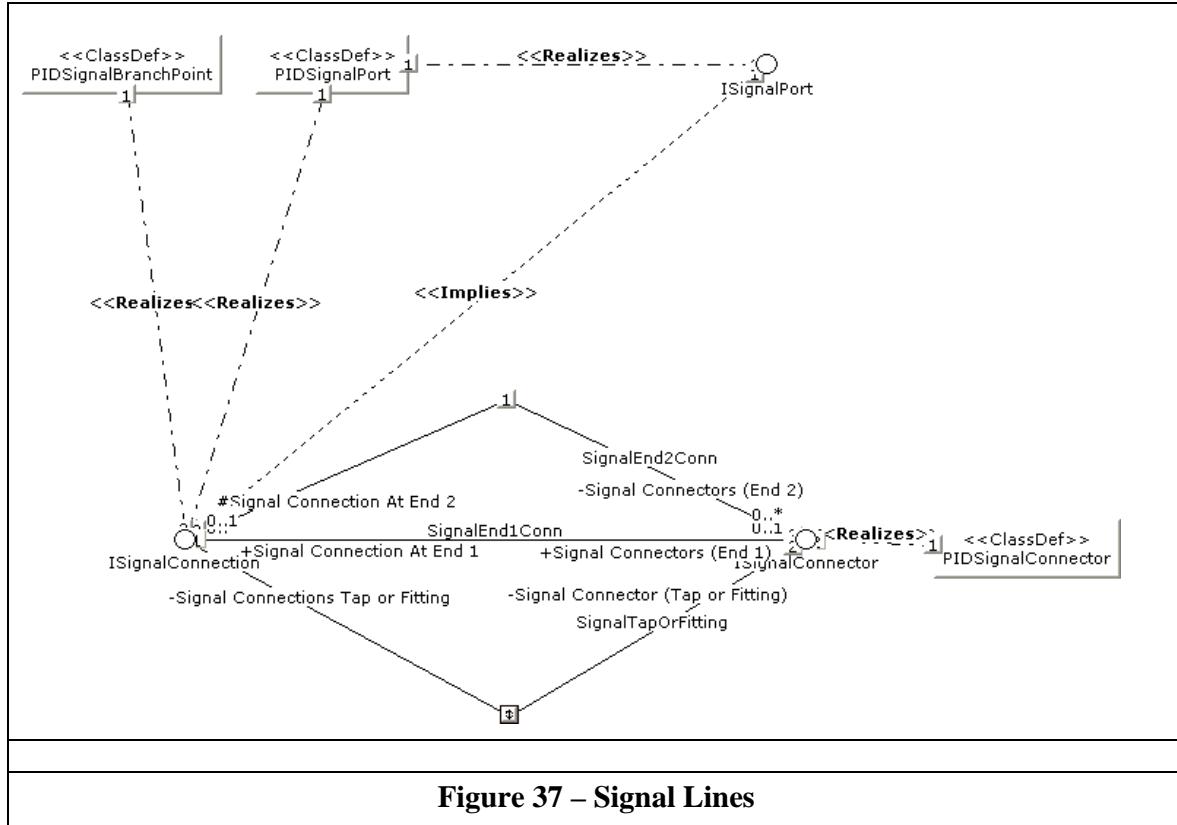


Figure 35 – Instrument Loop Assembly

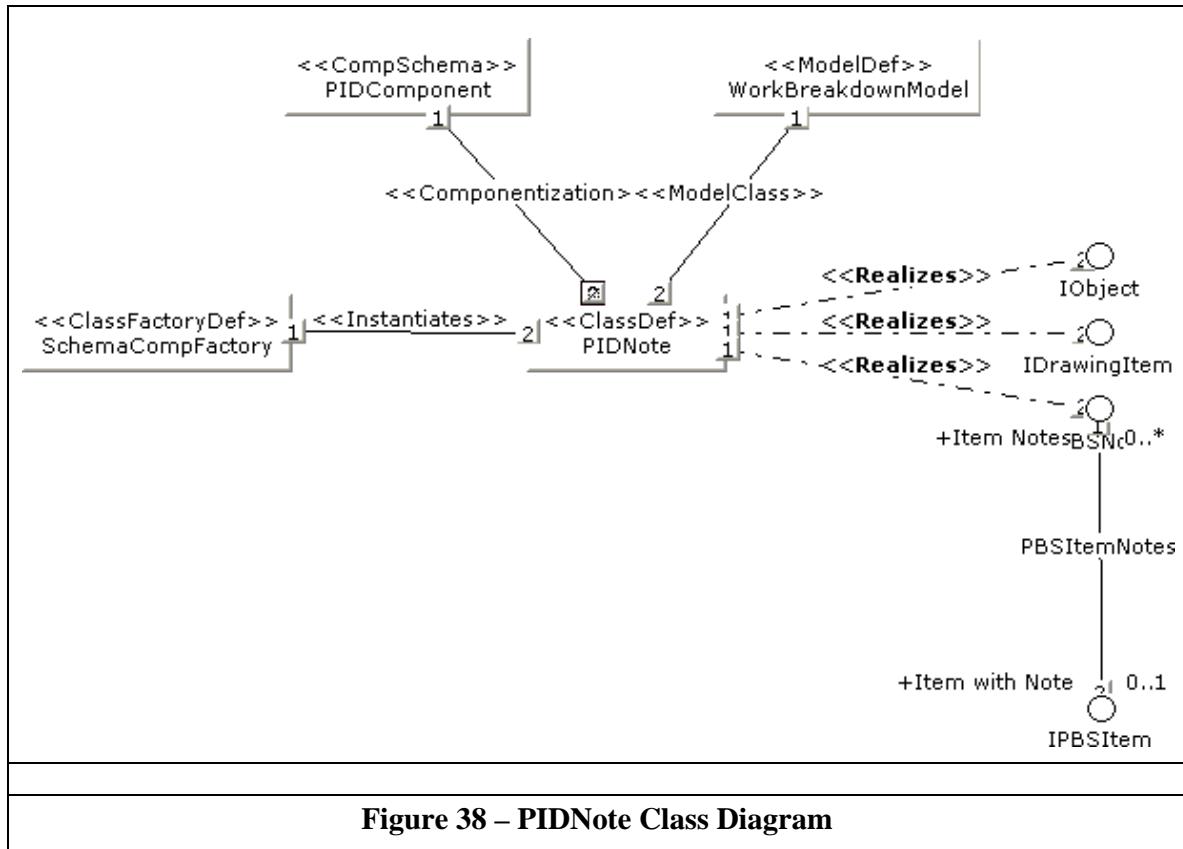
Figure 35 shows the relationship between instruments and loops published by SmartPlant P&ID. Notice that instruments, inline instruments, and control system functions are all loop members.



The **PIDControlSystemFunction** ClassDef is published for DCS instruments.



SmartPlant P&ID publishes control or signal lines in much the same way it publishes process lines. Notice the three ClassDefs involved: PIDSIGNALCONNECTOR, PIDSIGNALPORT, and PIDSIGNALBRANCHPOINT. Also, notice the relationships between ISIGNALPORT and ISIGNALCONNECTOR. There are RelDefs for end 1 and end 2 connectivity as well as the Tap or Fitting RelDef for a “T” relationship.



One last ClassDef to mention in the PID component is the PIDNote. This is freeform text that may be associated with almost any item on a P&ID.

Notice that this functionality is defined in a very generic way. The IPBSNote InterfaceDef carries the note properties, and the PBSItemNotes RelDef is defined on the IPBSItem InterfaceDef. This means that notes can be associated with nearly every object in SPF.

A.8 Instrument Index Component Schema

SmartPlant Instrumentation publishes ClassDefs from two component schemas: InstrumentIndexComponent and DimensionalDatasheetComponent. Most documents published by SPI fall under the instrument index component. This includes the instrument index, loop diagram, process data for instruments, and the instrument spec sheet. These documents are published as INDXDocument.

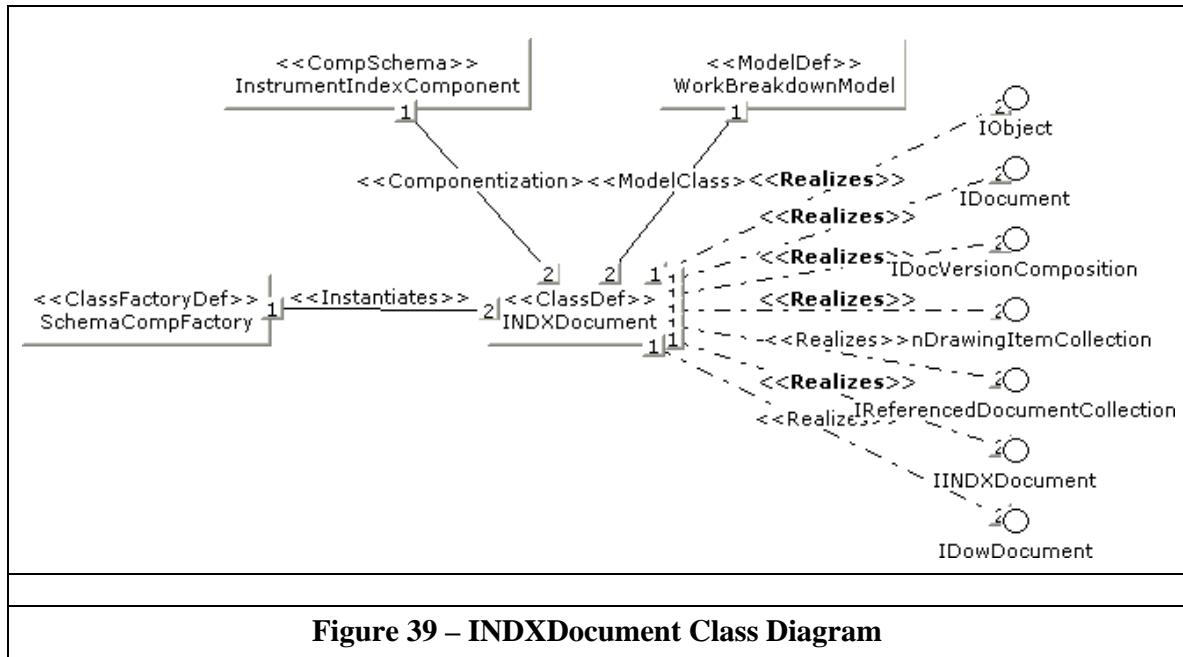
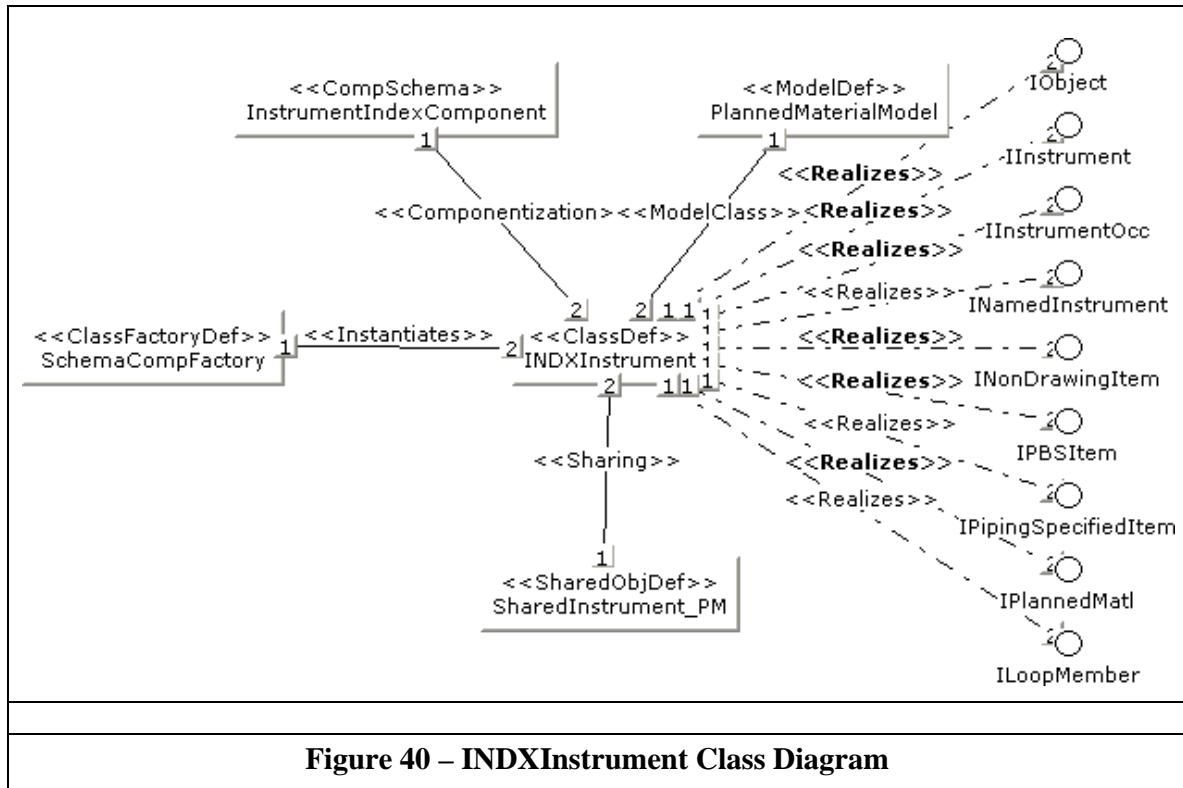


Figure 39 – INDXDocument Class Diagram

You can see from Figure 39 that INDXDocument is a typical document. The instruments and loops published in the document are related through the NonDrawingItemCollection relationship. The document type being published, however, determines the type of view files and data published. Table 3 shows the data and file formats published with each type.

Table 3 – INDXDocument Types

Document Type	Data Types	File Formats
Instrument index	Instruments and Loops	PDF
Loop Diagram	Loop (1)	PDF, SMA (RAD)
Process Data	Instrument (1)	PDF, IPD
Spec Sheet	Instruments	PDF, ISF



Instruments are published as INDXInstrument; Figure 40 shows that class diagram. Standard connectivity to the PBS and Document exist through IPBSItem and INonDrawingItem respectively.

The instruments are typed through the ProcFunc, InstrumentType and InstrumentType3 properties on IInstrument. These properties are scoped by the InstrumentProcessFunctions EnumListType.

Instruments are related to the instrument loop via ILoopMember (see Figure 42).

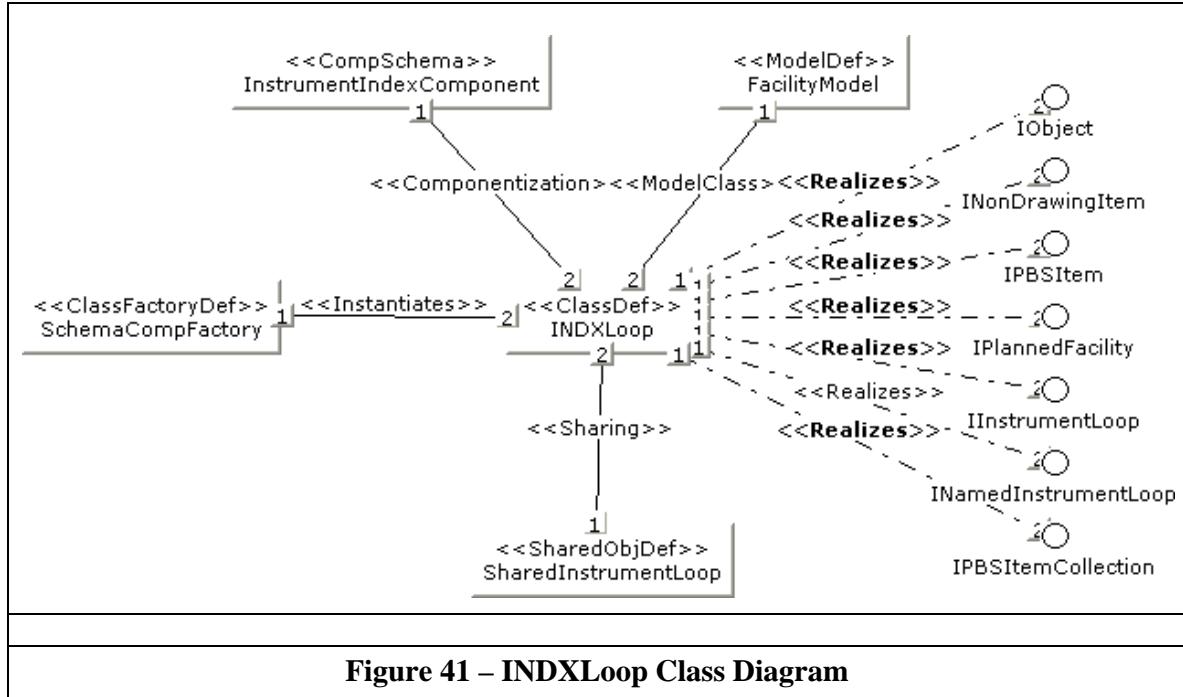


Figure 41 – INDXLoop Class Diagram

Figure 41 shows the class diagram for INDXLoop, which is an instrument loop. Connectivity to the PBS and document is published through IPBSItem and INonDrawingItem respectively.

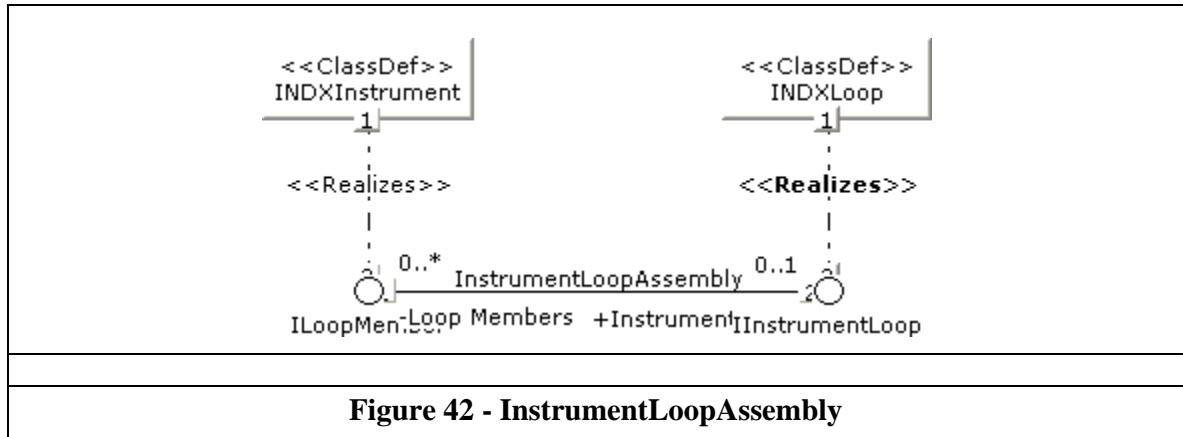


Figure 42 - InstrumentLoopAssembly

Figure 42 shows the relationship between instruments and loops.

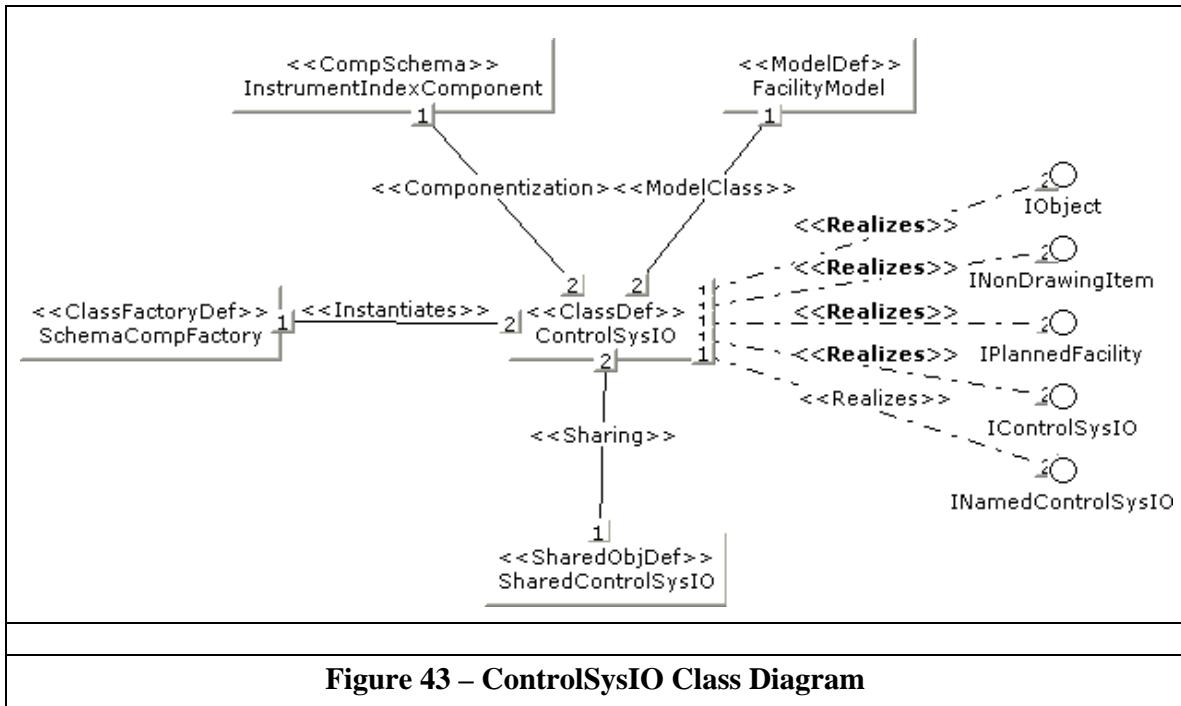


Figure 43 – ControlSysIO Class Diagram

The final object published via INDXDocument is of ClassDef `ControlSysIO`. This object represents the name of the instrument as it appears in the control system.

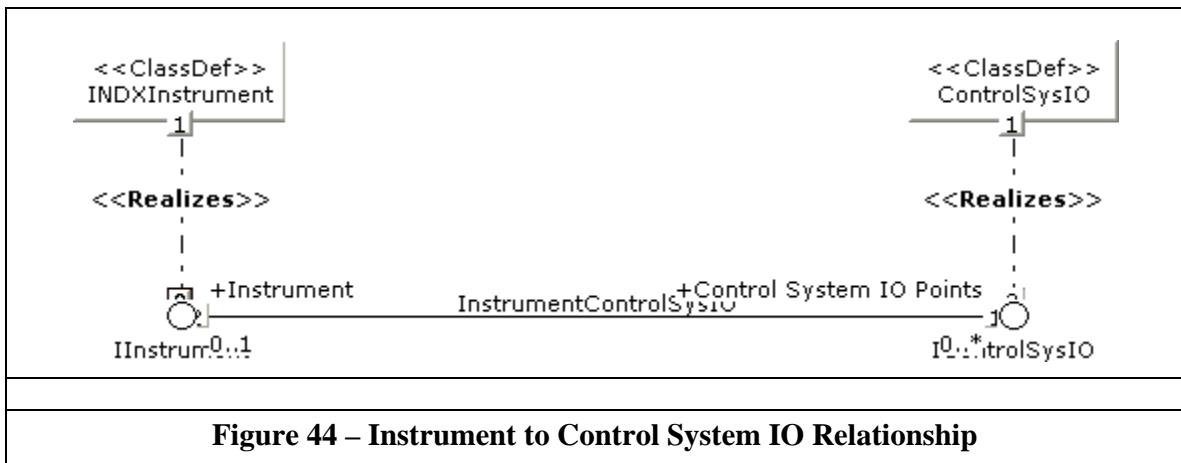
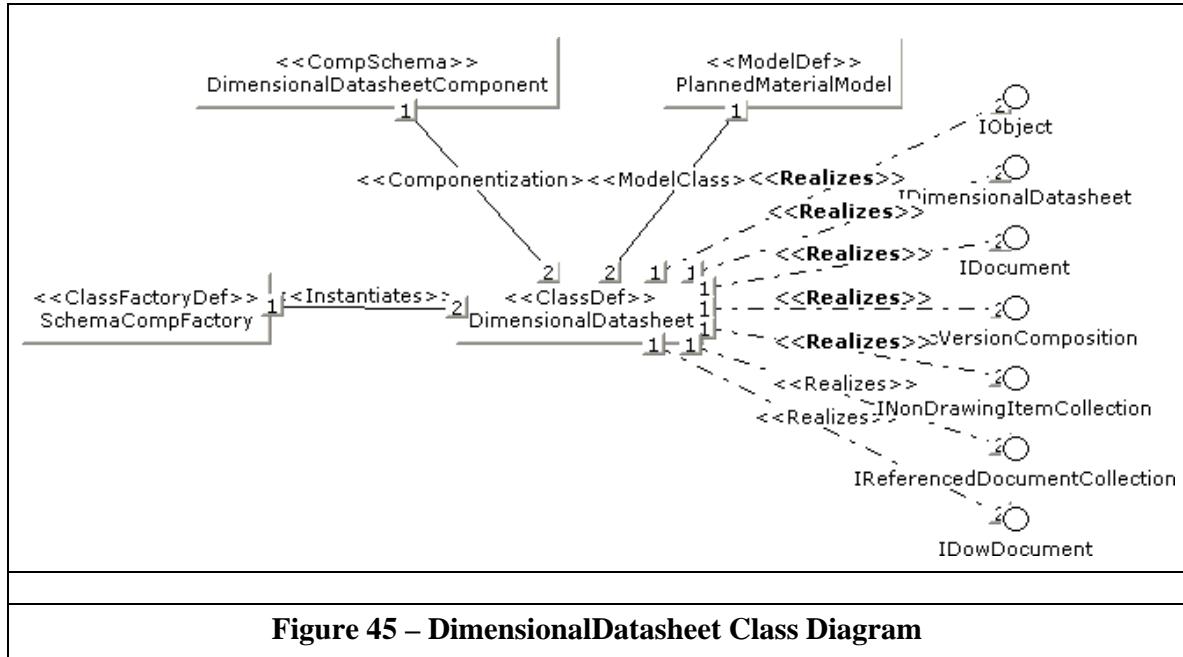


Figure 44 – Instrument to Control System IO Relationship

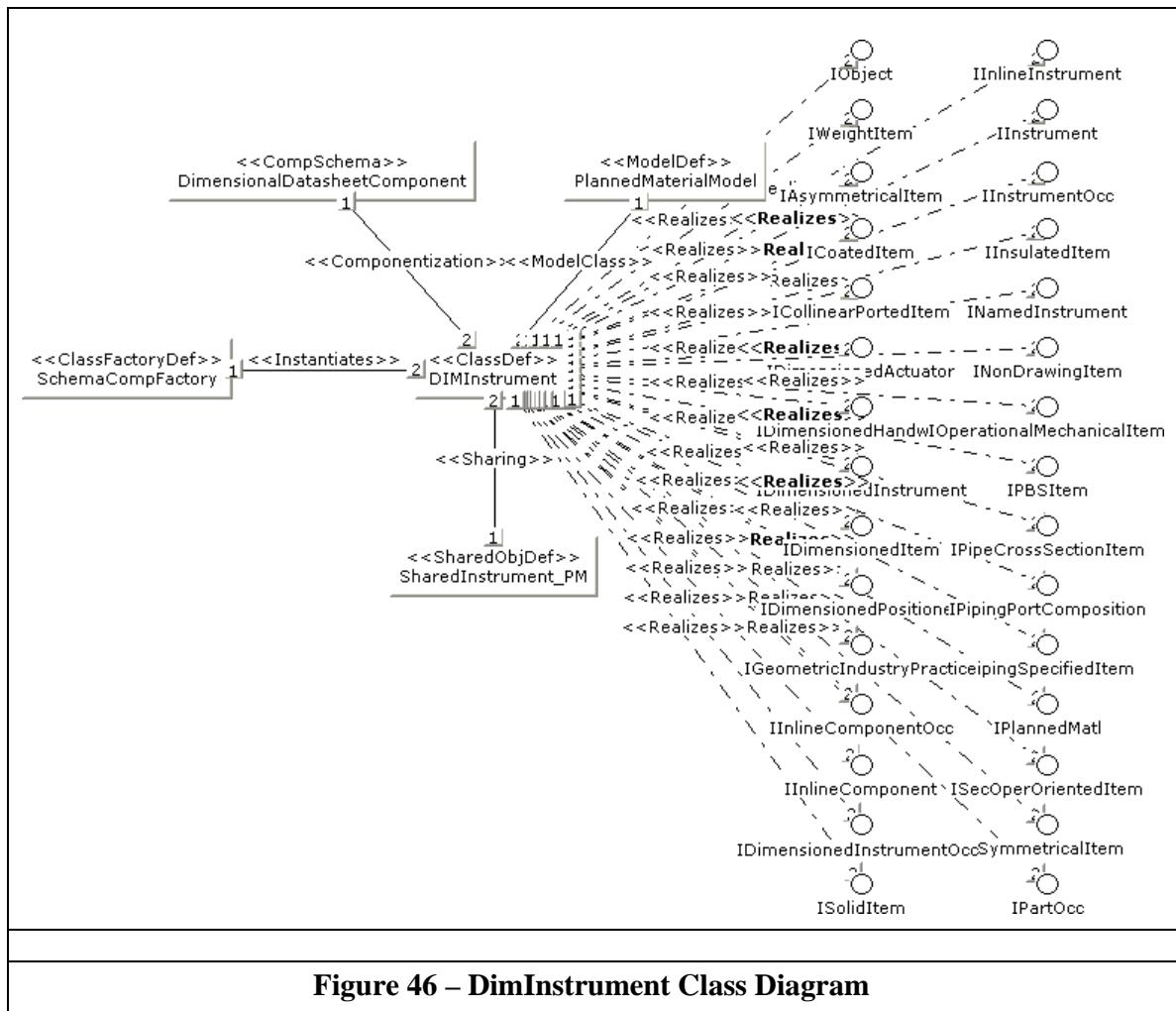
The InterfaceDef `INamedControlSysIO` has properties for each component of the name of this object. Figure 44 shows the relationship between instruments and control system IO tags. Notice that an instrument may have multiple control system IO tags.

A.9 Dimensional Instrument Component Schema

The second of SPI's component schemas is the Dimensional Instrument component. This component schema defines the publish of a dimensional datasheet.



As you can see from its class diagram in Figure 45, the DimensionalDatasheet is a typical publishable document. Because it is not a graphical document as published by SPI, it realizes INonDrawingItemCollection for connectivity to the data.



Two ClassDefs describe all the data that is published with a dimensional datasheet. DIMInstrument and DIMPipingPort. Notice that the ClassDef for DIMInstrument is extremely rich with realizes relationships. Table 4 shows shows the properties exposed by those interfaces. Almost all the properties with a length or angle UOM are dimensional properties.

Table 4 – DIMInstrument Class Table

Interface	Property	Type
IObject	UID	string128
IObject	Name	string128
IObject	Description	string
IWeightItem	EmptyWeight	MassUoM
IWeightItem	WaterFilledWeight	MassUoM
IAsymmetricalItem	FacetoEndDimension	LengthUoM
IAsymmetricalItem	Face1ToCenterlineDimension	LengthUoM
IAsymmetricalItem	Face2ToCenterlineDimension	LengthUoM
IAsymmetricalItem	Face3ToCenterlineDimension	LengthUoM
IAsymmetricalItem	Face4ToCenterlineDimension	LengthUoM
ICoatedItem	CoatingRequirement	CoatingReq1
ICoatedItem	CoatingType	CoatingReq2
ICoatedItem	ExteriorSurfaceTreatment	ExteriorSurfaceTreatments
ICoatedItem	InteriorSurfaceTreatment	InteriorSurfaceTreatments
ICollinearPortedItem	FaceToFaceDimension	LengthUoM
IDimensionedActuator	ActuatorOffset	LengthUoM
IDimensionedActuator	ActuatorDiameter	LengthUoM
IDimensionedActuator	ActuatorHeight	LengthUoM
IDimensionedHandwheel	HandwheelDiameter	LengthUoM
IDimensionedInstrument	InstrumentWidth	LengthUoM
IDimensionedInstrument	InstrumentDiameter	LengthUoM
IDimensionedInstrument	InstrumentHeight	LengthUoM
IDimensionedInstrument	InstrumentLength	LengthUoM
IDimensionedInstrument	InstrumentDimGroup	InstrumentDimGroups
IDimensionedItem	BendBranchAngle	AngleUoM
IDimensionedItem	ValveHeight	LengthUoM
IDimensionedItem	ValveHeight1	LengthUoM
IDimensionedItem	FlowDiameter	LengthUoM
IDimensionedItem	CylHeight	LengthUoM

Interface	Property	Type
IDimensionedItem	InstrumentOffset	LengthUoM
IDimensionedItem	InstrumentHeight1	LengthUoM
IDimensionedItem	InstrumentLength1	LengthUoM
IDimensionedItem	InstrumentHeight2	LengthUoM
IDimensionedItem	InstrumentLength2	LengthUoM
IDimensionedItem	InstrumentHeight3	LengthUoM
IDimensionedItem	InstrumentLength3	LengthUoM
IDimensionedItem	ActuatorLength3	LengthUoM
IDimensionedItem	ActuatorWidth3	LengthUoM
IDimensionedItem	HandWheelOffset1	LengthUoM
IDimensionedItem	ActuatorCylDiameter	LengthUoM
IDimensionedItem	OperationDiameter1	LengthUoM
IDimensionedItem	ValveOutertoCenter	LengthUoM
IDimensionedItem	InstrumentWidth2	LengthUoM
IDimensionedItem	InstrumentWidth3	LengthUoM
IDimensionedItem	ActuatorLength2	LengthUoM
IDimensionedItem	ActuatorHeight1	LengthUoM
IDimensionedItem	ActuatorHeight2	LengthUoM
IDimensionedItem	ActuatorWidth2	LengthUoM
IDimensionedItem	ActuatorOffset1	LengthUoM
IDimensionedItem	ActuatorCyl1Diameter	LengthUoM
IDimensionedItem	ActuatorCyl1Length	LengthUoM
IDimensionedItem	ActuatorCylLength	LengthUoM
IDimensionedItem	ActuatorHeight3	LengthUoM
IDimensionedItem	ActuatorHeight4	LengthUoM
IDimensionedItem	ActuatorHeight5	LengthUoM
IDimensionedItem	ActuatorLength1	LengthUoM
IDimensionedItem	ActuatorLength4	LengthUoM
IDimensionedItem	ActuatorLength5	LengthUoM

Interface	Property	Type
IDimensionedItem	ActuatorWidth1	LengthUoM
IDimensionedItem	ActuatorWidth4	LengthUoM
IDimensionedItem	ArmHeight	LengthUoM
IDimensionedItem	ArmHeight1	LengthUoM
IDimensionedItem	ArmLength	LengthUoM
IDimensionedItem	OrificeFlangeClearance	LengthUoM
IDimensionedItem	HandWheelLength	LengthUoM
IDimensionedItem	HandWheelOffset	LengthUoM
IDimensionedItem	HandWheelOffset2	LengthUoM
IDimensionedItem	InstrumentDiameter1	LengthUoM
IDimensionedItem	InstrumentDiameter2	LengthUoM
IDimensionedItem	InstrumentHeight4	LengthUoM
IDimensionedItem	InstrumentHeight5	LengthUoM
IDimensionedItem	InstrumentHeight6	LengthUoM
IDimensionedItem	InstrumentHeight7	LengthUoM
IDimensionedItem	InstrumentHeight8	LengthUoM
IDimensionedItem	InstrumentHeight9	LengthUoM
IDimensionedItem	InstrumentLength4	LengthUoM
IDimensionedItem	InstrumentLength5	LengthUoM
IDimensionedItem	InstrumentRadius	LengthUoM
IDimensionedItem	InstrumentRadius1	LengthUoM
IDimensionedItem	InstrumentRadius2	LengthUoM
IDimensionedItem	InstrumentRadius3	LengthUoM
IDimensionedItem	Motor1Diameter	LengthUoM
IDimensionedItem	Motor1Length	LengthUoM
IDimensionedItem	Motor1Offset	LengthUoM
IDimensionedItem	Motor2Diameter	LengthUoM
IDimensionedItem	Motor2Length	LengthUoM
IDimensionedItem	Motor2Offset	LengthUoM

Interface	Property	Type
IDimensionedItem	Motor3Diameter	LengthUoM
IDimensionedItem	Motor3EndtoCenter	LengthUoM
IDimensionedItem	Motor3Length	LengthUoM
IDimensionedItem	Motor4Diameter	LengthUoM
IDimensionedItem	Motor4Length	LengthUoM
IDimensionedItem	Motor4Offset	LengthUoM
IDimensionedItem	NozzleOffset	LengthUoM
IDimensionedItem	NozzleOffset1	LengthUoM
IDimensionedItem	NozzleOffset2	LengthUoM
IDimensionedItem	NozzleOffset3	LengthUoM
IDimensionedItem	PositionerOffset1	LengthUoM
IDimensionedItem	PositionerOffset2	LengthUoM
IDimensionedItem	ValveOutertoHandWheel	LengthUoM
IDimensionedItem	PortRotation1	AngleUoM
IDimensionedItem	PortRotation2	AngleUoM
IDimensionedItem	PortRotation3	AngleUoM
IDimensionedItem	PortRotation4	AngleUoM
IDimensionedItem	ActuatorRotation1	AngleUoM
IDimensionedItem	ActuatorRotation2	AngleUoM
IDimensionedItem	ActuatorRotation3	AngleUoM
IDimensionedItem	ActuatorRotation4	AngleUoM
IDimensionedItem	ActuatorDiameter1	LengthUoM
IDimensionedItem	ActuatorLength	LengthUoM
IDimensionedItem	ActuatorWidth	LengthUoM
IDimensionedItem	CylOffset	LengthUoM
IDimensionedItem	CylOffset1	LengthUoM
IDimensionedItem	InstrumentOffset1	LengthUoM
IDimensionedItem	InstrumentWidth1	LengthUoM
IDimensionedItem	InstrumentWidth4	LengthUoM

Interface	Property	Type
IDimensionedItem	OperationHeight1	LengthUoM
IDimensionedItem	OperationWidth1	LengthUoM
IDimensionedItem	ValveWidth	LengthUoM
IDimensionedItem	Motor3toCenter	LengthUoM
IDimensionedPositioner	PositionerHeight	LengthUoM
IDimensionedPositioner	PositionerLength	LengthUoM
IDimensionedPositioner	PositionerOffset	LengthUoM
IGeometricIndustryPractice		
IInlineComponentOcc		
IInlineInstrument		
IInstrument	InstrumentType	InstrumentTypes2
IInstrument	ProcFunc	InstrumentProcessFunctions
IInstrument	InstrumentType3	InstrumentTypes3
IInstrument	InstrumentSysIOType	ElecSystemIOTypes
IInstrumentOcc	InstrumentService	string
IInstrumentOcc	InstrumentLocation	InstrumentLocations
IInstrumentOcc	InstrumentManufacturerDeviceID	string
IInstrumentOcc	InstrumentManufacturerRevision	string
IInsulatedItem	InsulTemp	TemperatureUoM
IInsulatedItem	InsulThickSrc	InsulThickSrcs
IInsulatedItem	InsulSpec	string
IInsulatedItem	InsulCompositeMatl	InsulCompositeMaterials
IInsulatedItem	AvgInsulDens	DensityUoM
IInsulatedItem	TotalInsulThick	LengthUoM
IInsulatedItem	InsulPurpose1	InsulPurposes1
IInsulatedItem	InsulPurpose2	InsulPurposes2
IInsulatedItem	InsulPurpose3	InsulPurposes3
INamedInstrument	InstrFuncModifier	string
INamedInstrument	InstrLoopSuffix	string

Interface	Property	Type
INamedInstrument	InstrTagPrefix	string
INamedInstrument	InstrTagSequenceNo	string
INamedInstrument	InstrTagSuffix	string
INamedInstrument	MeasuredVariable	string
INonDrawingItem		
IOperationalMechanicalItem	OperationDiameter	LengthUoM
IOperationalMechanicalItem	OperationHeight	LengthUoM
IOperationalMechanicalItem	OperationLength	LengthUoM
IOperationalMechanicalItem	OperationWidth	LengthUoM
IPBSItem	ConstructionStatus	ConstructionStati
IPBSItem	ConstructionStatus2	ConstructStati2
IPipeCrossSectionItem	ScheduleOrThickness	ScheduleThicknessPractices
IPipeCrossSectionItem	WallThickness	LengthUoM
IPipeCrossSectionItem	ScheduleThickness2	ScheduleThicknesses2
IPipeCrossSectionItem	SchedulesOverridden	boolean
IPipeCrossSectionItem	OuterDiameter	LengthUoM
IPipeCrossSectionItem	NominalDiameter	LengthUoM
IPipingPortComposition	InstrWeldingRqmt	WeldingRqmts
IPipingPortComposition	InstrBoltingRqmt	BoltingRqmts
IPipingPortComposition	InstrGasketRqmt	GasketReqmts
IPipingSpecifiedItem	PipingMaterialsClass	string
IPlannedMatl		
ISecOperOrientedItem	OperRotation	AngleUoM
ISymmetricalItem	FaceToCenterDimension	LengthUoM
IPartOcc	CatalogName	CatalogNames
IPartOcc	CatalogPartNumber	CatalogPartNumbers
IInlineComponent		
IDimensionedInstrumentOcc	DimDataStatus	DimDataStati
IDimensionedInstrumentOcc	DimDataStatusChgDate	YMD

Interface	Property	Type
IDimensionedInstrumentOcc	DimDataStatusRevision	string
ISolidItem		

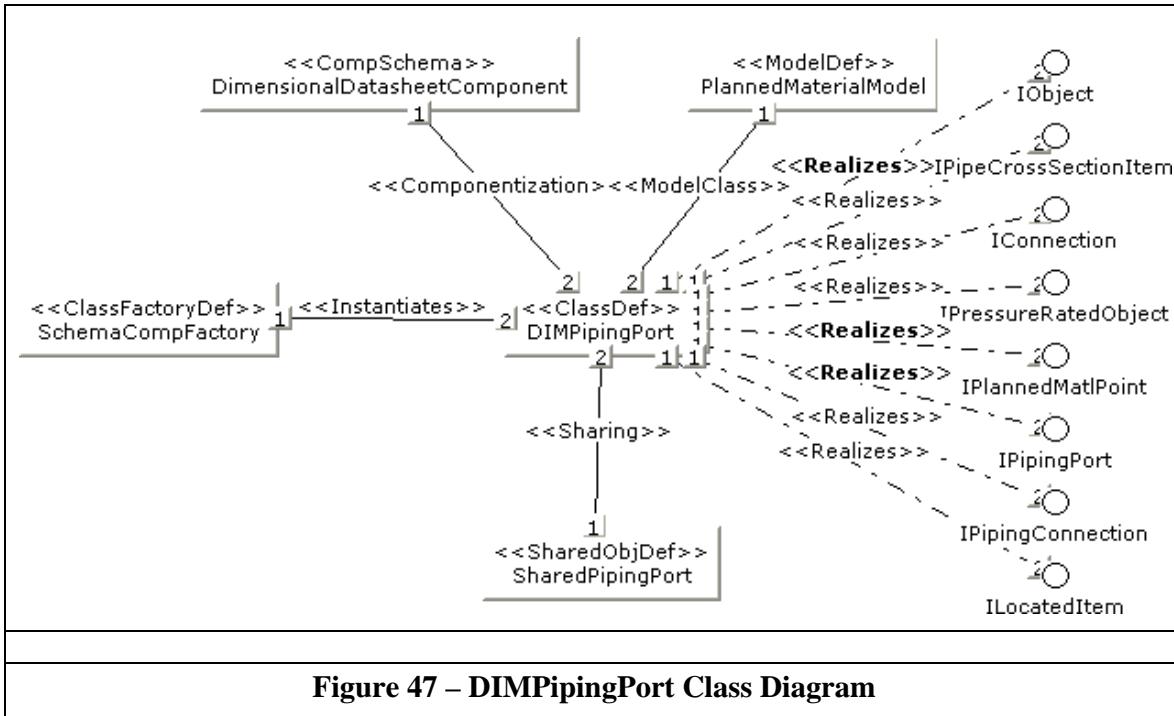


Figure 47 shows the class diagram for the **DIMPipingPort**. The only dimensional properties are exposed from **IPipeCrossSectionItem**. Because the piping ports are components of the instrument, these are ports of the dimensional datasheet as well.

A.10 3D Component Schema

The P3DComponent schema defines all the data published by SmartPlant 3D today. There are only three ClassDefs in this component: P3DDocument, P3DSmartPlantReviewDocument, and P3DEquipment.

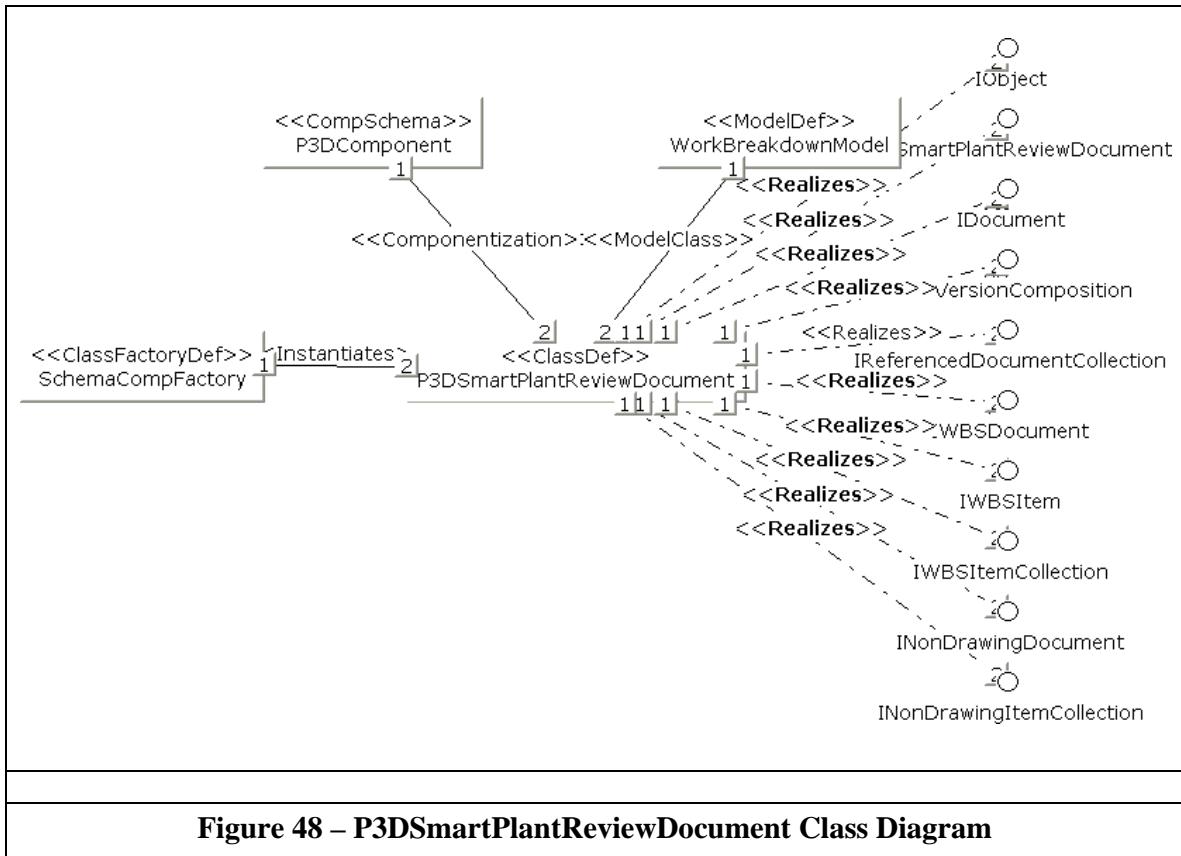


Figure 48 – P3DSmartPlantReviewDocument Class Diagram

Figure 48 shows the class diagram for P3DSmartPlantReviewDocument. This is the ClassDef published when SmartPlant 3D publishes a view of the model. Notice the lack of completeness compared to other published documents. Also, notice that the P3DSmartPlantReviewDocument ClassDef realizes the INonDrawingDocument and INonDrawingItemCollection interfaces. While a 3D model is decidedly not a drawing, it is a graphic document. To support graphic navigation, SPF requires representations to be published along with each of the objects, but SmartPlant 3D does not publish drawing representations.

Keep in mind that publishing the 3D model in SmartPlant 3D and viewing the 3D model in SPF are currently prototype software. As it becomes delivered product, you will see the modeling behind it firm up.

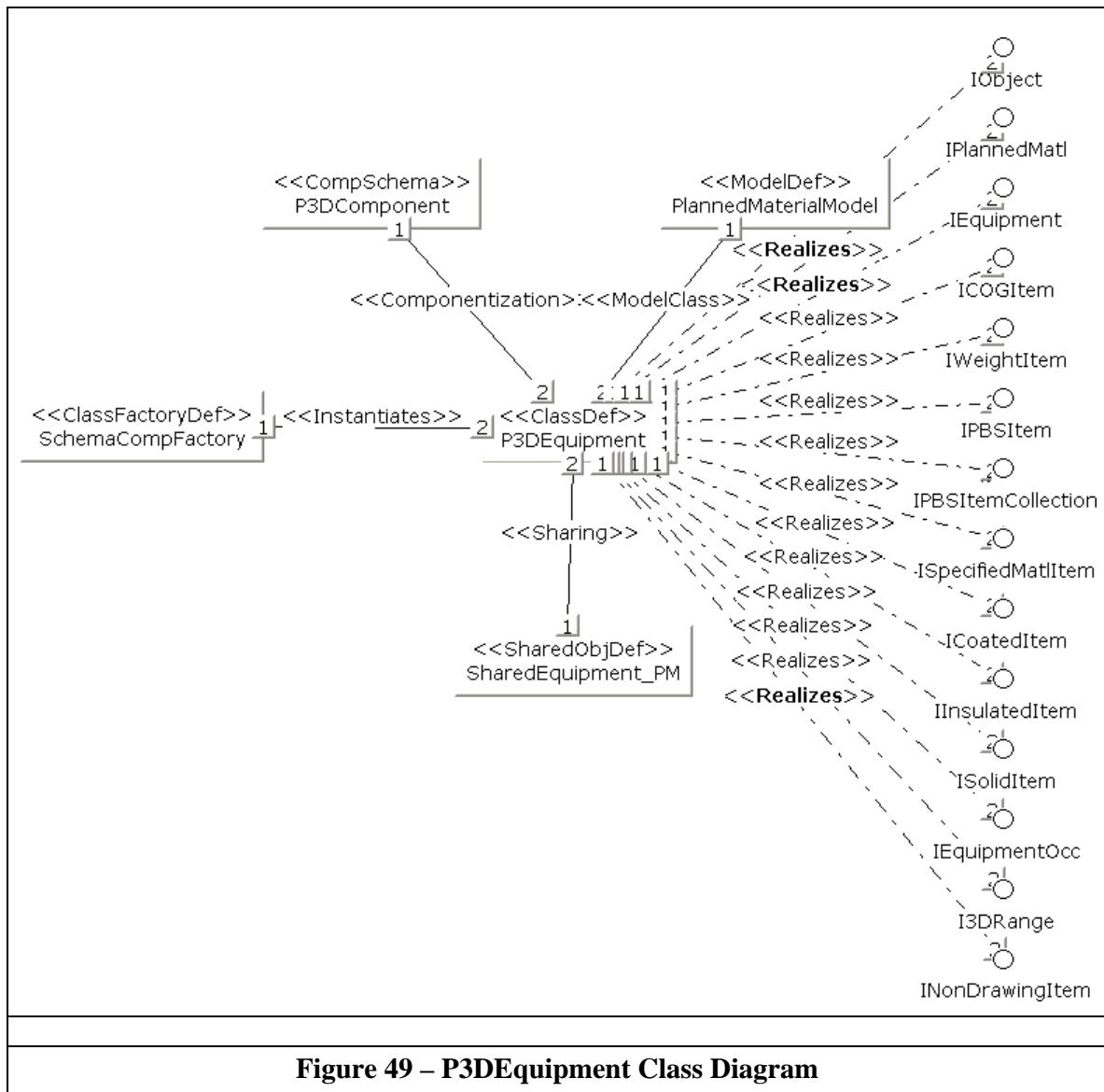


Figure 49 shows the class diagram for **P3DEquipment**. Equipment is currently the only type of data published by SmartPlant 3D. This data is published along with the 3D model. The only connectivity published to these objects is the **PBSItemCollection** relationship and the **NonDrawingItems** relationship.

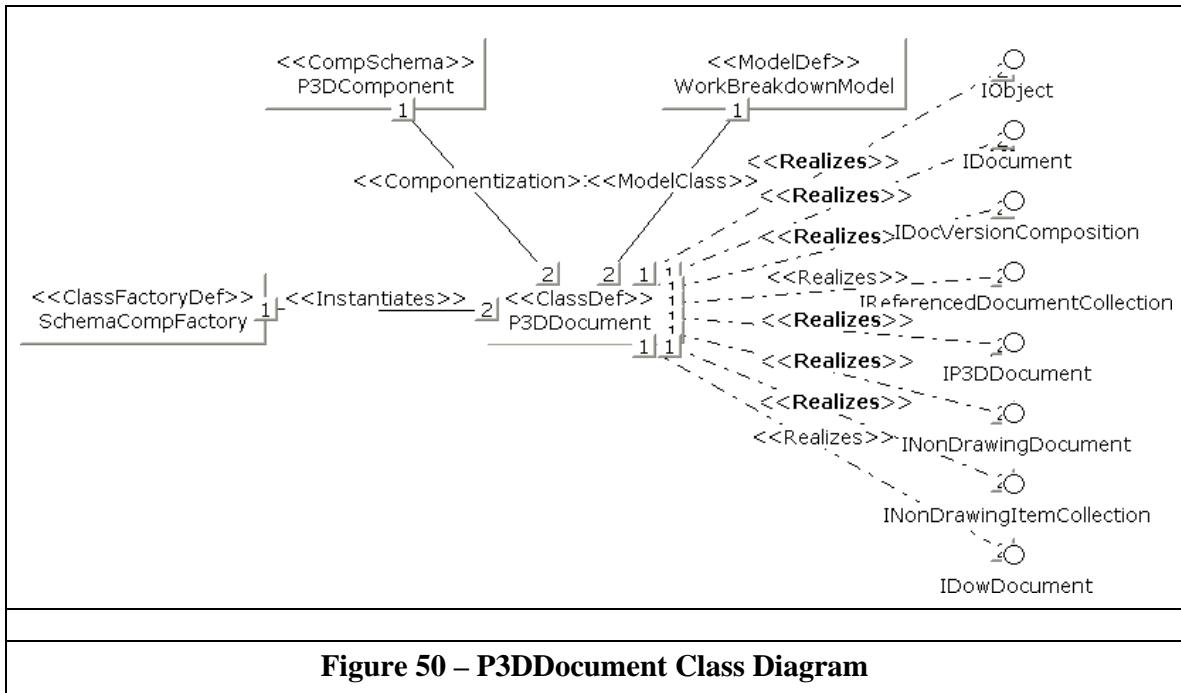


Figure 50 shows the class diagram for P3DDocument. This is the ClassDef for the 2D drawings published by the SmartPlant 3D Drawings environment. Once again, you can see that this ClassDef realizes INonDrawingDocument and INonDrawingItemCollection. This seems to be clearly in error. Even though the Drawings environment does not publish any data with the drawings, it should not realize those interfaces. In fact, the INonDrawingItemCollection InterfaceDef exists solely to form relationships between the data and the document. The SmartPlant 3D Drawings environment will soon (version 4.1?) publish data and representations to support graphical navigation.

A P P E N D I X

B

SmartPlant Schema Evolution

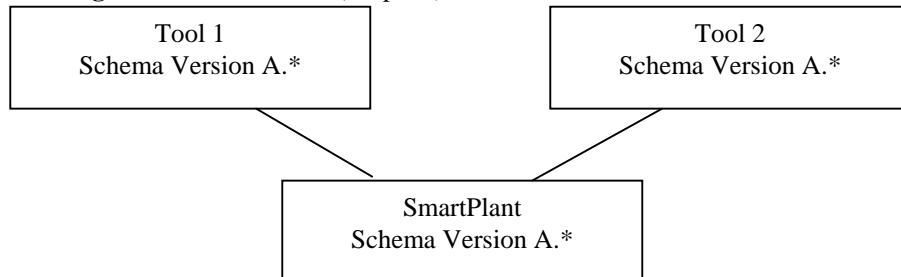
B. SmartPlant Schema Evolution

This document describes the rules for evolution of the SmartPlant meta schema and the SmartPlant schema (and indirectly all the component schemas). Unless explicitly stated otherwise, all of the rules defined in this document apply to both the meta schema and the schema.

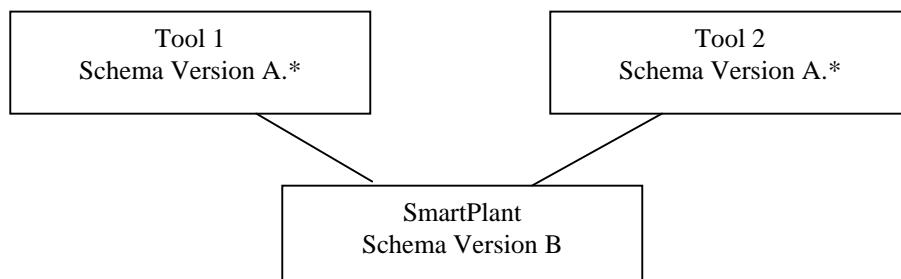
B.1 Configurations Supported by the SmartPlant Schema Evolution Rules

The following configurations of software schema are intended to be supported by the schema evolution rules:

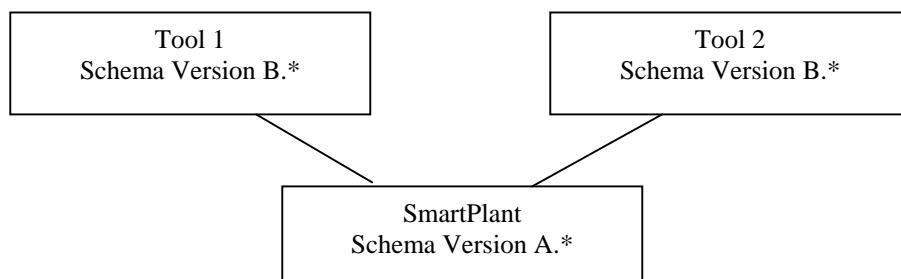
Configuration 1 – All tools (adapters) and SmartPlant on same schema version

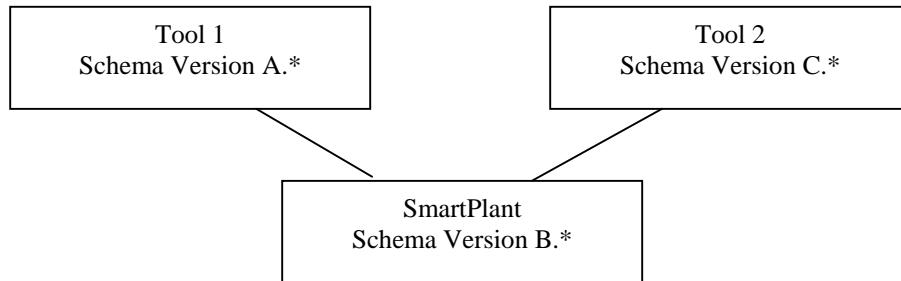


Configuration 2 – Tools on older schema version than SmartPlant



Configuration 3 – Tools on newer schema version than SmartPlant



Configuration 4 – Tools on older and newer schema versions than SmartPlant

In these diagrams, tool 1 and tool 2 are simply applications that have been written to integrate with SmartPlant.

The schema version is made up of two parts. The first part indicates the version of the schema as released by Intergraph and is given a letter designation (for example, A, B, C). The second part includes an optional * to indicate 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 has been written. The inclusion of the * indicates that the metadata (map file and so on) used by that tool has been updated to the latest set of customizations made by user and delivered by the SmartPlant schema. Since, for all configurations the * is included, the assumption is that, for all configurations, the metadata utilized by the adapter has been updated to work against the latest SmartPlant

Configuration 1 is the simplest case. In this software configuration, all of the tools and SmartPlant are written against the same version of the schema. In this configuration, there should be no issues.

In configuration 2 a new version of the SmartPlant schema has been loaded. However, each of the tools are running with their adapter written to work against the previous version. In this case, the new SmartPlant schema cannot include any changes that preclude the old tool adapters from working with the new SmartPlant schema.

In configuration 3, new versions of the tools have been loaded. For these tools, the adapters have been written against a version of the SmartPlant schema that is newer than what has been loaded. To handle this case, the tool adapters must support backward compatibility (in which they will work with less functionality against the older version of the SmartPlant schema).

Configuration 4 is the worst-case scenario. In this scenario, some of the tool adapters are working against a version of the schema older than what has been loaded and some of the tools are working against a version of the schema newer than what has been loaded.

B.2 SmartPlant Schema Evolution Rules

The schema evolution rules defined in this document are intended to avoid breaking existing tool adapter software (configuration 2 and part of configuration 4) as well as to allow newer versions of the tool adapters to work with older versions of the SmartPlant schema (configuration 3 and the rest of configuration 4).

Any changes that might break existing publish or retrieval logic are not allowed (unless the explicit rules that the adapter should adhere to in order to avoid problems are explicitly defined) since these would break configurations 2 and 4.

Tool adapters must be written to work with old versions of the SmartPlant schema. This means that additions made to the SmartPlant schema after initial release must not be hard-coded into a tool adapter unless that hard-coding is done in such a fashion (such as using If statements) as to allow the adapter to continue to work with an older version of the schema that does not include those additions. 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 loading of those differential instructions is then driven by which version of the SmartPlant schema is being used.

The detailed design that tool adapters should (must?) follow to support schema evolution is defined in another document.

B.3 Enforcement of Schema Evolution Rules

Every schema object will be stamped with the official version against which it was created. Objects that predate the latest released version must adhere to the rules defined in this document (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. The detailed design of how this will be supported is covered in another document.

B.4 Creation Rules

New objects of any type can be created. However, relationships between new objects and existing objects must not violate the Revision rules described later in this document.

New relationships between existing objects can only be created if they do not violate any of the Revision rules defined in this document.

B.5 Deletion Rules

With a few exceptions, existing schema objects cannot be deleted (because this could easily break publish and/or retrieval software). The exceptions to this rule are objects that are not exposed in the published data. This includes, but is not limited to, edge definitions, graph definitions, view definitions, and UML view definitions.

B.6 Revision Rules

The rules for changing existing schema objects are described in the following subsections. If a schema object type is not included in this section of the document, then any type of revision of the object is allowed.

B.6.1 All Object Types

Once the SmartPlant schema is delivered, all UIDs for all objects in that schema will adhere to the rules for UIDs:

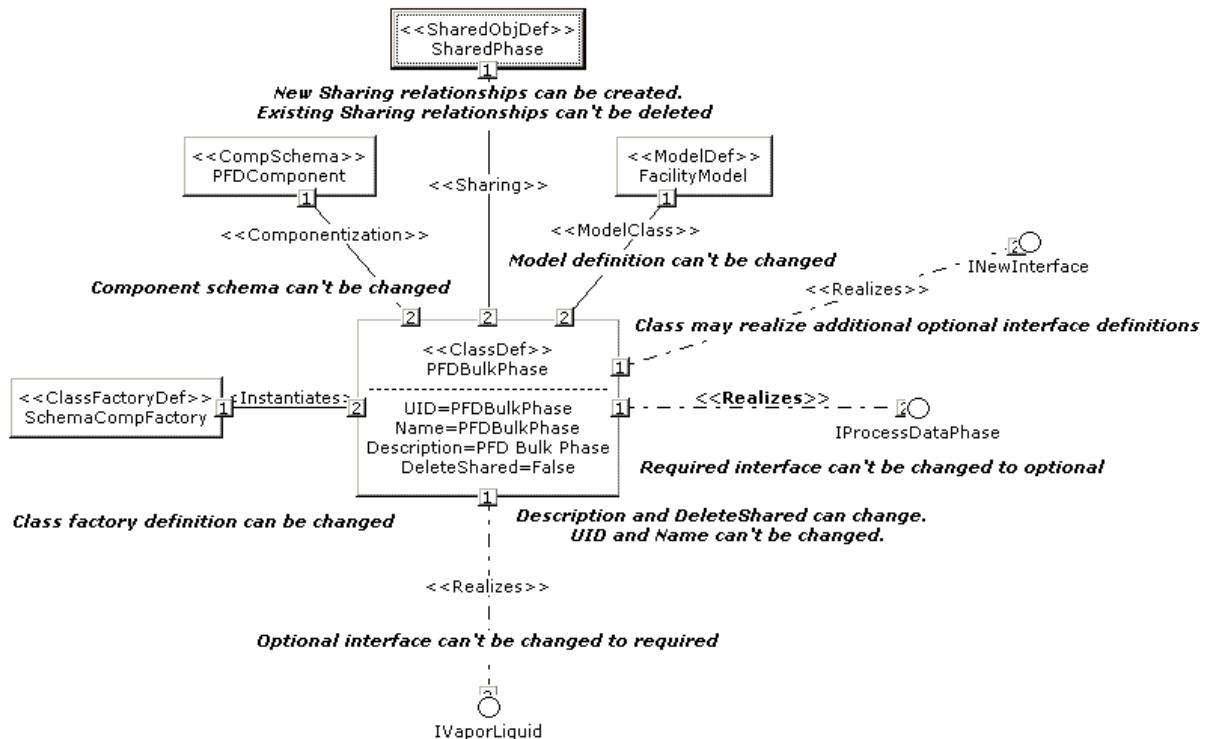
- All UIDs will be unique within the context of the SPF database.
- A UID for an object will never change.
- If an object is deleted, its UID will never be reused.

All schema objects must abide by the rules as defined by the meta schema. Revisions that would break any of these rules are not allowed.

B.6.2 Class Definitions

For existing class definitions, the following rules apply (both schema and meta schema):

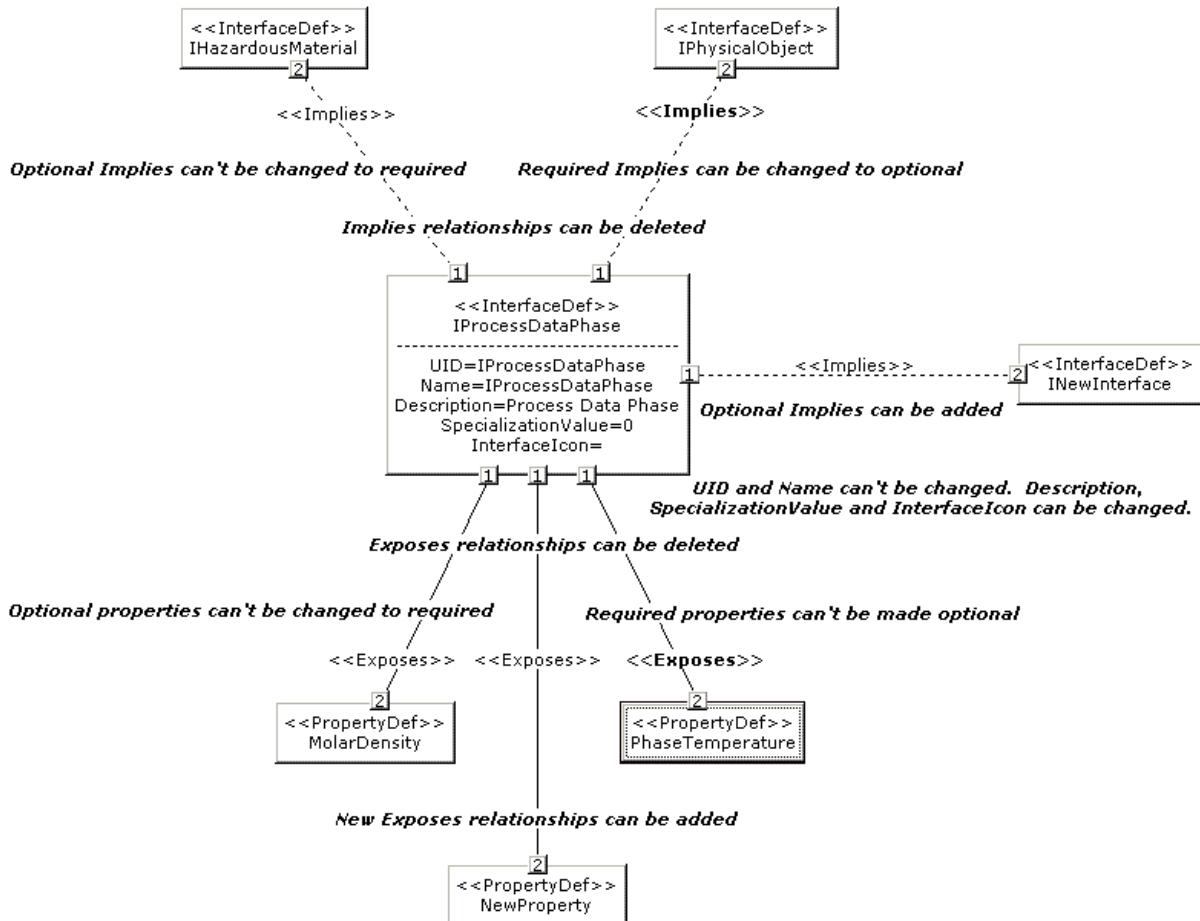
- UID and Name cannot be changed.
- Description and DeleteShared can be changed.
- Class definitions may realize additional optional interface definitions.
- An optional interface definition cannot be changed to required.
- A required interface definition cannot be changed to optional.
- The component schema for the class definition cannot be changed.
- The model definition for the class definition cannot be changed.
- The class factory definition for the class definition cannot be changed.
- New Sharing relationships can be created.
- Existing Sharing relationships cannot be deleted.



B.6.3 Interface Definitions

Although the COM rules state that interfaces are immutable, the rules for existing interface definitions in the SmartPlant schema are not so rigid:

- UID and Name cannot be changed.
- Description, SpecializationValue, and InterfaceIcon 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 be optional.
- Optional property definitions cannot be changed to be required.
- Existing Implies relationships cannot be deleted.
- New optional Implies relationships can be added.
- Required Implies relationships can be changed to optional.
- Optional Implies relationships cannot be made required.

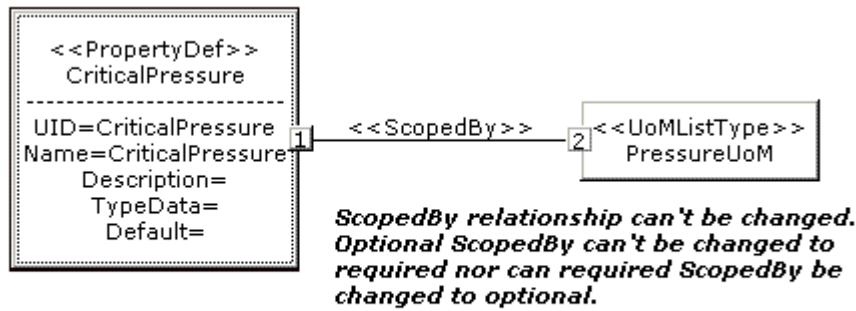


To handle the movement of property definitions from one interface definition, adapters should be written to not explicitly hard-code interface UIDs for property definitions unless the property definition is on the list of property definitions that cannot be moved.

B.6.4 Property Definitions

The following rules apply to existing property definitions:

- Name and UID cannot be changed.
- Description can be changed.
- Although not currently used, TypeData, and Default can be changed.
- The ScopedBy type for the property definition cannot be changed.
- The ScopedBy relationship cannot be changed from required to optional nor from optional to required.

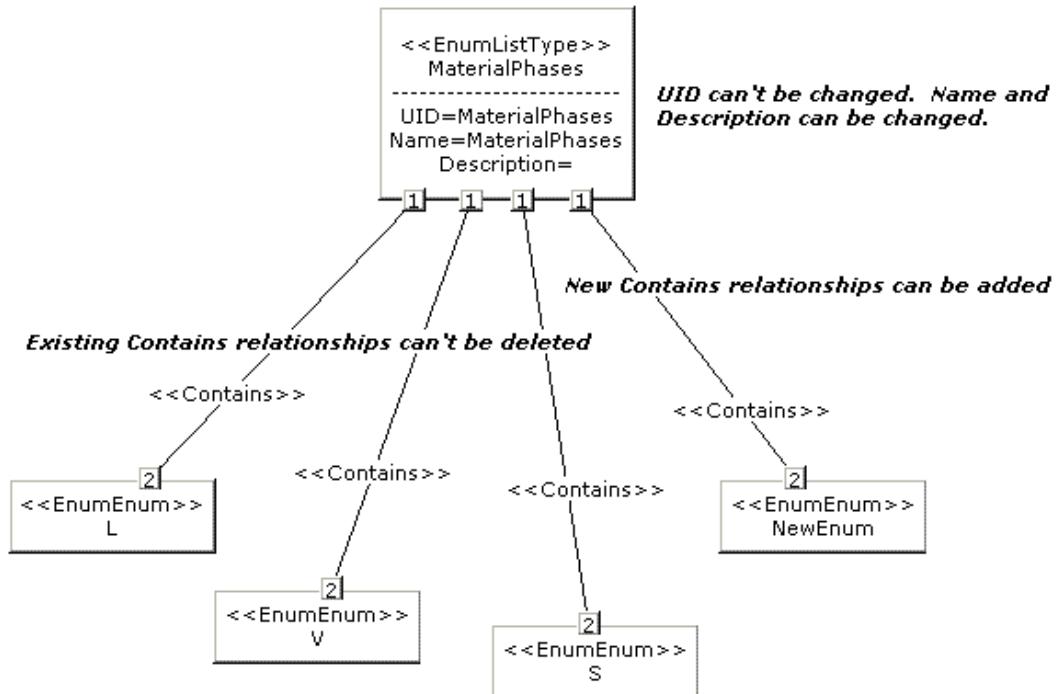


UID and Name can't be changed. Description, TypeData and Default can be changed.

B.6.5 Enumerated Lists

The following rules apply to existing enumerated lists:

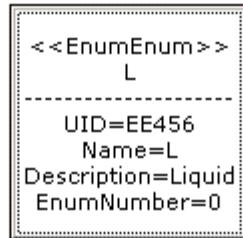
- UID cannot be changed.
- Because Name is not tied to UID, it can be changed.
- Description can be changed.
- EnumNumber, if undefined, can be given a value.
- EnumNumber, if defined, cannot be changed.
- Existing Contains relationships cannot be deleted.
- New Contains relationships can be added.



B.6.6 Enumerated Entries

The UID for an existing enumerated entry cannot be changed. However, the Name (short description) and Description can be changed.

EnumNumber, if undefined, can be given a value. If defined, EnumNumber cannot be changed.

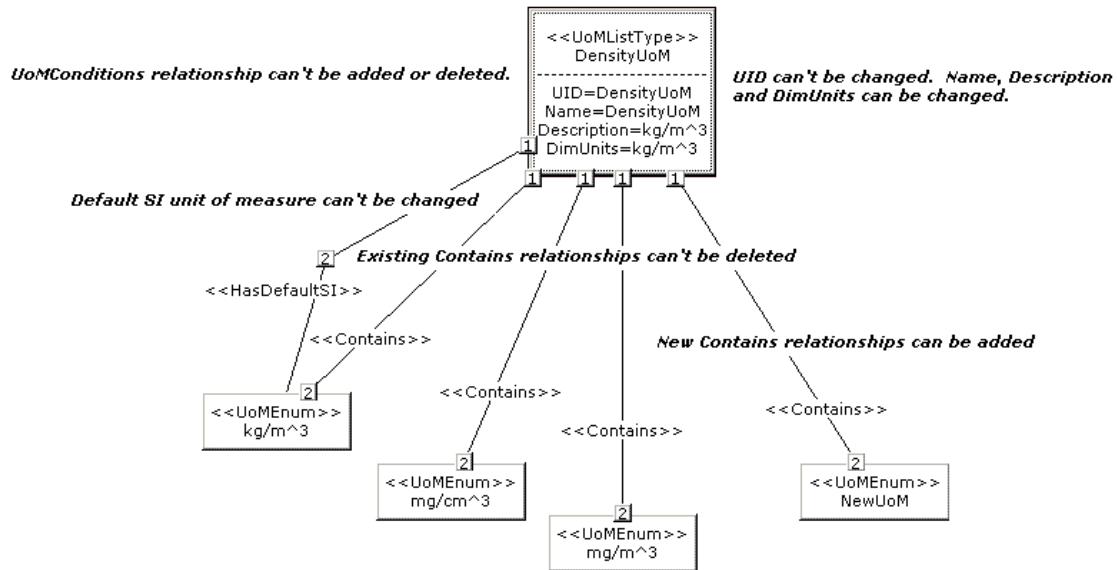


UID can't be changed. Name and Description can be changed. EnumNumber, if undefined, can be defined. EnumNumber, if defined, can't be changed.

B.6.7 Unit of Measure Lists

The following rules apply to existing enumerated lists:

- UID cannot be changed.
- Because Name is not tied to UID, it can be changed.
- Description and DimUnits can be changed.
- EnumNumber, if undefined, can be given a value.
- EnumNumber, if defined, cannot be changed.
- Existing Contains relationships cannot be deleted.
- New Contains relationships can be added.
- UoMConditions relationship cannot be deleted.
- The default SI unit of measure for the UoM list cannot be changed.



B.6.8 Unit of Measure

The UID for an existing unit of measure cannot be changed. However, the Name (short description) and Description can be changed.

EnumNumber, if undefined, can be given a value. If defined, EnumNumber cannot be changed.

The A (ACnv) and B (BCnv) conversion factors can be changed. However, this should only happen if there is an error in the value or if greater precision is desired.

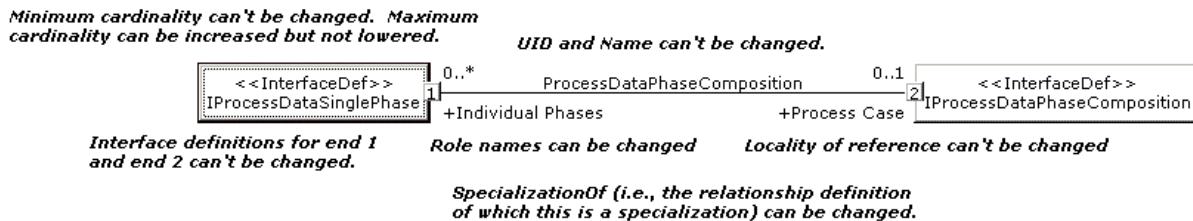
```
<<UoMEnum>>
kg/m^3
-----
UID=UoM.DensityUoM.04
Name=kg/m^3
Description=kilogram per cubic meter
EnumNumber=0
ACnv=1
BCnv=0
```

*UID can't be changed. Name and Description can be changed.
EnumNumber, if not defined, can be defined. EnumNumber, if defined, can't be changed. ACnv and BCnv can be changed but should only be changed if additional precision is desired or if an error is detected.*

B.6.9 Relationship Definitions

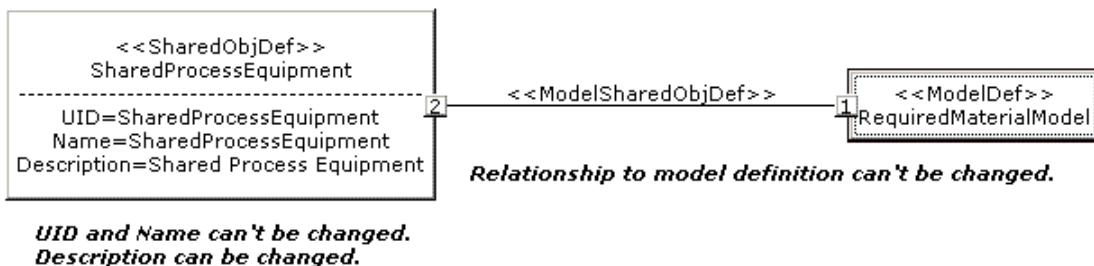
For existing relationship definitions, the revision rules are:

- UID and Name 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 the relationship must not change.
- The relationship definition of which this relationship definition is a specialization can change.



B.6.10 Shared Object Definitions

For shared object definitions, the Name and UID cannot be changed, but the Description can be changed. The model definition for the shared object definition cannot be changed.



A P P E N D I X

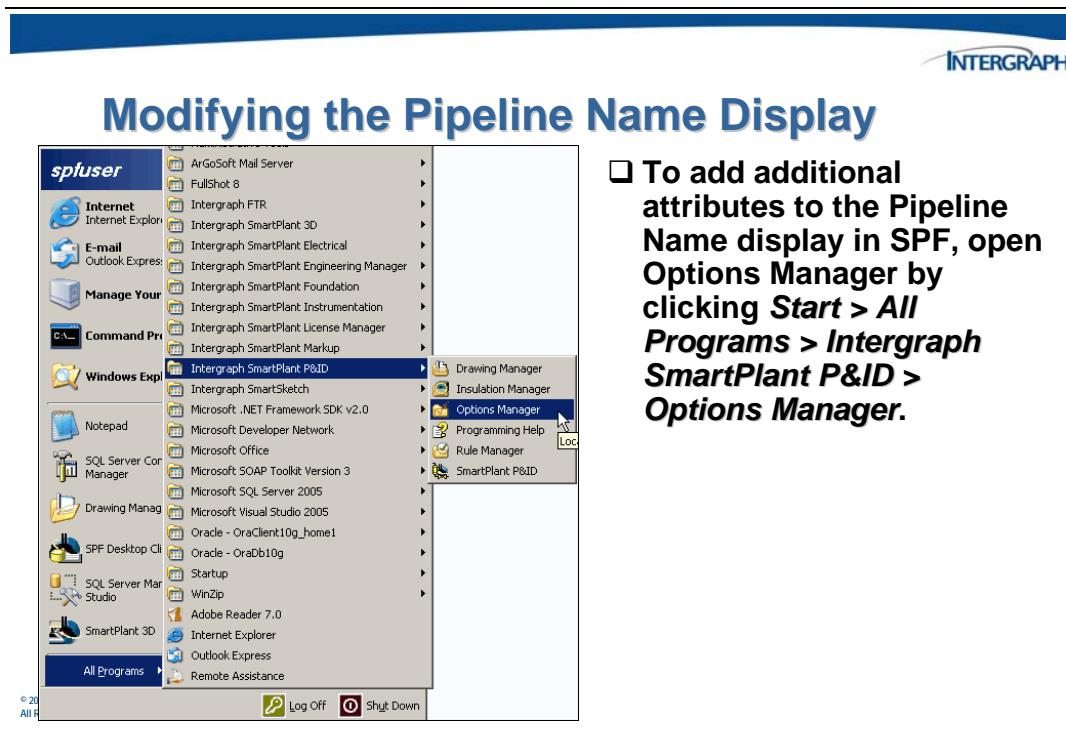
C

Modifying Pipeline Name Display

C. Modifying the Pipeline Name Display

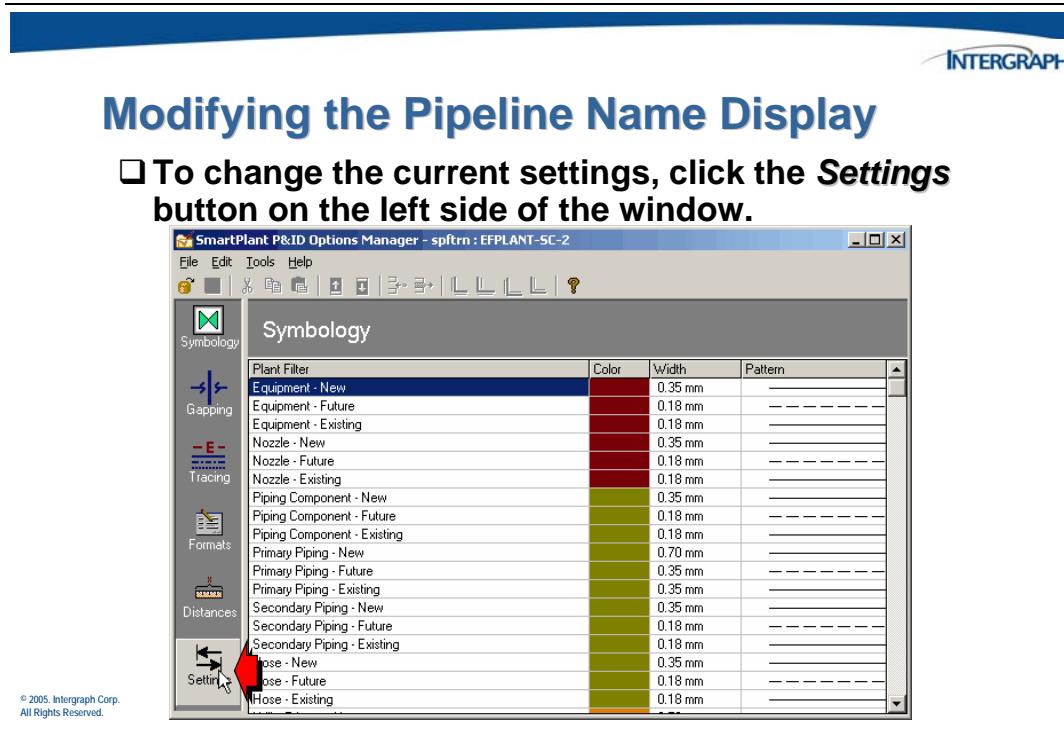
The steps for modifying the Pipeline Name display in SmartPlant Foundation are a common procedure that is frequently requested. Because of this, it was decided to define the steps to accomplish Pipeline renaming in this section. This type of operation is another of the most common ones performed.

Begin by adding the needed attributes or properties to the Pipeline Name setting in SmartPlant P&ID Options Manager.

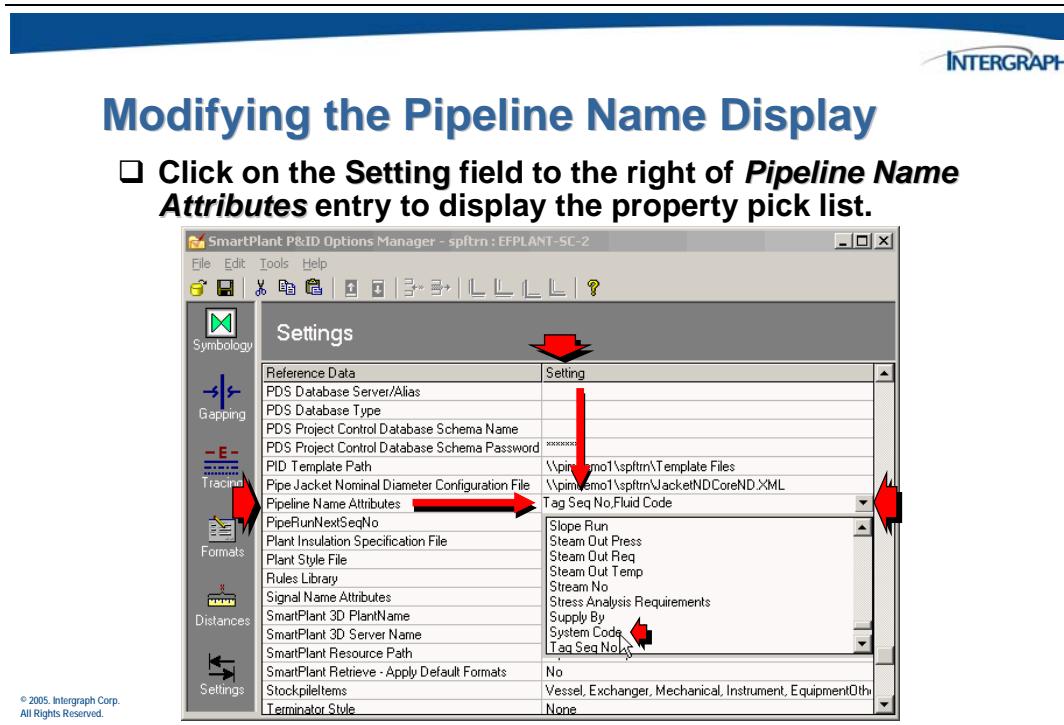


The following example describes how to add entries to the *PipeLine Name Attributes* field from the drop down list.

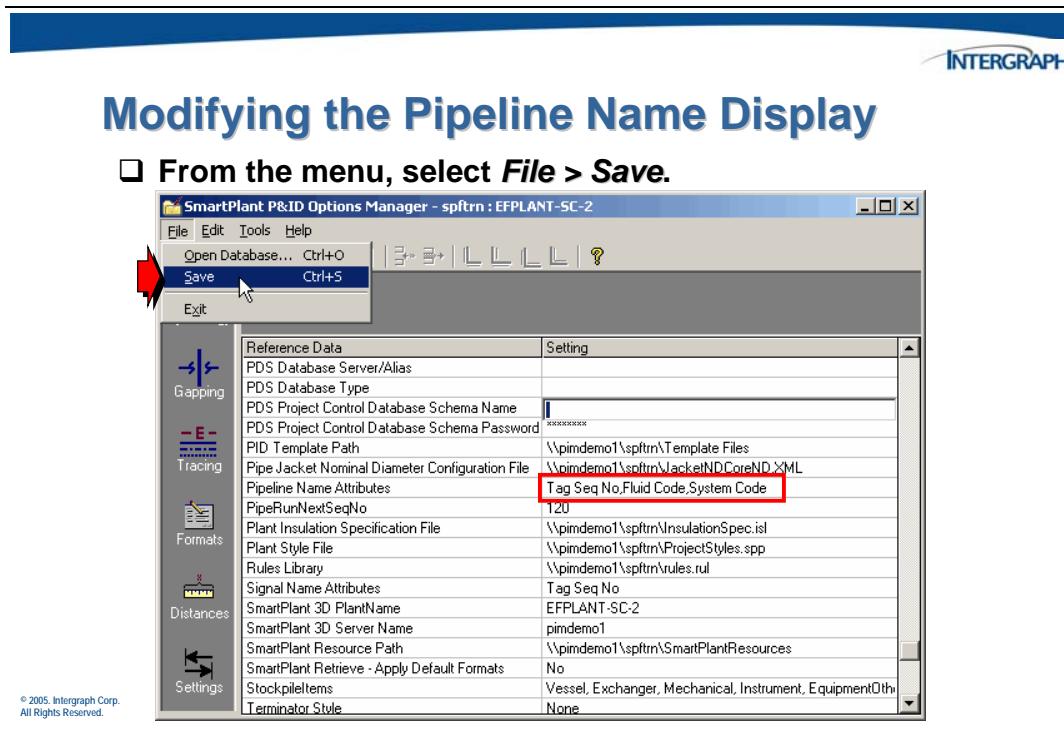
The *SmartPlant P&ID Options Manager* application window will display.



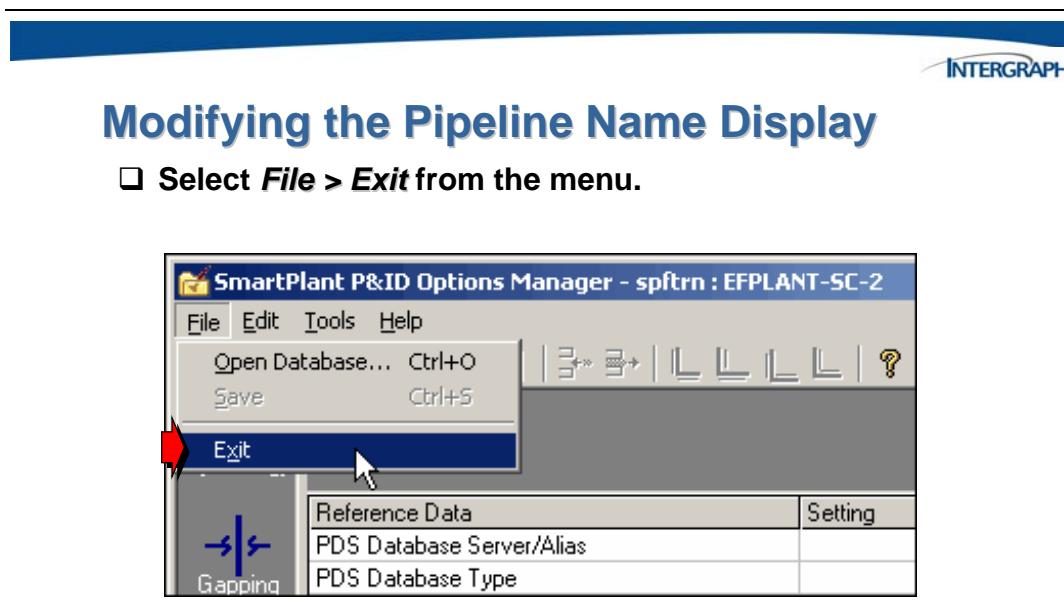
The *Settings* screen will display. The default properties (attributes) for the Pipeline Name consist of *Tag Seq No* and *Fluid Code*.



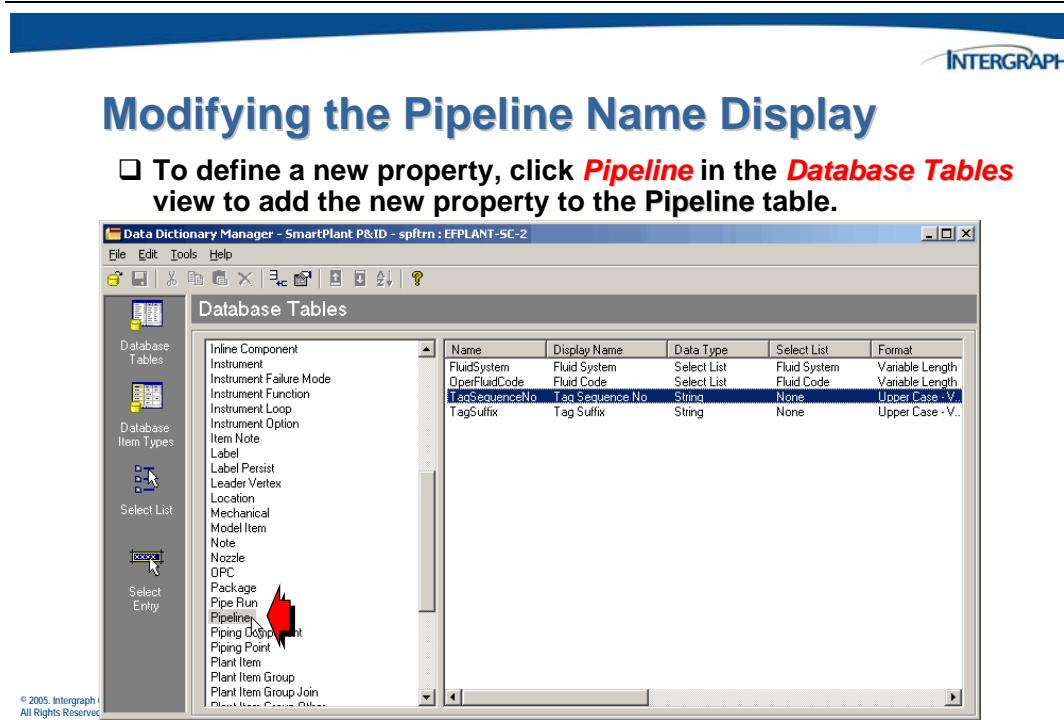
This will append the new property *System Code* as part of the Pipeline Name display within the SmartPlant Foundation Desktop Client.



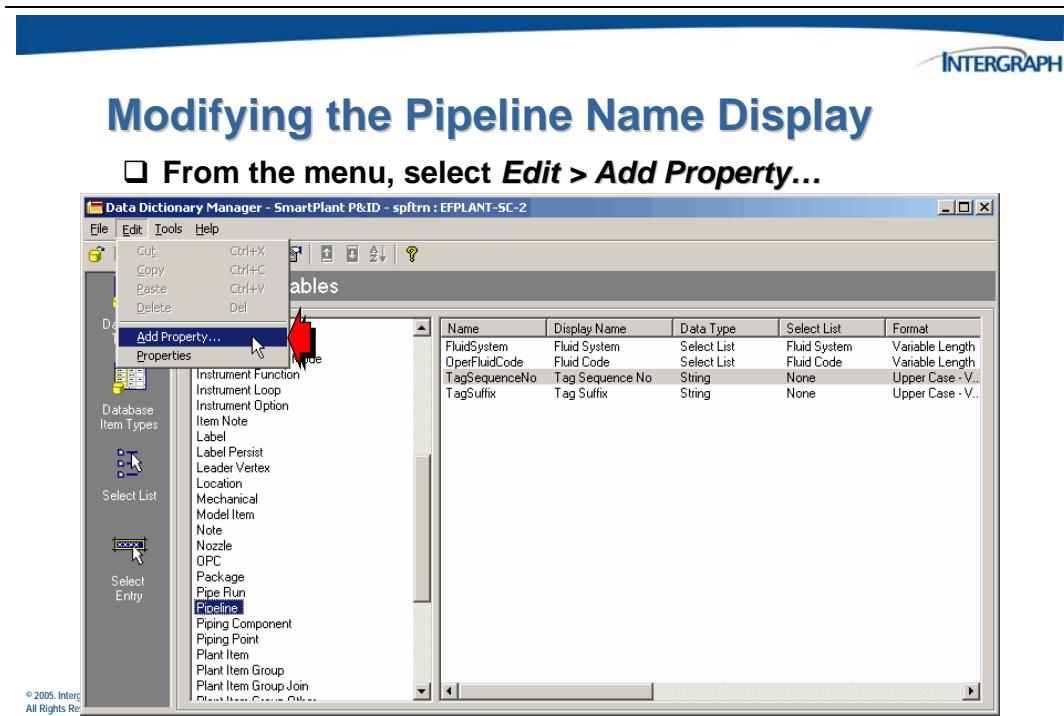
Once the changes have been saved, close the Options Manager.



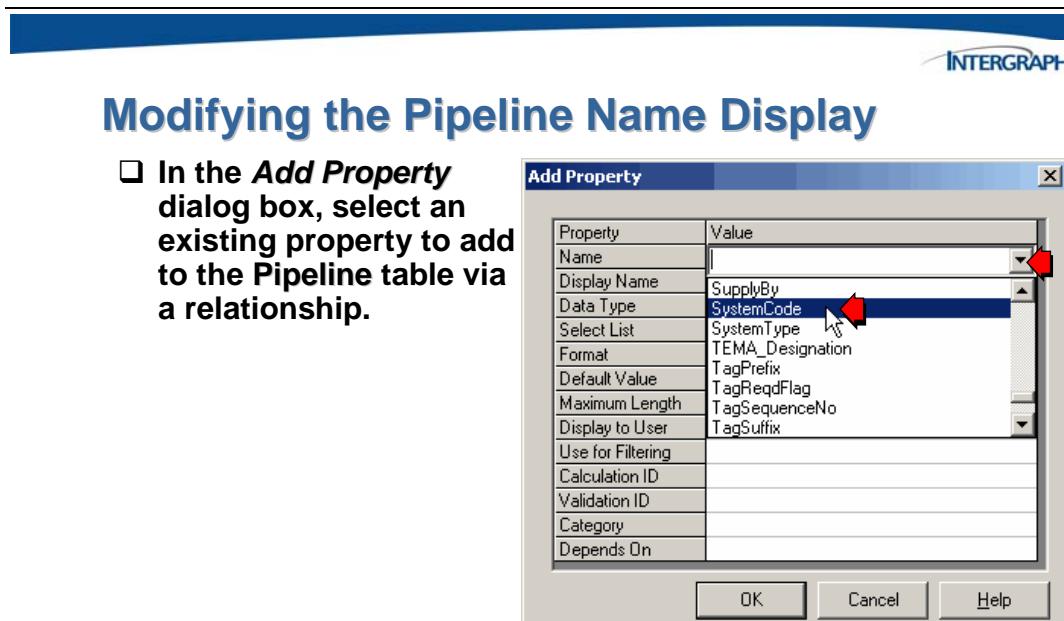
To add a new property to SPPID that is used by the Pipeline Name, begin by starting the **Data Dictionary Manager**.



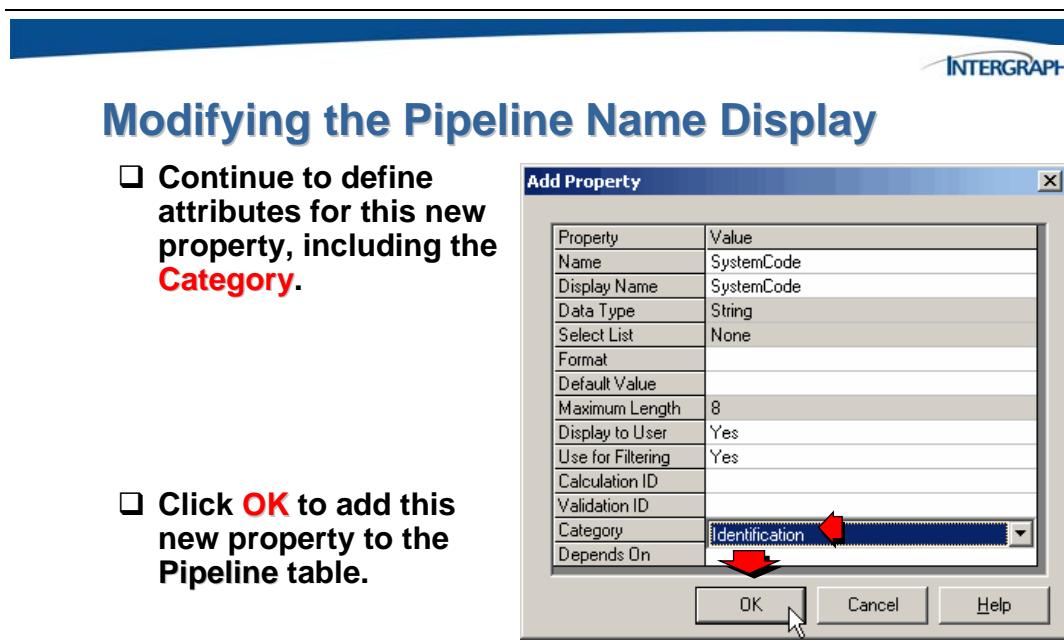
Edit the **Data Dictionary** to add this property to be available for the *ItemTag* entry for Pipeline. This will be the “Name” of the pipeline as contained in the published XML file.



This will associate the existing **SystemCode** property with the *Pipeline* table.



© 2005, Intergraph Corp.
All Rights Reserved.



© 2005, Intergraph Corp.
All Rights Reserved.

For this example when selecting **SystemCode**, it already exists as a *Plant Item* so we've opted to not actually create a new property but rather use the one that already exists.

Modifying the Pipeline Name Display

Since the **SystemCode property already exists in the Plant Item hierarchy, it will be used rather than a new property created.**

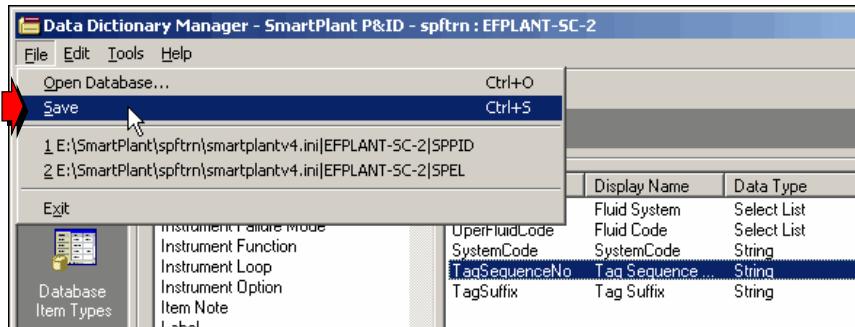


© 2005, Intergraph Corp.
All Rights Reserved.

When you click **Yes**, the list of properties for the *Plant Item* table updates with the new property. In this example we've added **SystemCode**.

Modifying the Pipeline Name Display

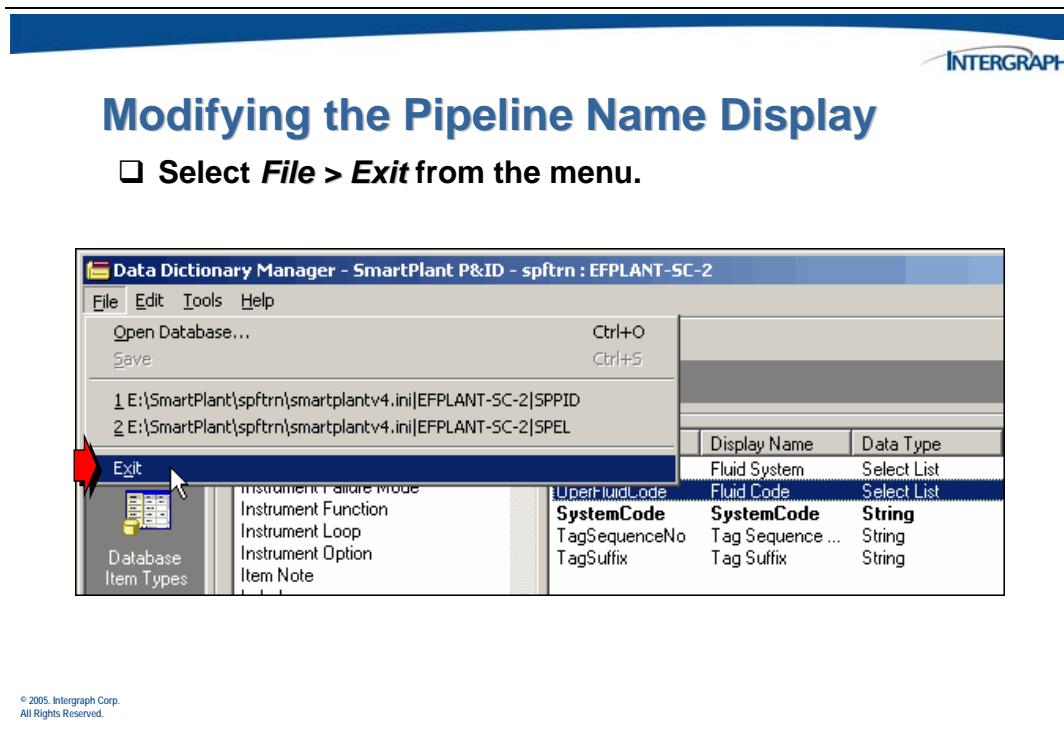
- From the menu, select **File > Save**.



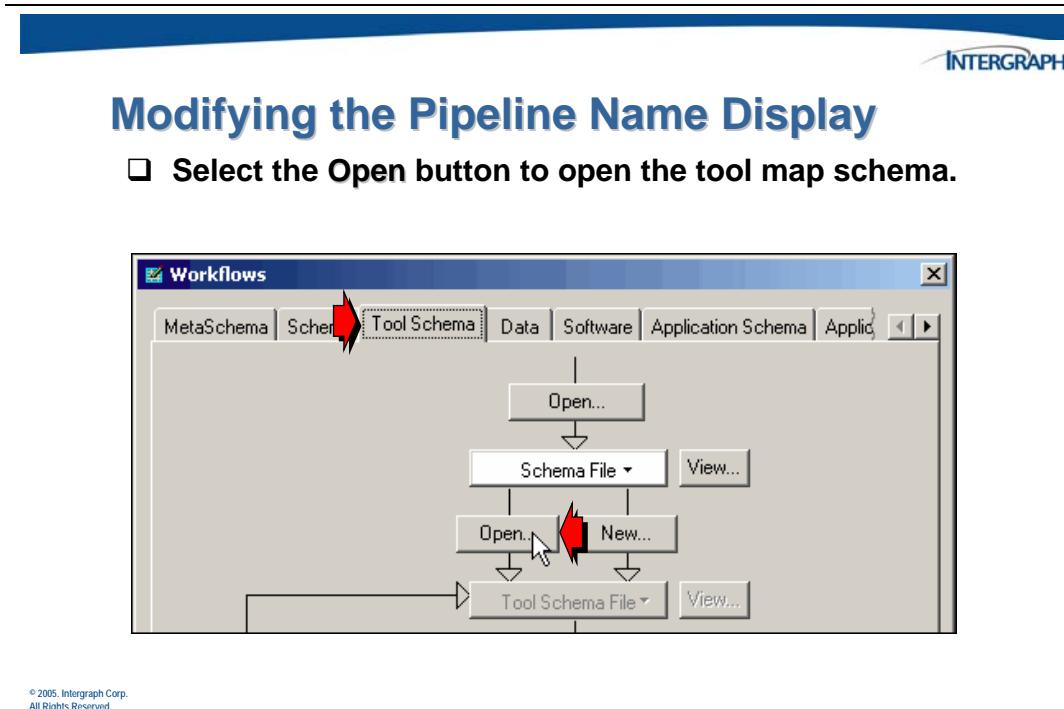
© 2005, Intergraph Corp.
All Rights Reserved.

After adding the property, save the changes to the SmartPlant P&ID meta data.

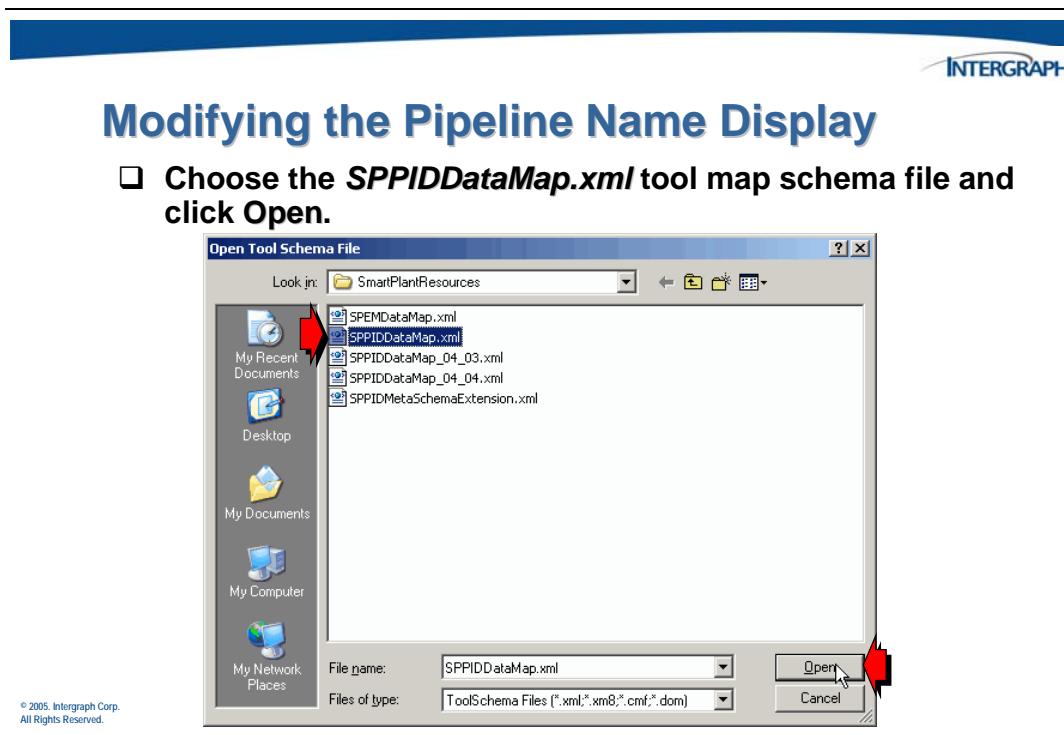
When you click **File > Exit**, this will close Data Dictionary Manager.



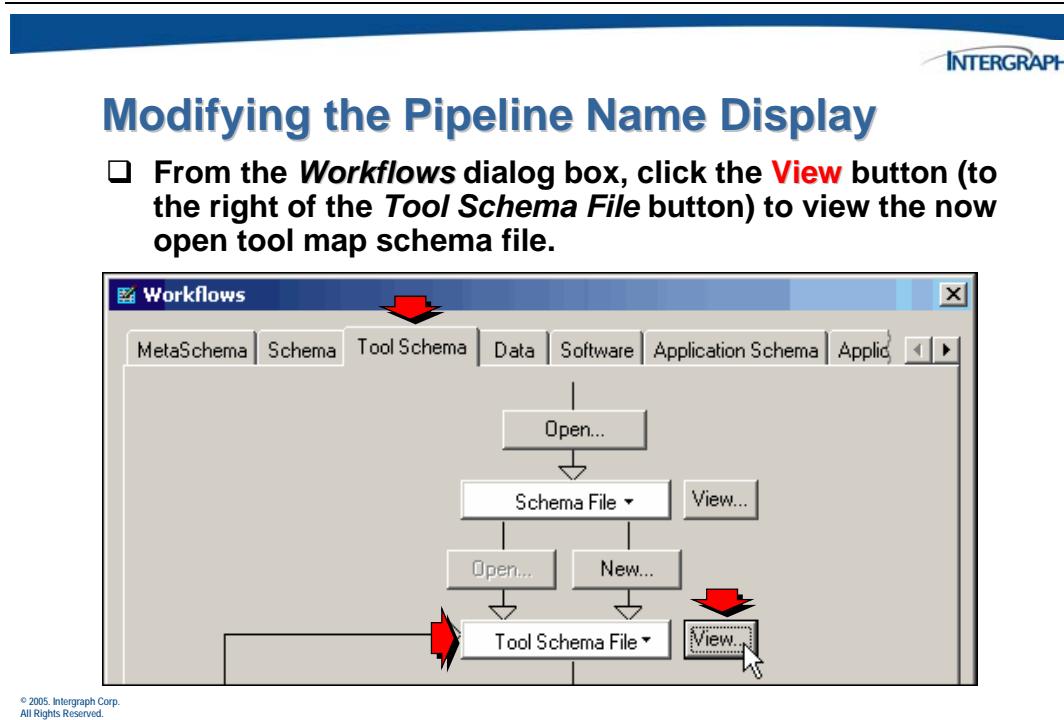
The tool map schema must be able to read the SmartPlant schema file during modifications so you must first open the *EFSchema.xml* file using the config file.



The *Open Tool Schema File* dialog displays.



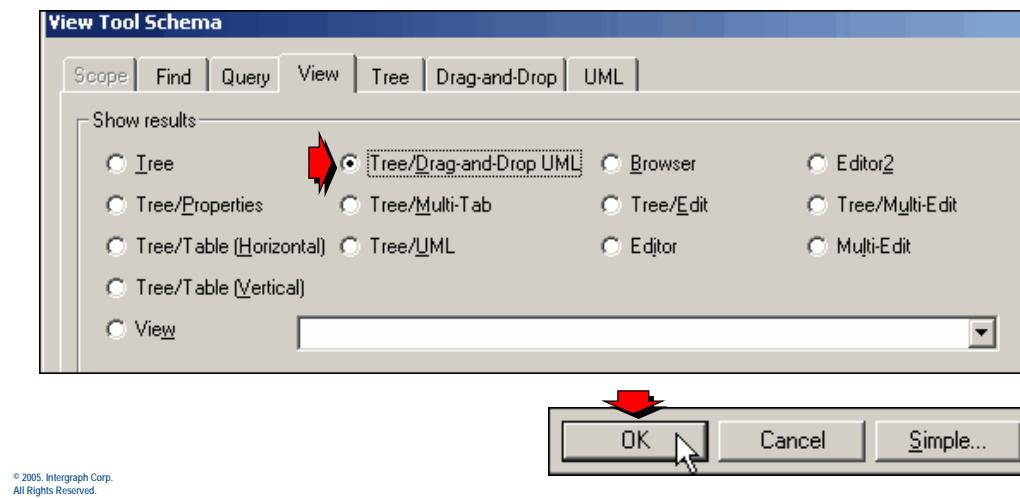
The tool map schema, which is delivered during the installation of the authoring tool, will be opened.





Modifying the Pipeline Name Display

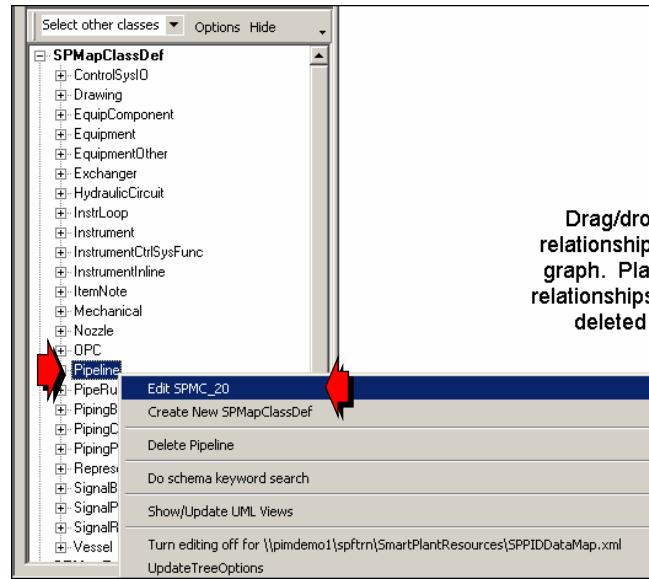
- In the **View Data** dialog, click the view type to use and then click **OK**.



Modify the *SPPIDDataMap.xml* to add **SystemCode** to the published properties for a Pipeline.



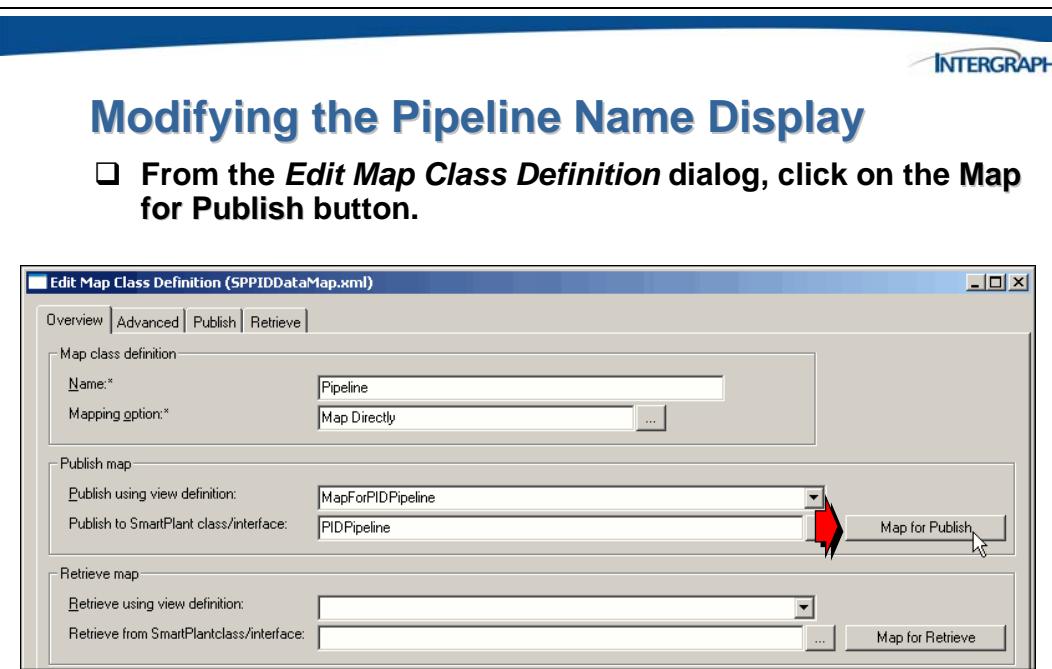
Modifying the Pipeline Name Display



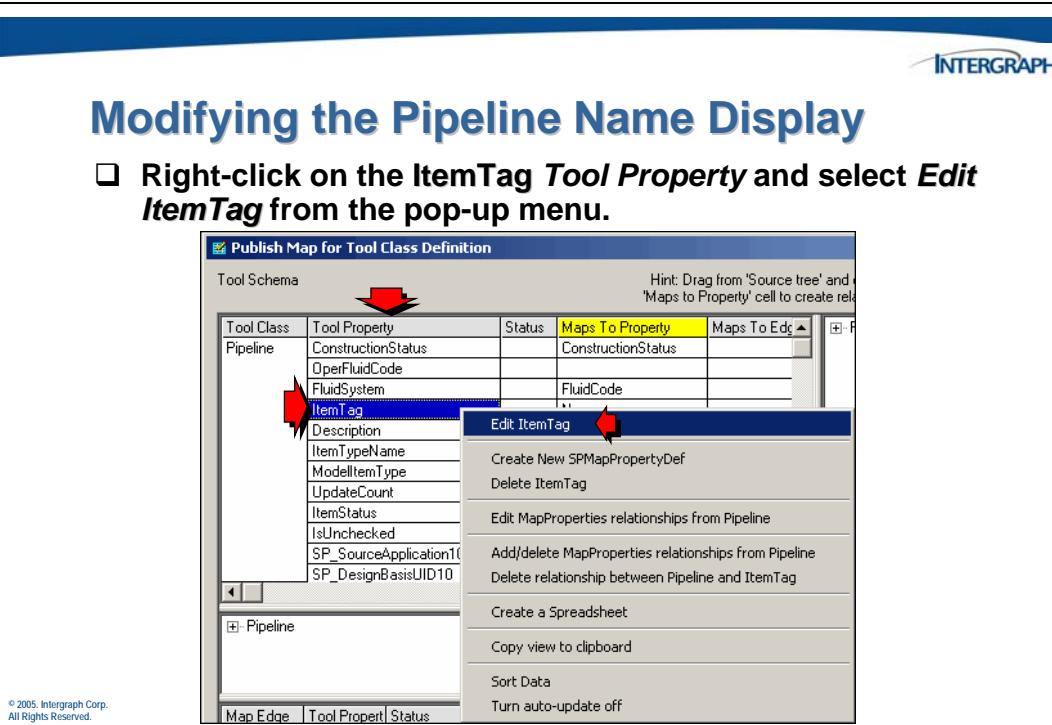
- Right-click on Pipeline and then select **Edit SPMC_20** from the right mouse menu.

Drag/drop relationships graph. Place relationships deleted (

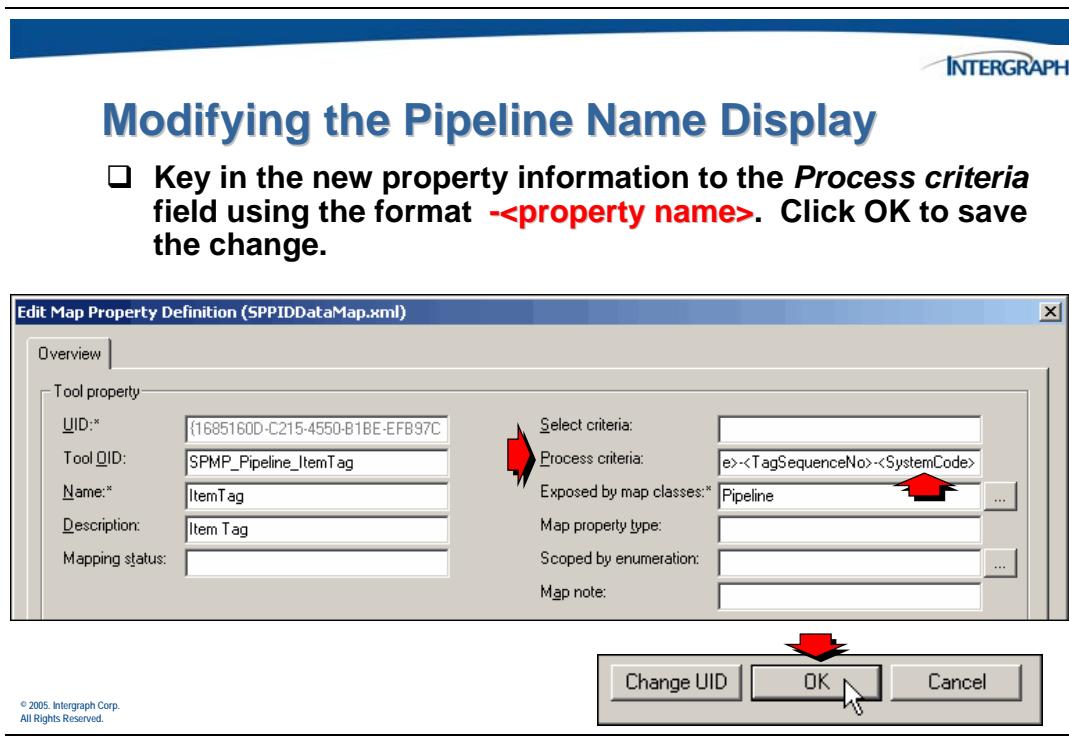
The *Edit Map Class Definition* dialog will be displayed.



Add the new property, **SystemCode**, to the ItemTag *Process criteria*.



The *Edit Map Property Definition* dialog will be displayed.



This will add the new custom property *SystemCode* to the **ItemTag** display along with *OperFluidCode* and *TagSequenceNo*. These three properties will display in the SPF Desktop Client as the Pipeline **Name**. The *Process criteria* field contents are described below.

If the data type of the SmartPlant P&ID property does not match the data type of the mapped SmartPlant property, a data type mismatch exists. In some cases, a property with a data type mismatch can still be published and retrieved using specialized code. The code to perform these data type conversions and other special handling is a Property Conversion object by putting the *ProgID* of the Property Conversion class in the **Process Criteria** option of the MapProperty. The ProgID for each of these classes is formed by prefixing the class name with the string EFAdapter.

The list of available Property Conversion classes is shown below:

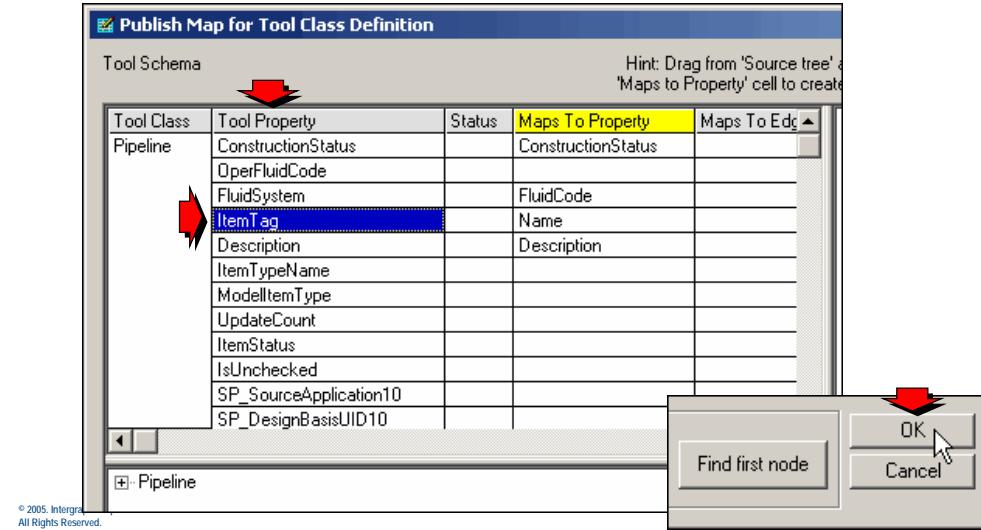
Class Name	Description
BoolToEnumConv	This property conversion supports the conversion needed to publish and retrieve the SmartPlant P&ID SP_IsSpecialtyItem property to/from the CommoditySpecialtyType property in SmartPlant.
CompFlowDirConv	This class only publishes the special property called <i>IsFlowDirectional</i> . There is no property named IsFlowDirectional in SmartPlant P&ID. This information is derived from the Flow Direction property value. If the

	Flow Direction on a component is set, then this will publish <i>IsFlowDirection</i> as True ; otherwise, it is published as False .
EnumConversions	This class implements the property conversion code for Hierarchical Enumerated Lists.
EnumToBoolConv	This class implements the property conversion code between Enumerated to Boolean. This property conversion module specifically targeted to handle the <i>StressAnalysisRequirement</i> property in SmartPlant P&ID.
EnumToStringConv	This class implements the property conversion code between Enumerated List to String.
NominalDiameterConv	Converts Nominal Diameter from an Enumerated List to a UoM during publish and from a UoM to Enumerated List during retrieve.
PortFlowDirectionConv	Special property conversion to handle the publishing of <i>FlowDirection</i> property on Connect Point. The Select List values for <i>FlowDirection</i> for SmartPlant P&ID Piping Point and SmartPlant PIDPipingPort do not match well. SmartPlant P&ID uses only one Select List for <i>FlowDirection</i> for both InlineComponent and PipingPoint whereas SmartPlant uses two separate Select Lists.
PpConnNameConv	This class implements the property conversion code for Piping Connector Name property.
TagPrefixConv	This class implements the property conversion code for <i>TagPrefix</i> property. It populates the Unit number value.
UoMtoDoubleConv	This class implements the property conversion code between UoM and Double. This property conversion module is specifically targeted to handle conversion of unitless attributes (such as Molecular Weight, CpCvRatio, Compressibility, SpecificGravity). In SmartPlant P&ID these attributes are treated like UoM attributes. However, in the SmartPlant schema they are simply Double.
UoMtoEnumConv	This class implements the property conversion code between UoM to Enumerated List.



Modifying the Pipeline Name Display

- Click OK to save the changes to the ItemTag Tool Property.

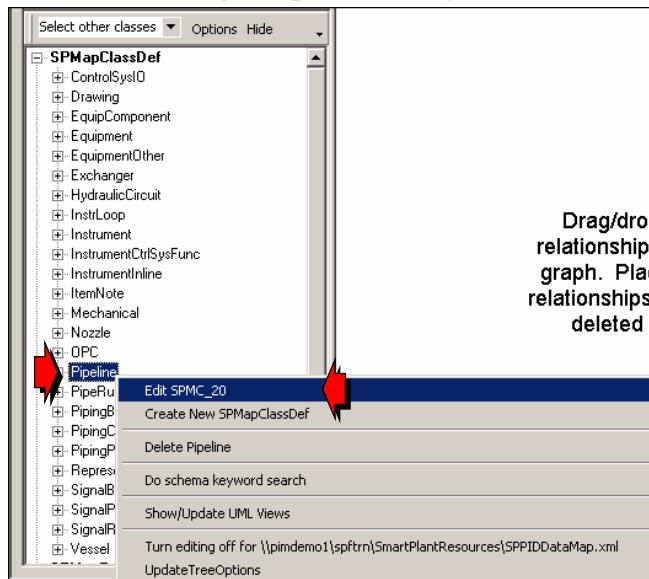


To review the **ItemTag Process Property Criteria**, use the edit command to display the *Publish Map for Tool Class Definition* dialog once more.

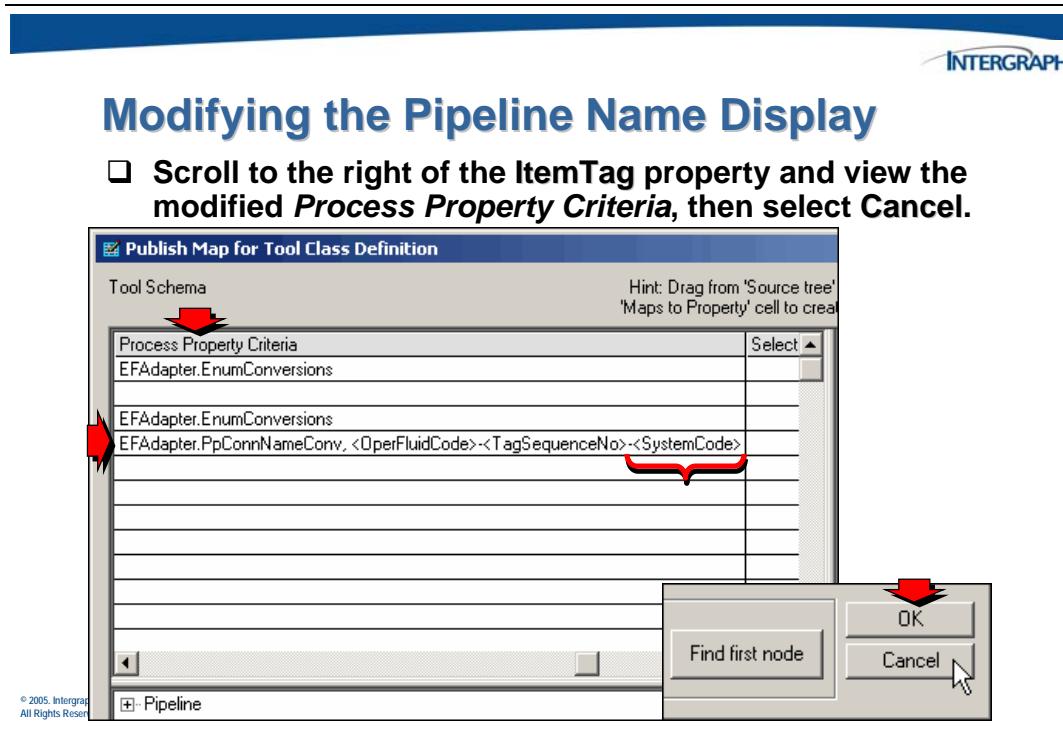


Modifying the Pipeline Name Display

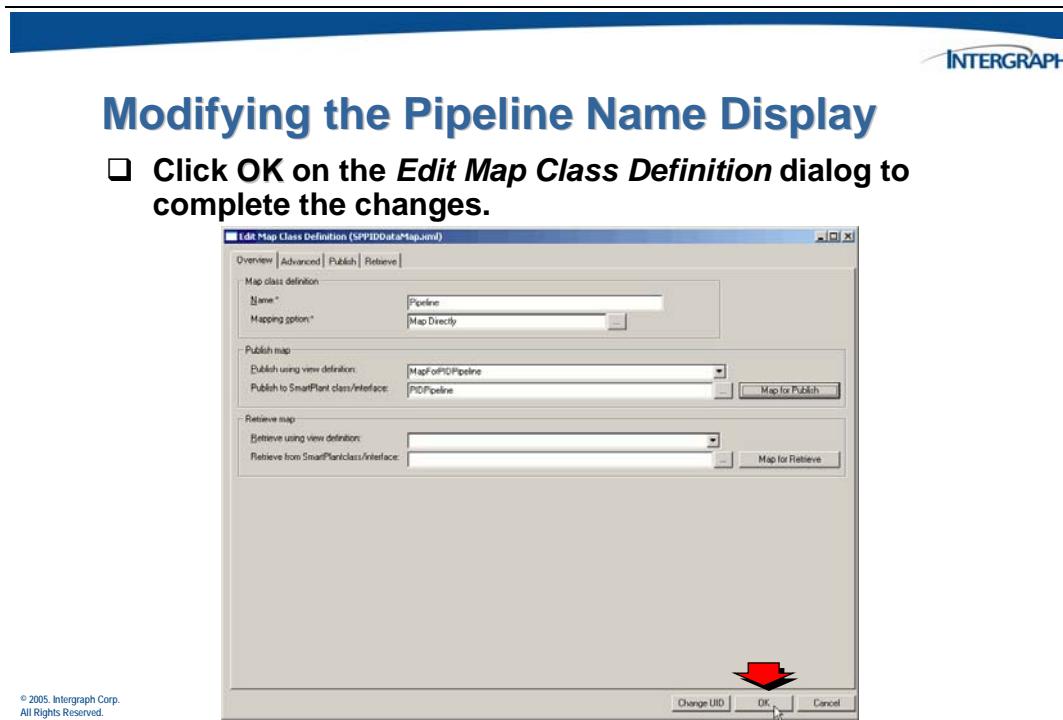
- Right-click on Pipeline and select **Edit SPMC_20** from the menu to verify the ItemTag change.



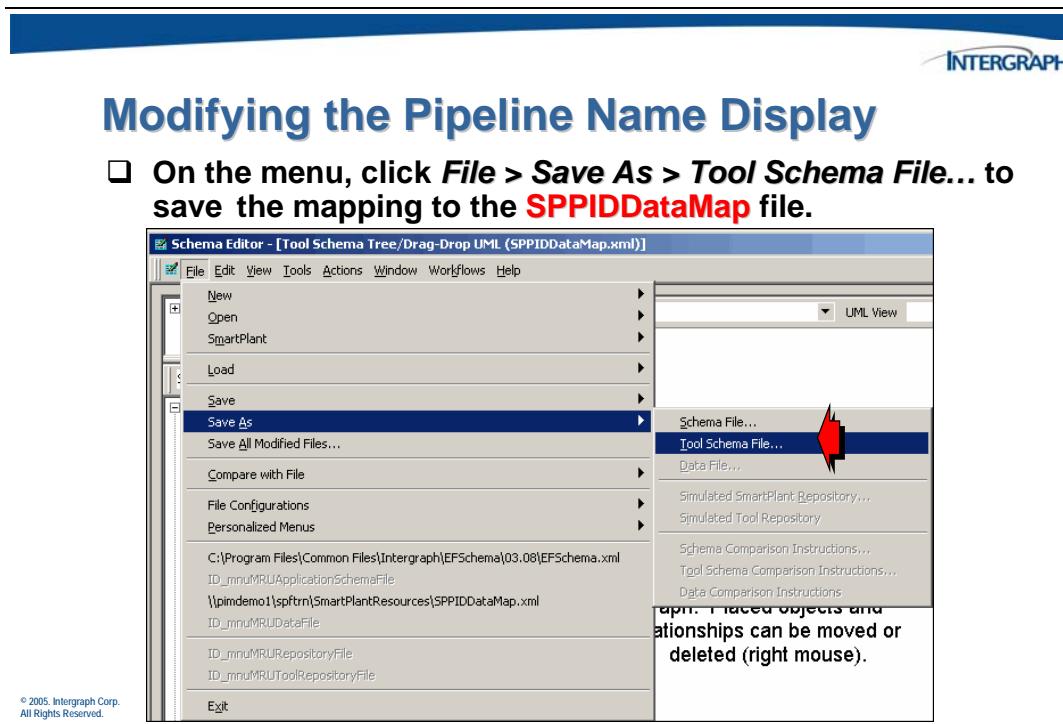
Note in the following example that the criteria now includes the **SystemCode** property.



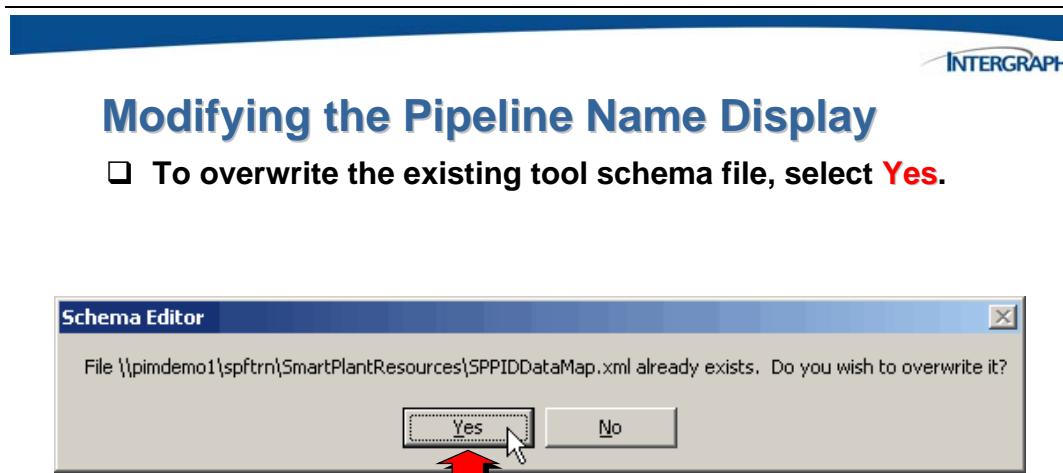
Since no changes were made to the *Publish Map for Tool Class Definition*, just cancel out of the dialog.



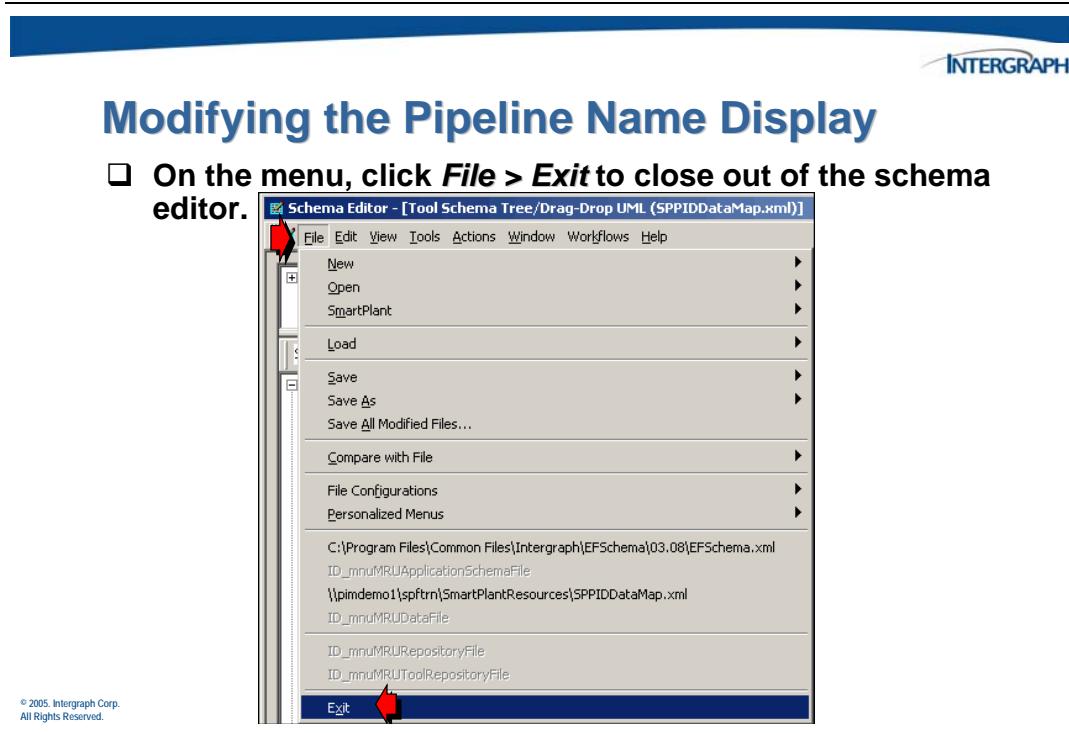
This will close the dialog and now the changed criteria can be saved to the tool map schema file.



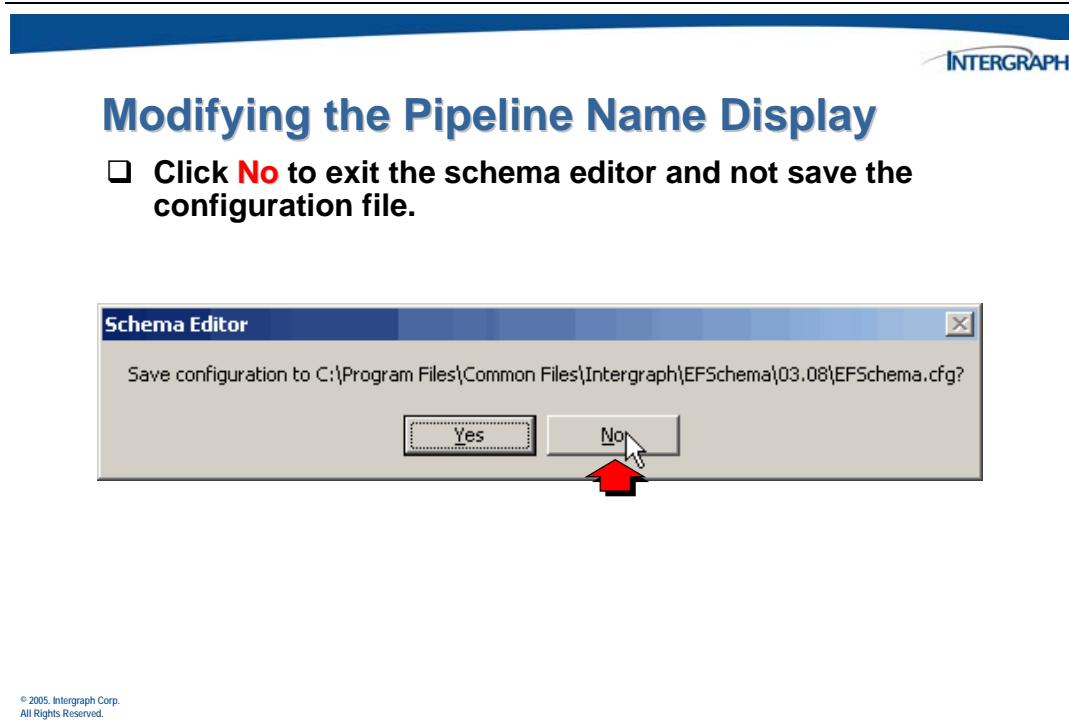
This will create an updated *SPPIDDataMap.xml* tool map schema file.



Now that the schema changes have been made, close out of the schema editor



No changes have been made to the schema configuration contents so no save is needed.

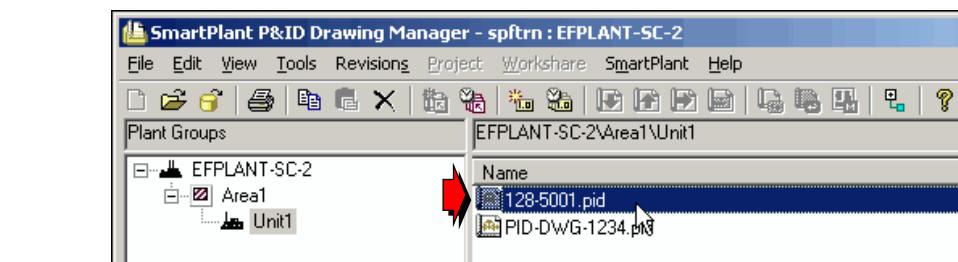


To see the changes, open *Drawing Manager* in order to publish a new version of a P&ID drawing.



Modifying the Pipeline Name Display

- Double-click on the drawing to be opened.



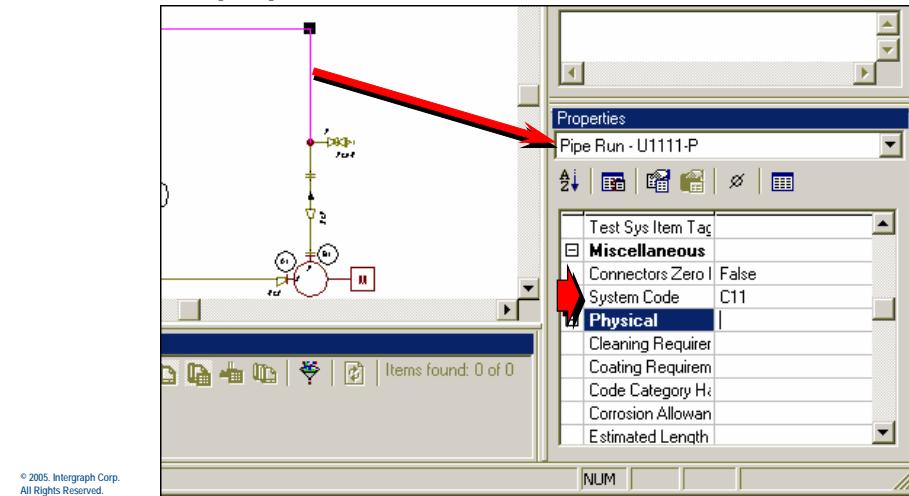
© 2005, Intergraph Corp.
All Rights Reserved.

Locate a *PipeRun* and make sure the **System Code** property has a value.



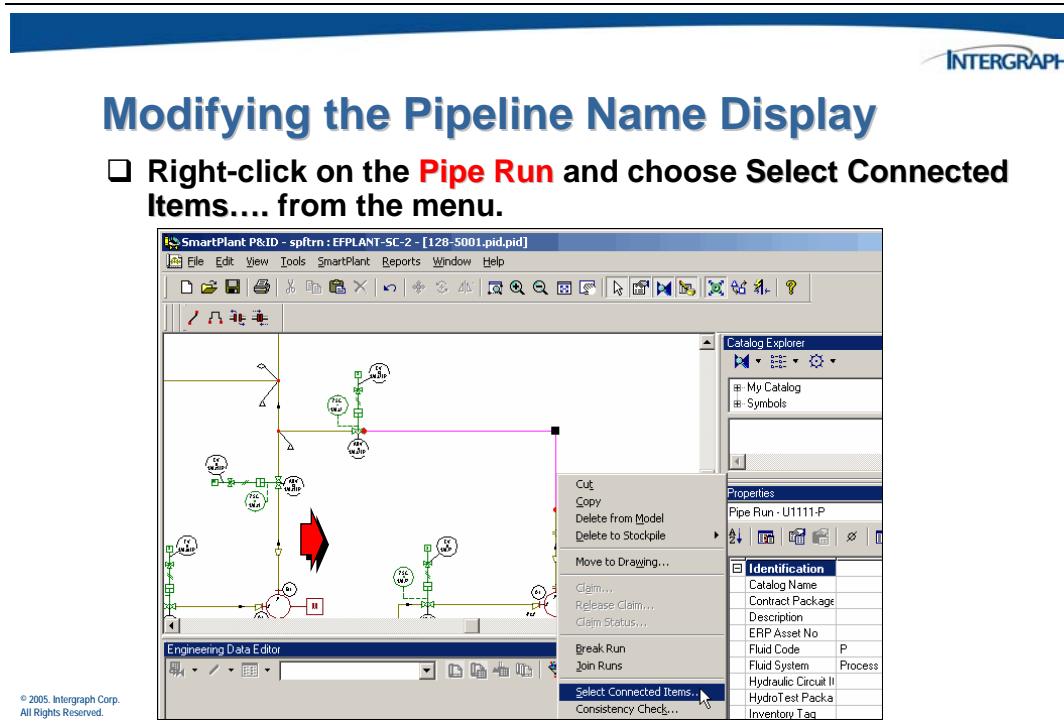
Modifying the Pipeline Name Display

- Click on one of the **Pipe Run's** and modify the one of the new properties.

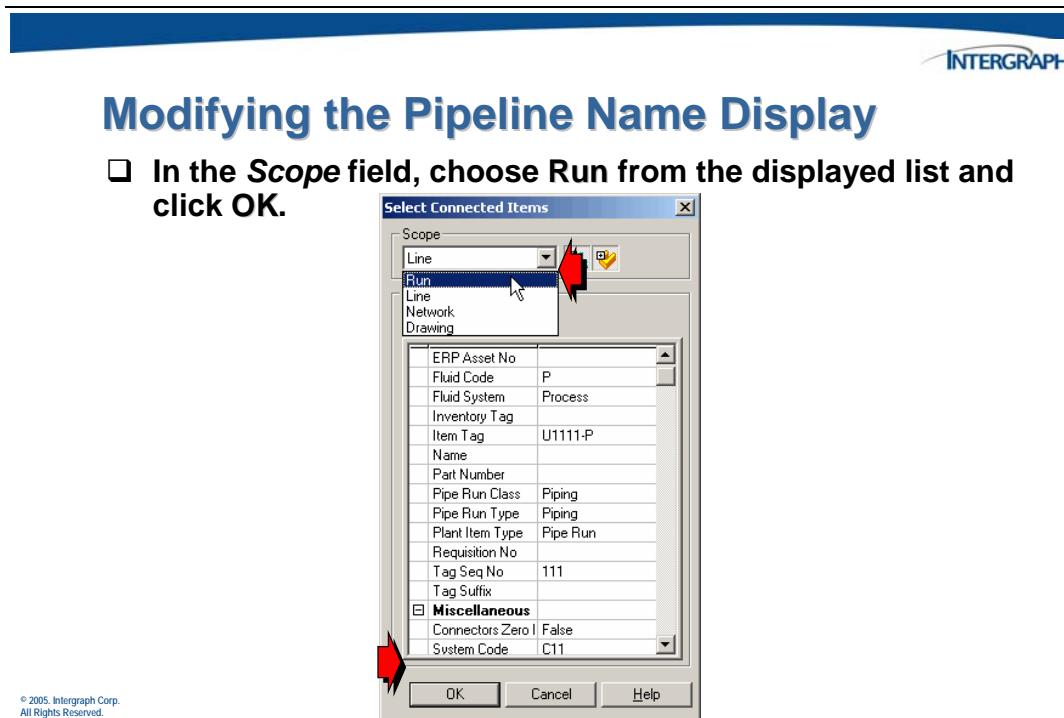


© 2005, Intergraph Corp.
All Rights Reserved.

Highlight and display the PipeRun in SmartPlant P&ID once the *System Code* has been updated.



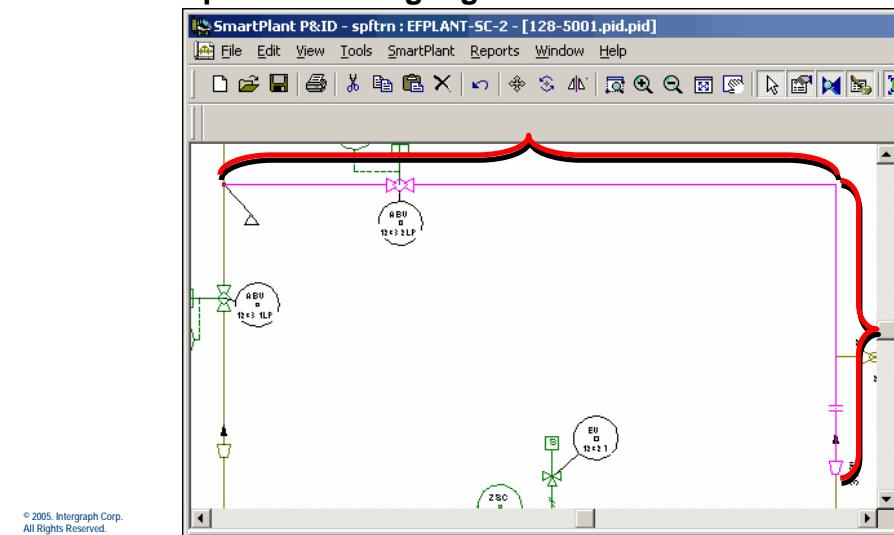
The *Select Connected Items* dialog will display.





Modifying the Pipeline Name Display

The Pipe Run will highlight in the view window.

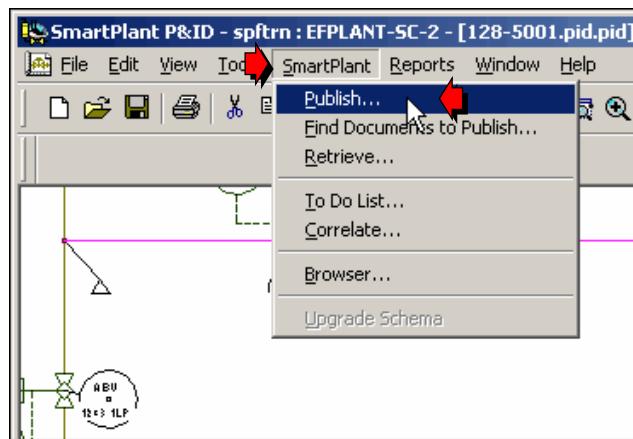


Now, publish this P&ID drawing to see the new PipeLine “Name” format in SPF.

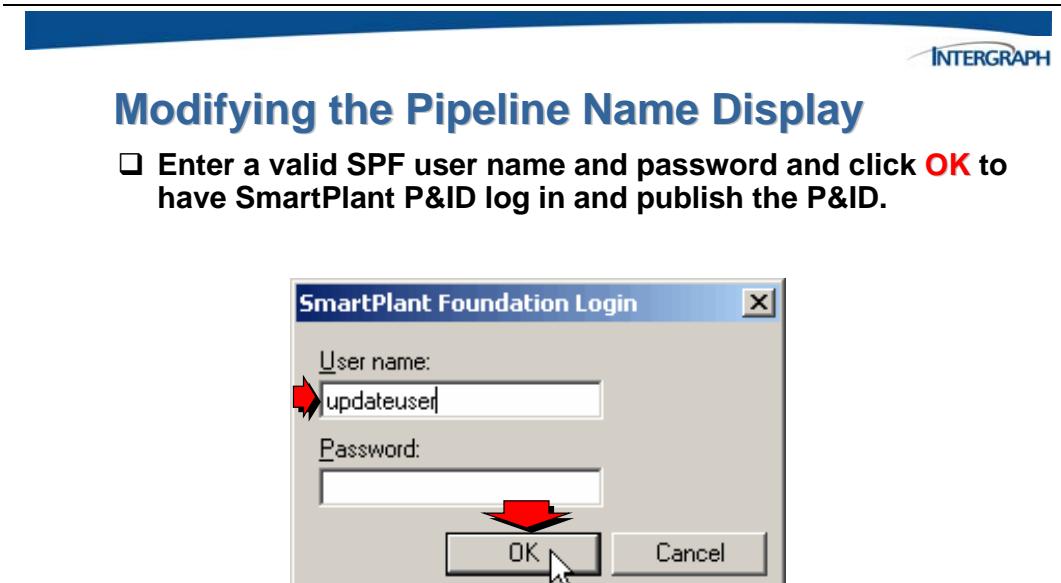


Modifying the Pipeline Name Display

- Select the **SmartPlant > Publish..** menu command.

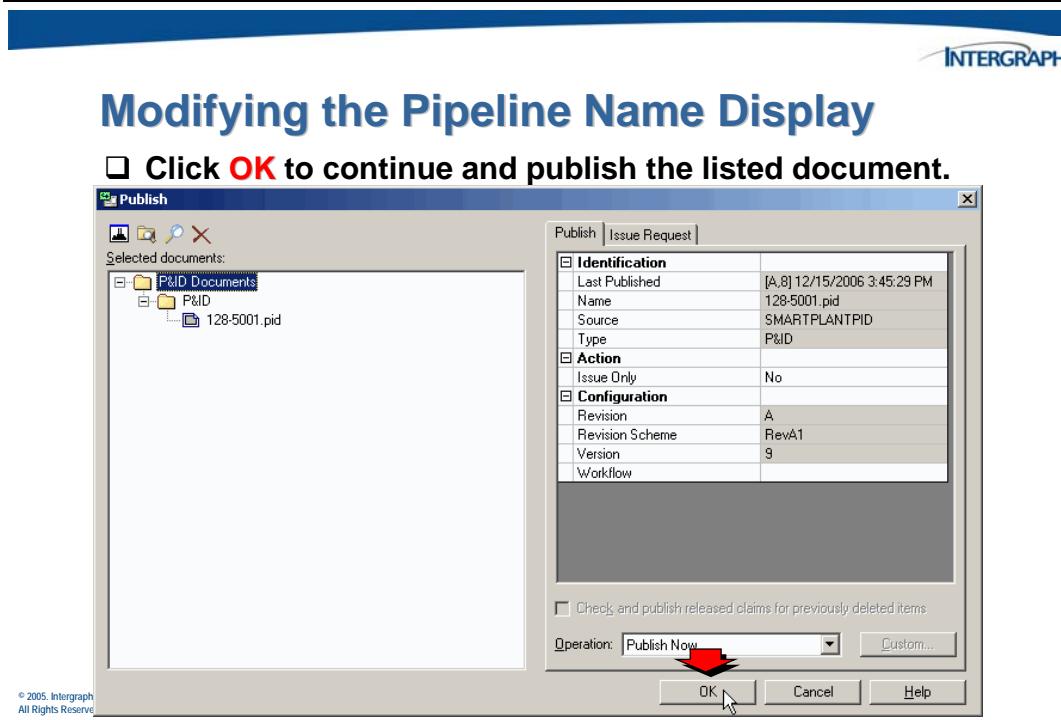


The *SmartPlant Foundation Login* dialog display allowing you to connect to the SPF server.

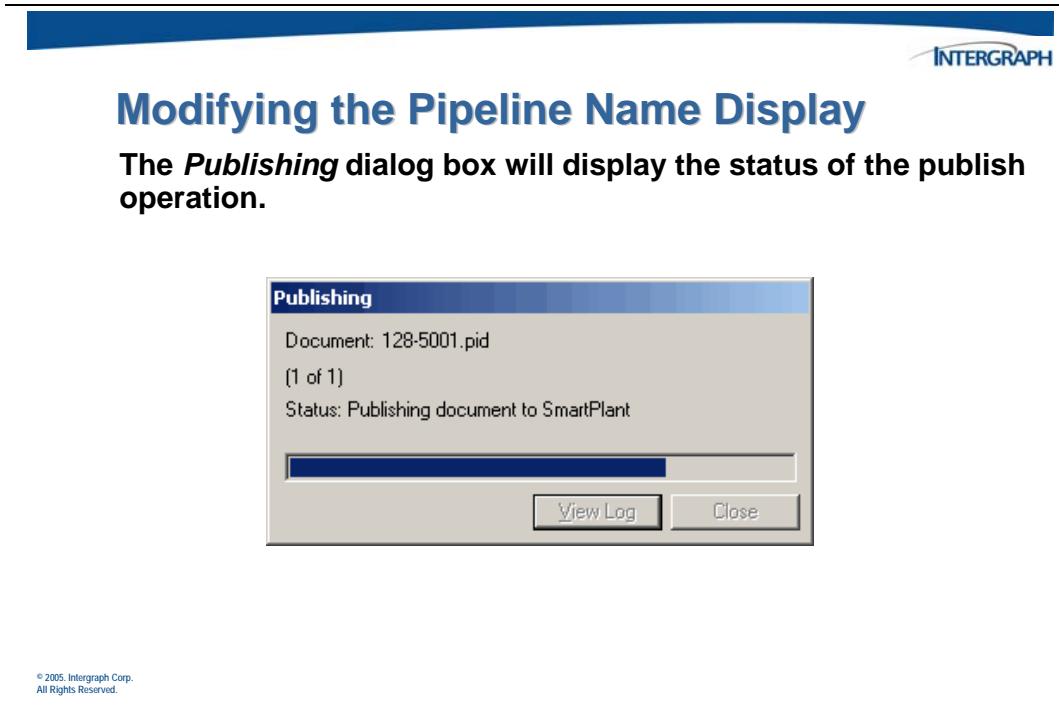


© 2005, Intergraph Corp.
All Rights Reserved.

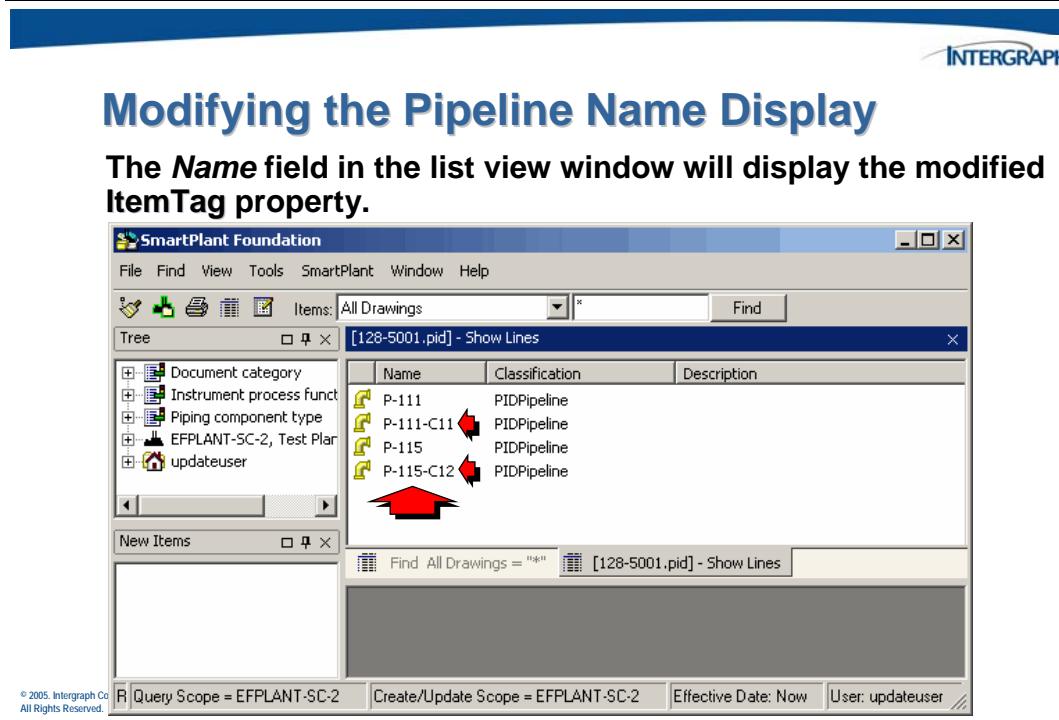
The *Publish* dialog will display showing the next version of the P&ID to be generated as a result of this publish operation.



The publish will generate a new xml file containing the objects to be instantiated in the SPF user database. Once this file exists it will be moved to the configured SPF Vault.



Perform a search for the P&ID Drawing and then use the Show Lines menu command to see the related Pipelines using the new “Name” format.



A P P E N D I X

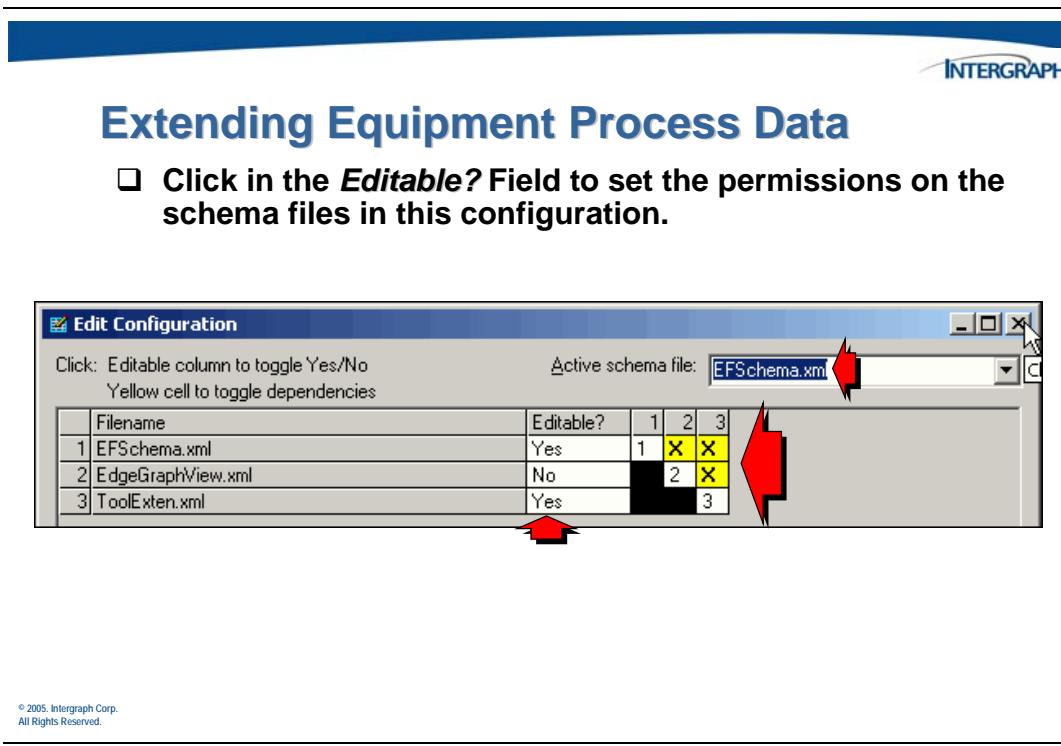
D

Extending Equipment Process Data

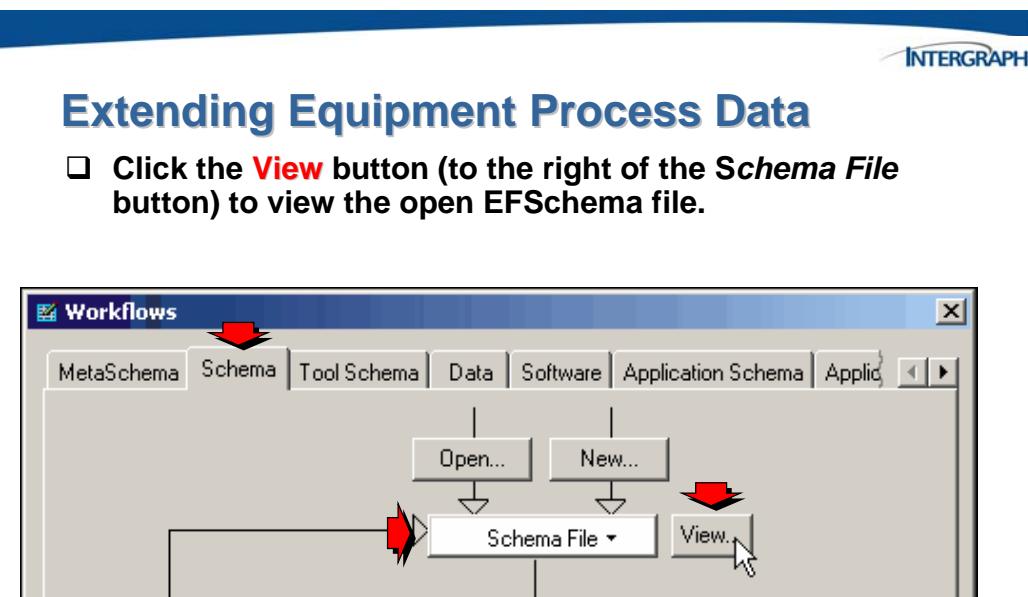
D. Extending Equipment Process Data

Another common example that could be used with SmartPlant P&ID data is the requirement to use site specific properties for Equipment Process Data. This Equipment Process Data then needs to be published to SmartPlant Foundation.

The properties that will be used already exist in the SmartPlant schema but some additional relationships will need to be added. In order to do this, the EFSchema will need to be editable. Modify the configuration (EFSchema.cfg) and set the correct *Active schema file* and toggle the *Editable?* field to the correct settings as shown below.

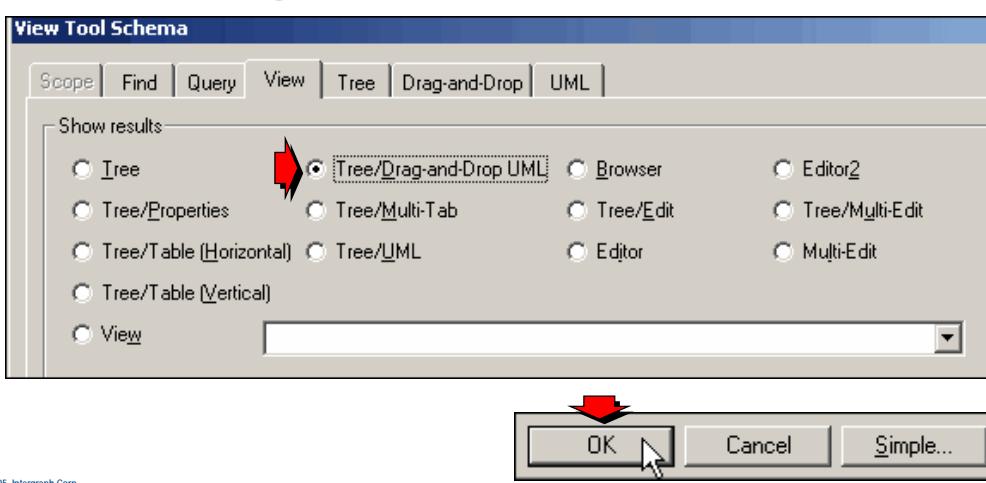


Next, re-open the schema editor using the modified configuration file, EFSchema.cfg.



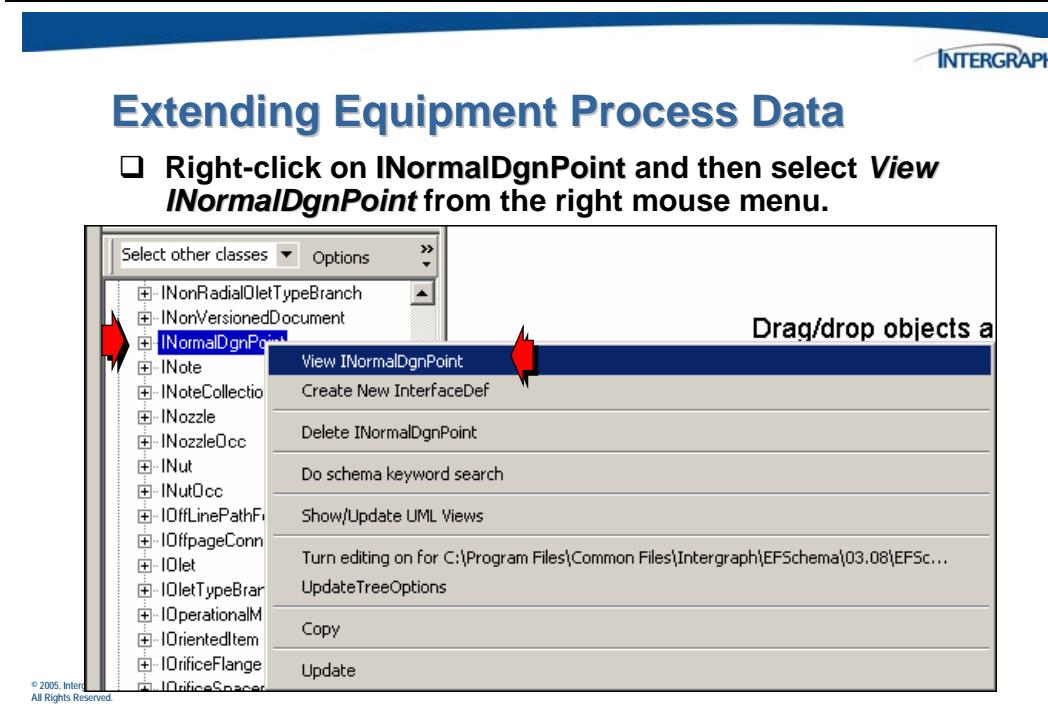
© 2005, Intergraph Corp.
All Rights Reserved.

To view and edit the master schema, use the **Tree/Drag-and-Drop UML** view.

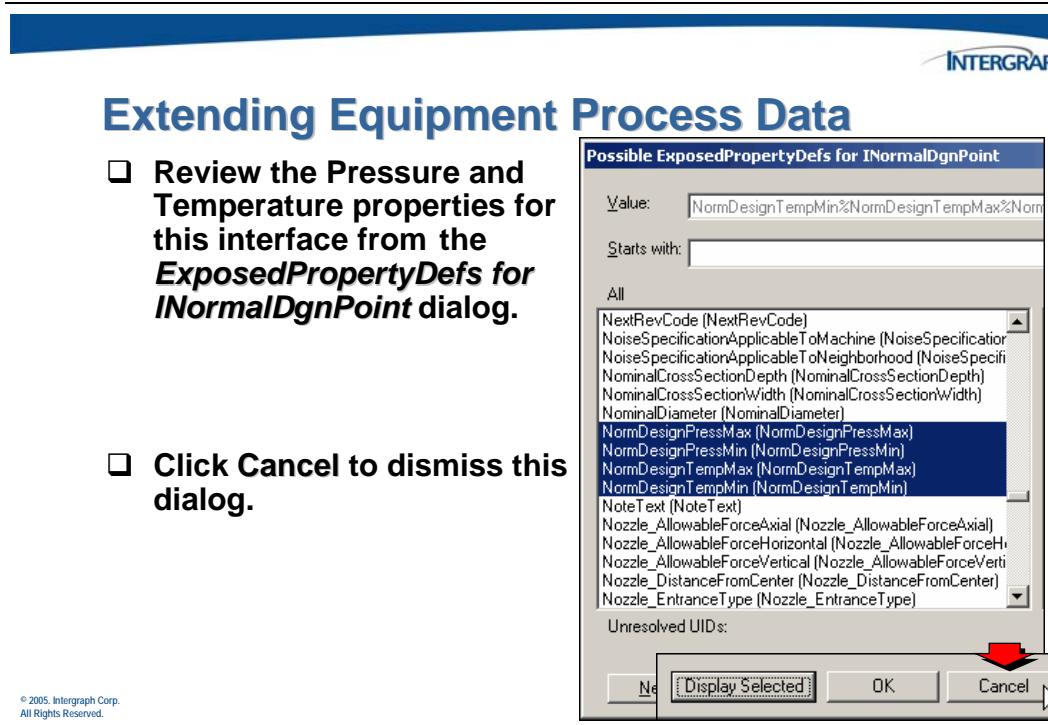


© 2005, Intergraph Corp.
All Rights Reserved.

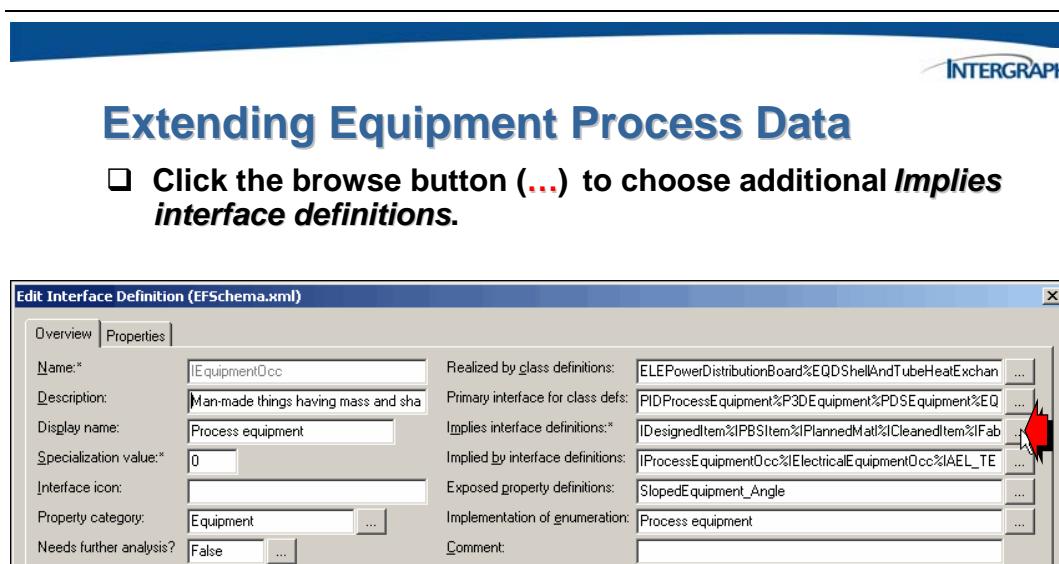
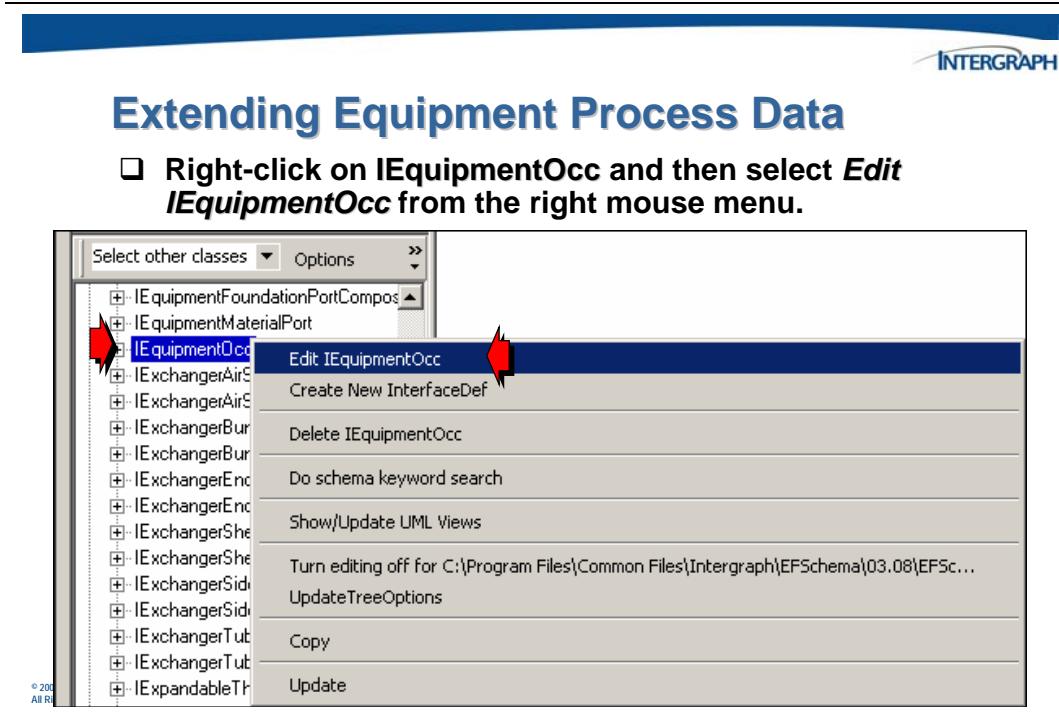
Review the schema and locate the existing properties to be used during the Equipment Process Data publish.



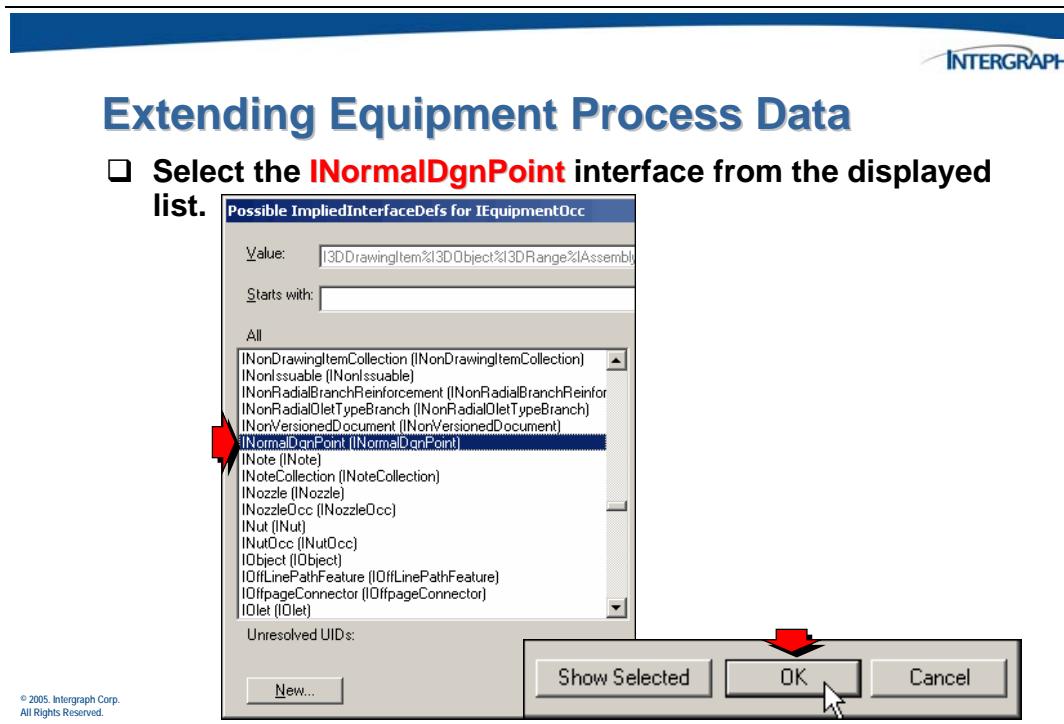
The following example uses the INormalDgnPoint interface since it contains the properties *NormDesignPressMin* and *NormDesignTempMin*.



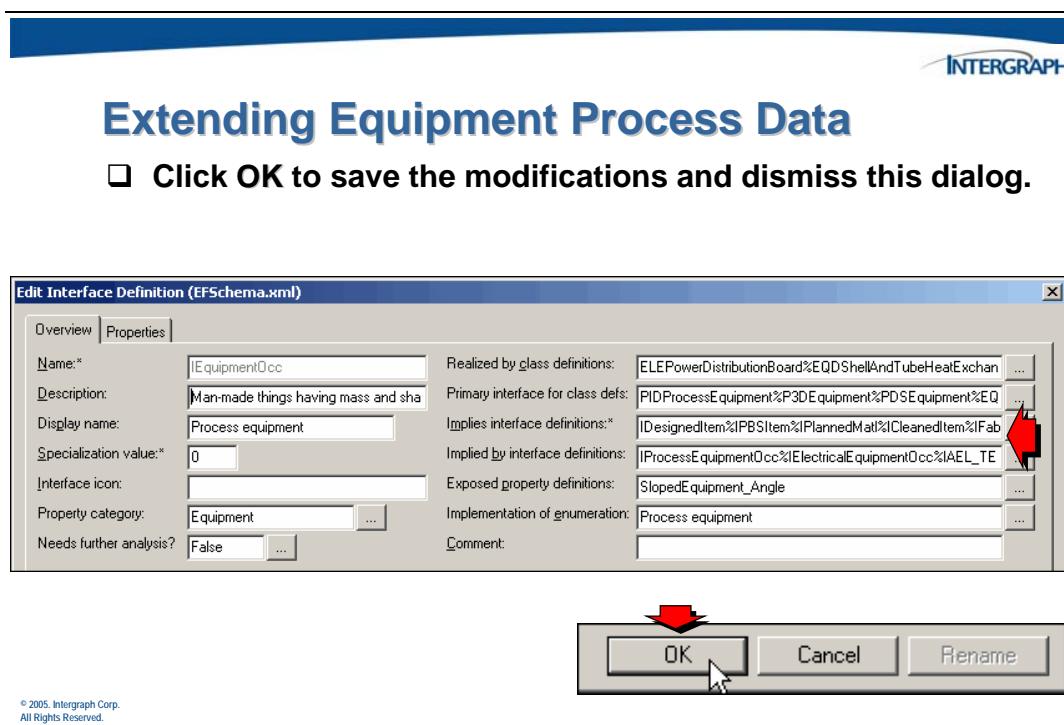
To expose these properties to an Equipment class, the `IEquipmentOcc` interface will need to imply (relate) the interface containing the `NormDesignPressMin` and `NormDesignTempMin` properties.



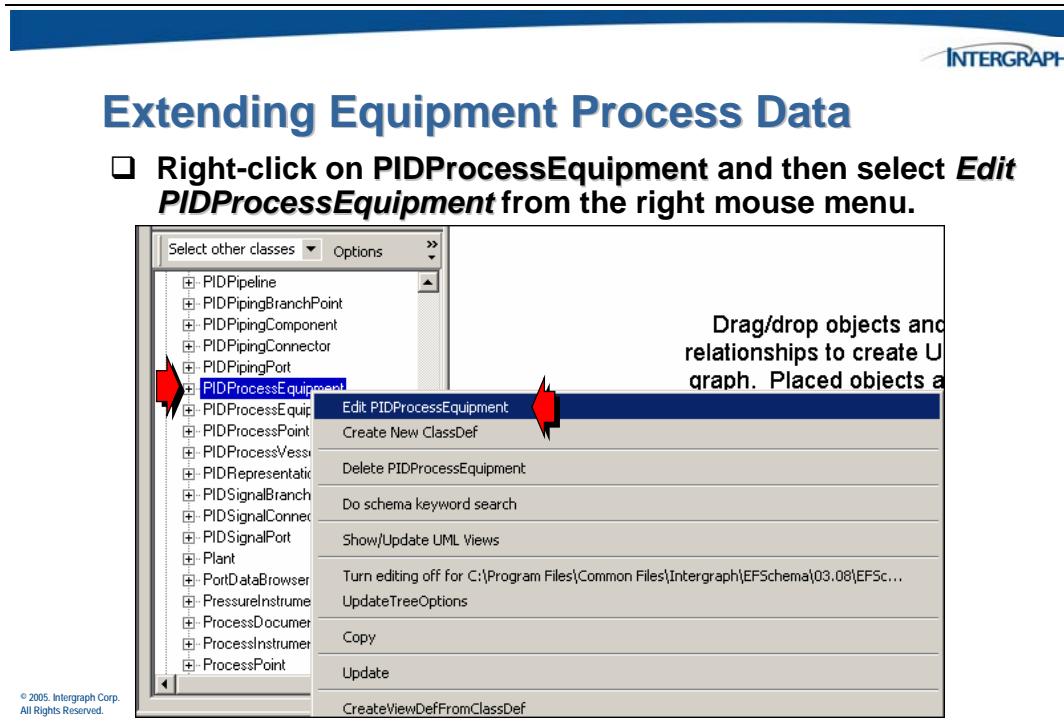
Remember, **INormalDgnPoint** exposes the properties *NormDesignPressMin* and *NormDesignTempMin*.



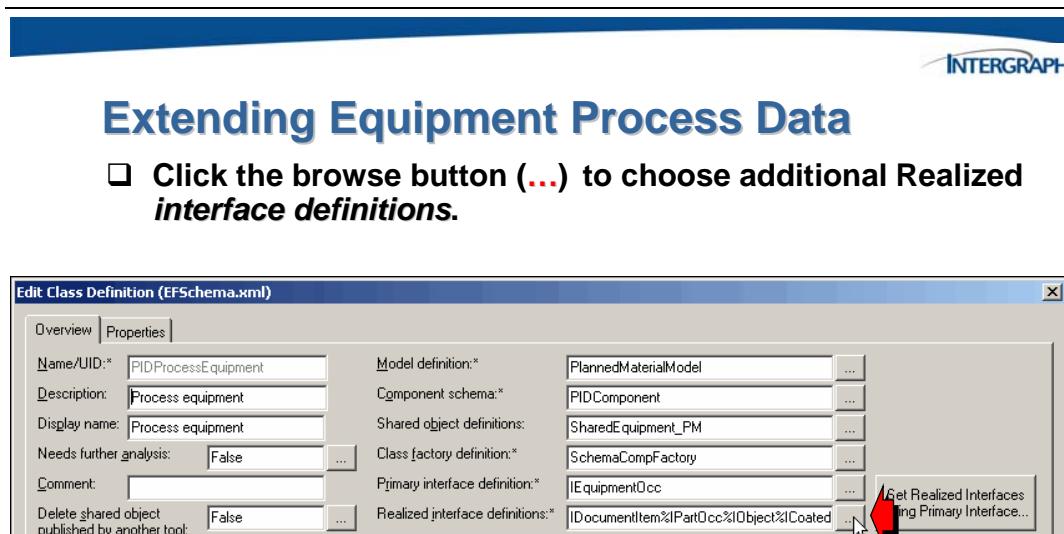
Once the implies relationship has been created, save this change.



Since in the **INormalDgnPoint** is implied by **IEquipmentOcc**, the **PIDProcessEquipment** class definition must also realize **INormalDgnPoint**.



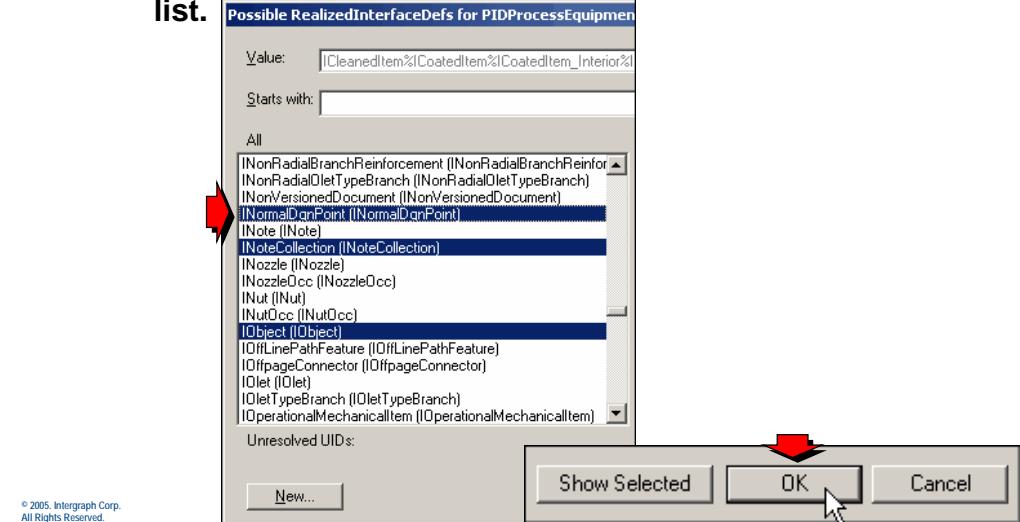
Edit the **PIDProcessEquipment** class def and add a realizes relationship to **INormalDgnPoint**.





Extending Equipment Process Data

- Select the **INormalDgnPoint** interface from the displayed list.

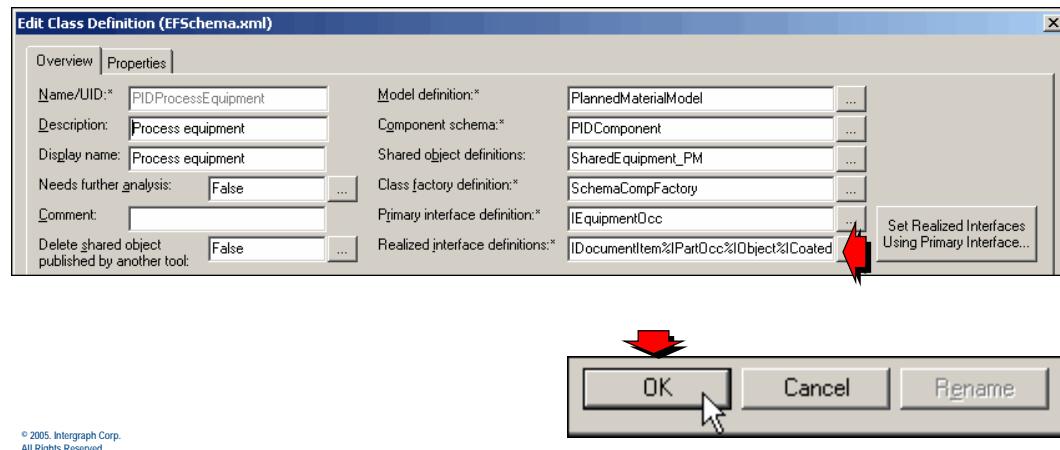


Now that the realizes relationship has been created, save this change.

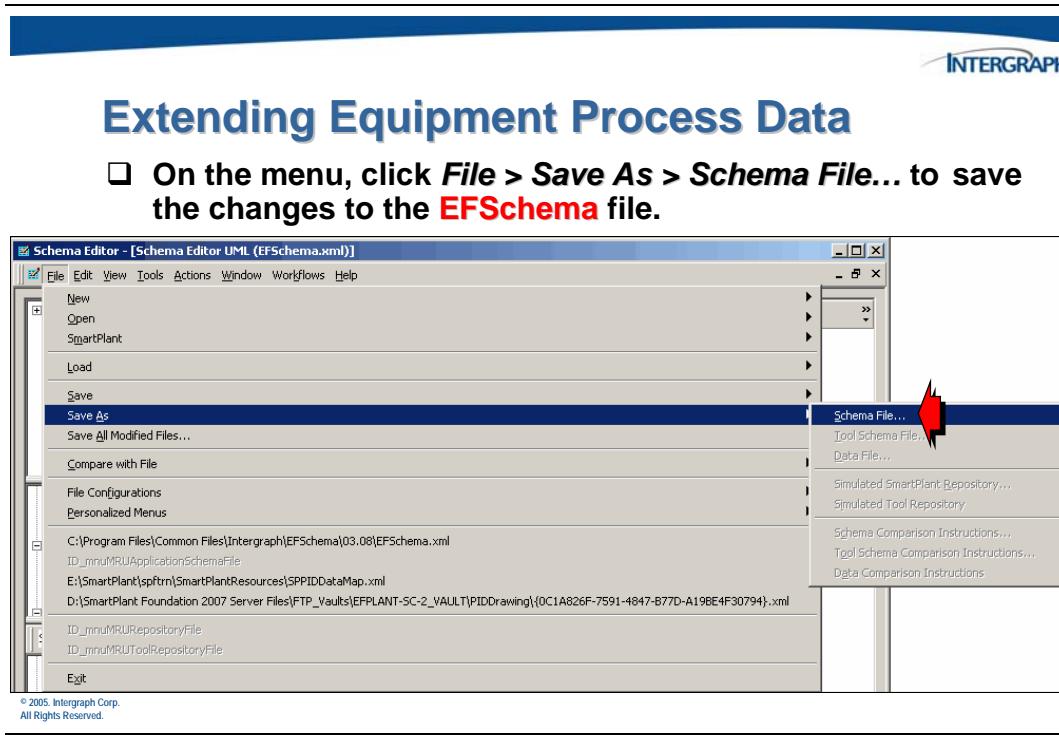


Extending Equipment Process Data

- Click OK to save the modifications and dismiss this dialog.



This will close the dialog and now the added relationships can be saved to the schema file.



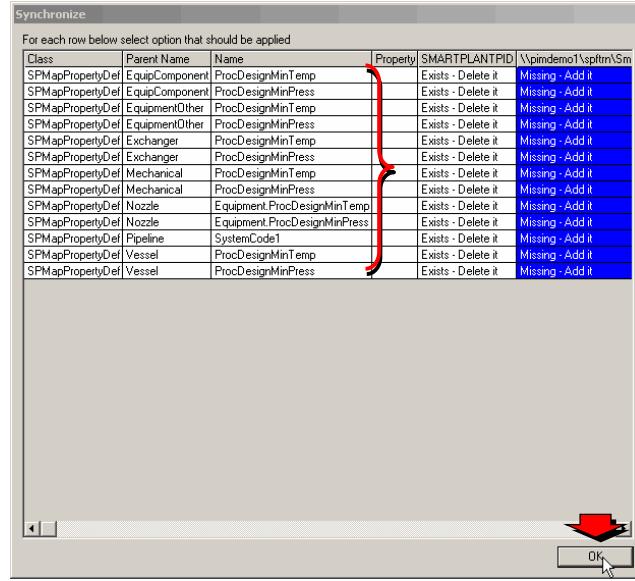
Close the schema editor and re-start it but instead of opening a configuration file, connect to SmartPlant in order to perform some synchronization and mapping.

Create the site specific application properties, *ProcDesignMinTemp* and *ProcDesignMinPress* for an Equipment Class. These new custom properties have been added to SmartPlant P&ID through the Data Dictionary Manager. The properties, *ProcDesignMinTemp* and *ProcDesignMinPress* will automatically be added to the SPPID tool map schema. These are the two properties that will be mapped to the SmartPlant existing properties *NormDesignPressMin* and *NormDesignTempMin*.



Extending Equipment Process Data

- Click OK from the Synchronize dialog to have the schema editor read the tool meta-schema and automatically add the new properties to the tool map schema.



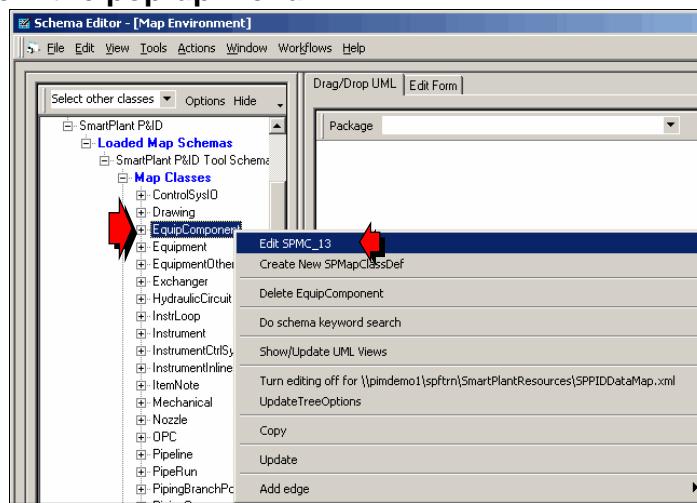
© 2005, Intergraph Corp.
All Rights Reserved.

During the synchronization process, the tool meta data properties will be automatically added to the tool map schema. All that is left to do is to perform the mapping.



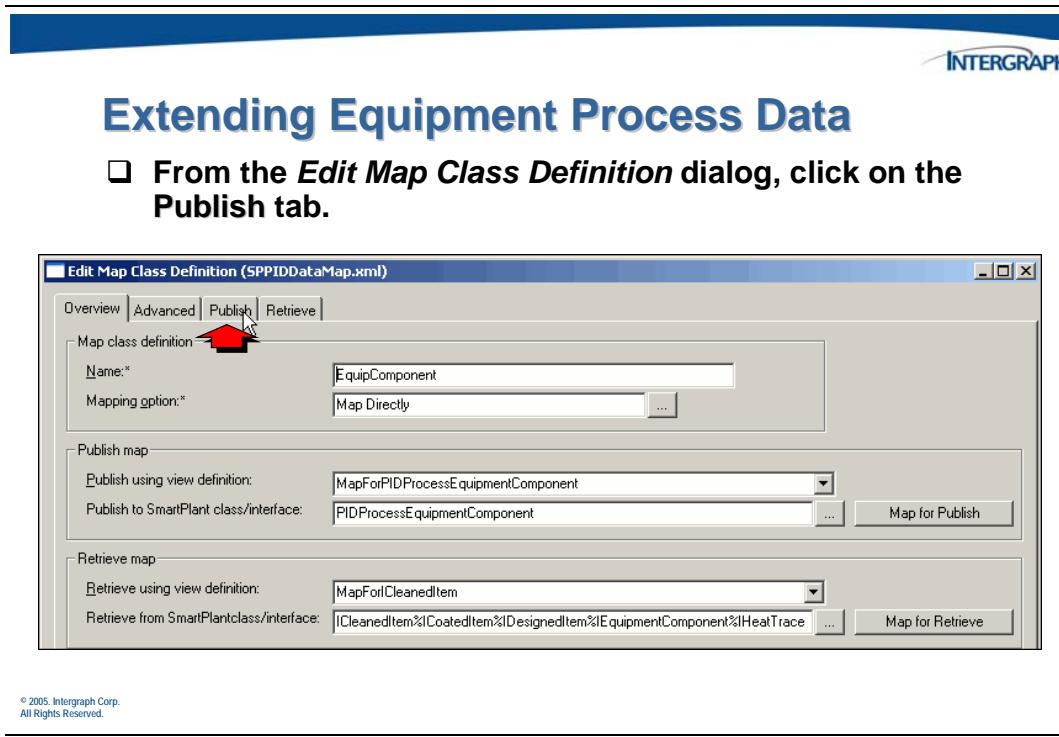
Extending Equipment Process Data

- Right click on **EquipComponent** and select *Edit SPMC_13* from the pop-up menu.

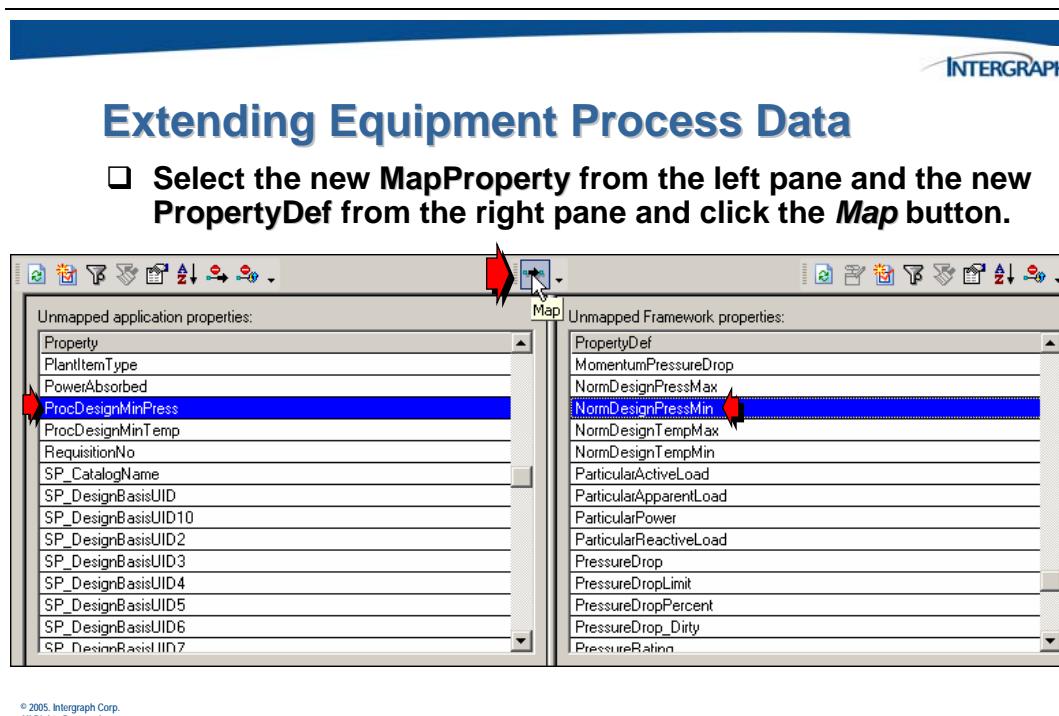


© 2005, Intergraph Corp.
All Rights Reserved.

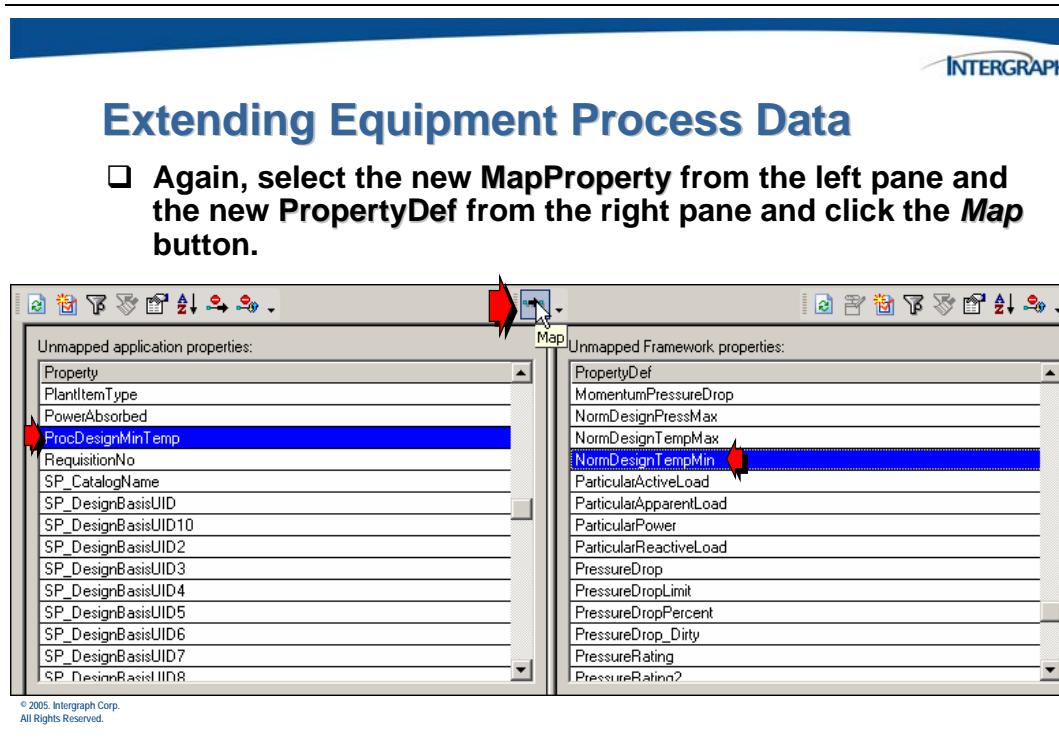
The *Edit Map Class Definition* dialog will be displayed.



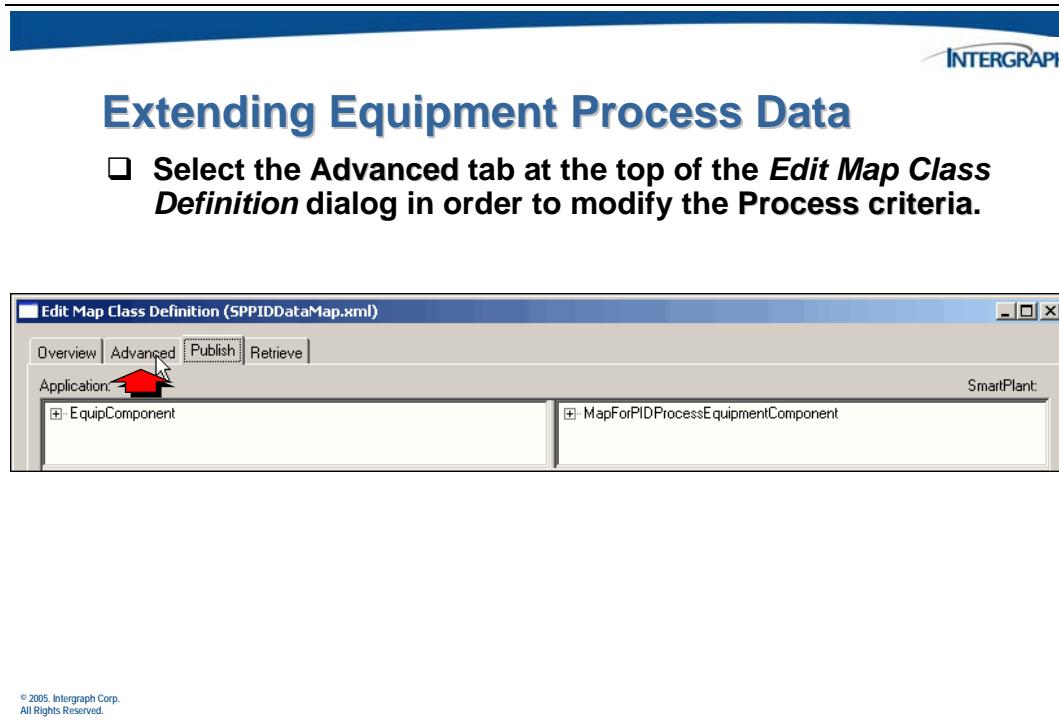
The two properties have been automatically added to the tool map schema and will be mapped to two existing SmartPlant schema properties that were identified earlier.



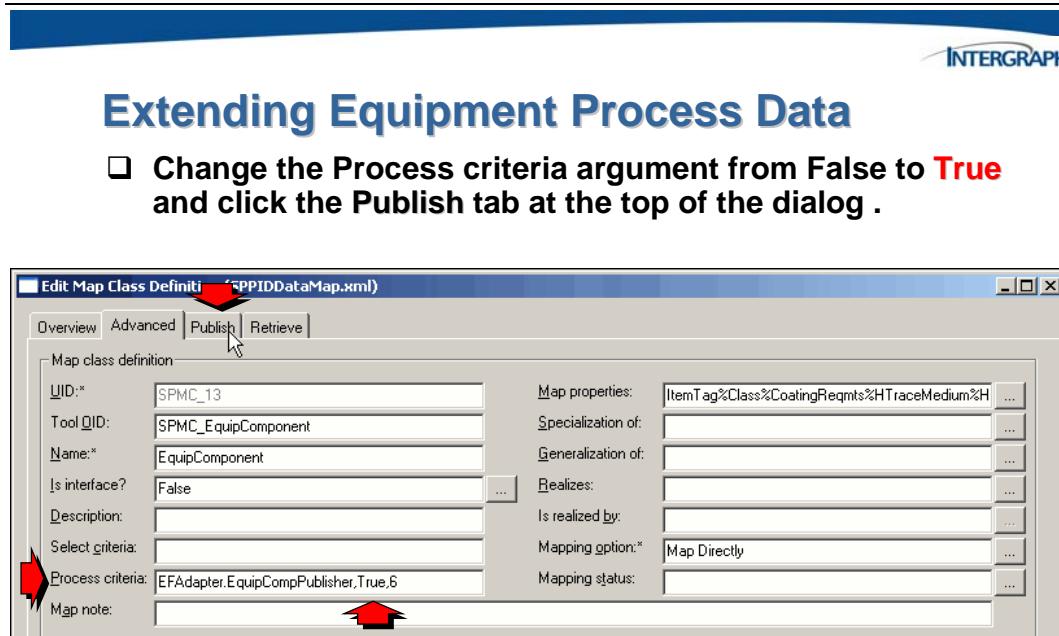
Repeat the mapping for the second new MapProperty, *ProcDesignMinTemp*.



The last step is to set the necessary *Process criteria* parameter in order for the publish to work correctly.

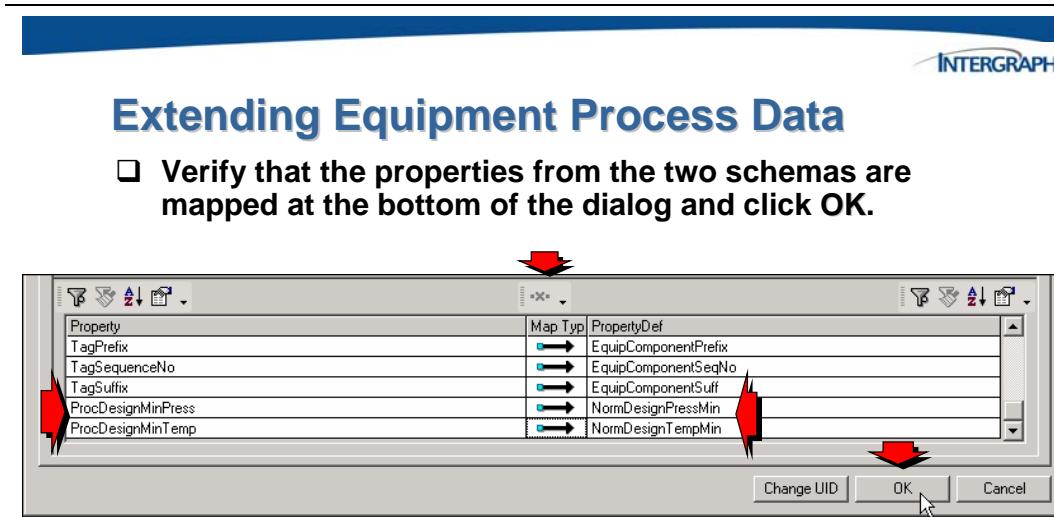


Set the *Process criteria* field to "EFAdapter.EquipCompPublisher,**True**,6". The default setting is "EFAdapter.EquipCompPublisher,**False**,6".



© 2005, Intergraph Corp.
All Rights Reserved.

This will change back to the *Publish* tab in order to verify the mapping before saving these changes and exiting.



© 2005, Intergraph Corp.
All Rights Reserved.

A PID drawing can now be published and the mapped properties will have the application assigned values written in the XML data file.

A P P E N D I X

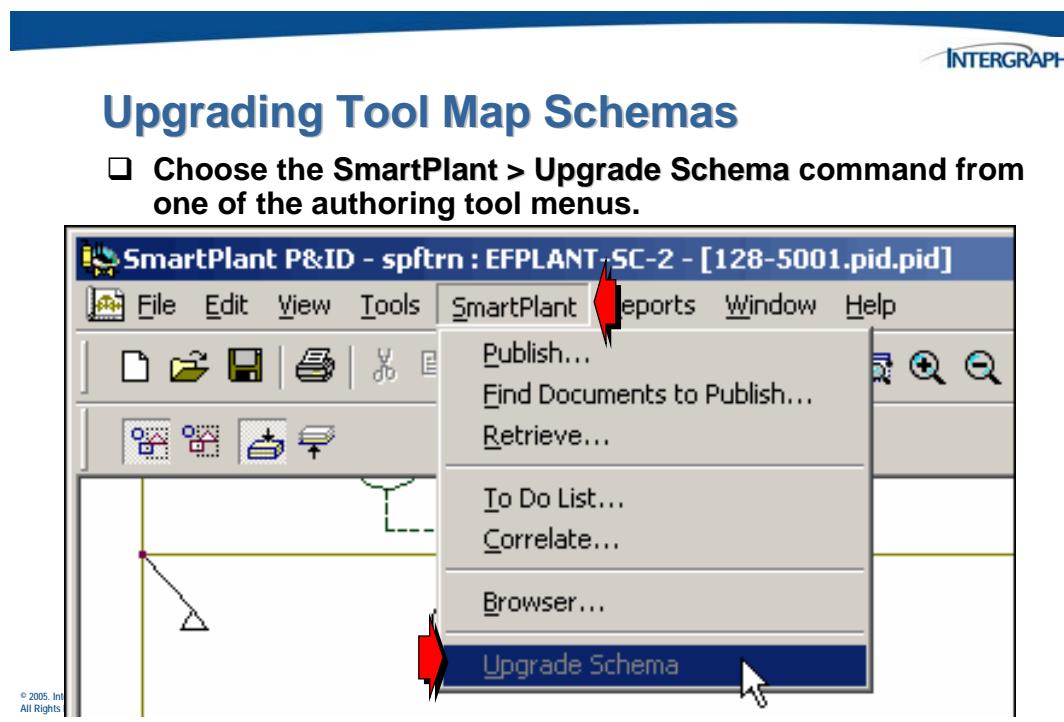
E

Tool Map Schema Upgrade

E. Upgrading a Tool Map Schema

If you have previously customized the tool schema for a previous version, you will need to upgrade the latest tool map schema file in order to preserve your custom changes. The last and current tool map schema will be delivered with each authoring tool.

You can use the **SmartPlant > Upgrade Schema** command from the authoring tool to perform a tool map schema update.



This will compare the last and current tool map schema and create an instructions file. Once the instructions file has been created, the differences are then applied to your site extended tool map schema which may be different from the one delivered with the software.

The software upgrades the tool map schema with the latest version data from the files in the “tool” folder path, and registers the schema version at the beginning of the file.

DSPF1-TP-100008A

SmartPlant Modeling and Mapping

Course Guide