

SYBASE®

XML Model
User's Guide

Sybase® PowerDesigner®

11.1

Windows

DOCUMENT ID: DC20014-01-1110-01

LAST REVISED: June 2005

Copyright © 1991-2005 Sybase, Inc. All rights reserved.

Information in this manual may change without notice and does not represent a commitment on the part of Sybase, Inc. and its subsidiaries.

Sybase, Inc. provides the software described in this manual under a Sybase License Agreement. The software may be used only in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, SYBASE (logo), ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Afaria, Answers Anywhere, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APTLibrary, ASEP, AvantGo.Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, DirectConnect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMap, ECRTTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, M2M Anywhere, Mach Desktop, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Anywhere, M-Business Channel, MBusiness Network, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, mFolio, Mirror Activator, ML Query, MobiCATS, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASiS, OASiS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Pylon, Pylon Anywhere, Pylon Application Server, Pylon Conduit, Pylon PIM Server, Pylon Pro, QAnywhere, Rapport, Relational Beans, RemoteWare, RepConnector, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase IQ, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, and XP Server are trademarks of Sybase, Inc. or its subsidiaries.

All other trademarks are property of their respective owners.

Contents

About This Book	ix
1	XML Model Basics.....1
	Functional overview 2
	What is an XML model? 3
	About XML..... 4
	Objects in an XML model 5
	How to link a child object to a parent object in an XML model? 8
	Defining the XML model environment..... 10
	Choosing the XML language of an XML model..... 10
	Changing the XML language of an XML model..... 11
	Selecting extended model definitions at model creation .. 13
	Defining model options..... 14
	XML model extended dependencies 15
	Defining an XML model 17
	Defining model properties 17
	Creating an XML model..... 24
	Opening an existing XML model 28
	Detaching an XML model from the workspace 28
	Saving and closing an XML model 29
2	Building an XML model.....31
	XML diagram basics 32
	Defining an XML diagram..... 32
	Why build an XML diagram? 33
	Creating an XML diagram 34
	Defining elements 35
	Defining element properties 36
	Creating an element 43
	How to link a child object to an element? 45
	Defining the attributes of an element..... 46
	Modifying element display preferences 52
	Linking child elements to a parent element..... 52
	Defining Any properties 58
	Modifying the Any display preference..... 61

Defining Any Attribute properties.....	61
Defining identity constraints	64
Defining a unique constraint.....	64
Defining a key constraint	66
Defining a keyRef constraint	67
Creating an identity constraint.....	69
Defining an identity constraint selector.....	70
Defining an identity constraint field.....	73
Defining groups.....	76
Defining a group of elements	76
How to link a child object to a group of elements?	81
Modifying the group display preference	82
Defining a group of attributes	82
Managing external shortcuts through references and data types	87
Defining simple types.....	89
What is a simple type?	89
Defining simple type properties	90
Modifying the simple type display preference.....	90
Creating a simple type.....	90
Defining complex types.....	92
What is a complex type?	92
Defining complex type properties	93
Modifying complex type display preferences.....	96
Creating a complex type.....	96
How to link a child object to a complex type?.....	98
Defining simple content properties	100
Defining complex content properties.....	101
Defining derivations	102
Deriving by extension	102
Deriving by restriction	104
Deriving by list	112
Deriving by union.....	113
Defining annotations	115
Defining annotation properties	116
Defining documentation properties	116
Defining application information properties	117
Creating an annotation	118
Defining notations	120
Defining notation properties	120
Creating a notation	121
Defining entities	122
Defining entity properties.....	122
Creating an entity	123
Defining import, include and redefine	125
Defining an import	125

Defining an include	127
Defining a redefine	128
Defining business rules.....	132
What is a business rule?	132
Defining business rule properties	132
Creating a business rule.....	133
Applying a business rule to an XML object	135

3

Working with an XML model	139
Checking an XML model.....	140
XML model check options	140
XML model object selection in the Check Model	141
Checking an XML model	142
Displaying previously applied check options in an XML model	144
Making corrections based on XML model check results.....	145
XML Model objects verified by Check Model.....	147
Group particle check	147
Model check	148
Business rule check in an XSM.....	149
Data source check	149
File check	150
Entity check	151
Include check	152
Simple type check	152
Complex type check	152
Element check.....	153
Group check.....	155
Attribute check.....	156
Notation check.....	156
Attribute group check	157
Import check.....	158
Redefine check.....	158
Key check.....	159
KeyRef check	160
Unique check.....	161
Extended object check	162
Extended link check	162
Replication check	163
Extension check	163
Restriction check	163
Simple type list check.....	164
Simple type union check	165
Annotation check.....	165
Mapping objects in an XML model.....	166

Understanding object mapping	166
Defining data sources in an XML model	166
Mapping XML Model objects to PDM objects	169
Mapping XML Model objects to OOM objects.....	171
Creating a mapping for an XML object.....	172
Modifying the mapping of an attribute	175
Manipulating XML objects graphically.....	176
Local objects	176
Global objects.....	177
Example: converting a local object into a global object..	177
Comparing and merging XML models	179
Generating an XML model from a Physical Data Model.....	180
Generating XML Model objects.....	180
Generating and updating an XML model	181
Defining XML model generation options	182
Generating a new XML model from a PDM	184
Updating an existing XML model	185
Generating an XML model from an Object-Oriented Model .	188
Generating XML Model objects.....	188
Generating and updating an XML model	189
Defining XML model generation options	190
Generating a new XML model from an OOM.....	192
Updating an existing XML model	193
Editing an XML model report	196
What is a report?.....	196
Creating an XML model report.....	196
How can a report underline the hierarchical structure of an XML model?	198

4

Generating and reverse engineering an XSD, a DTD or an XDR file	201
Generating an XSD, a DTD or an XDR file.....	202
Why generate an XSD file?	202
Why generate a DTD file?	202
Why generate an XDR file?.....	202
Defining generation parameters.....	203
How to generate an XSD, a DTD or an XDR file?.....	205
Reverse engineering an XSD, a DTD or an XDR file	208
What is reverse engineering?	208
Why reverse engineer an XSD, a DTD or an XDR file?.....	208
How to reverse engineer an XSD, a DTD or an XDR file?	208

5	Exchanging data with databases supporting XML	215
	Why use XML in databases?	216
	Generating an annotated schema for Microsoft SQL Server 2000	217
	Mapping XML objects to PDM objects	217
	Reinforcing mapping with extended attributes	225
	Generating an annotated schema for Oracle 9i2.....	230
	Generating a DAD file for IBM DB2	237
	Generating SQL/XML queries.....	249
6	Generating from an XML model	255
	Generation basics	256
	Target models parameters	256
	Generating an XML model from an XML model	259
	Why generate an XML model from an XML model?	259
	Generating and updating an XML model.....	259
	XML Model Glossary.....	267
Index	271

About This Book

Subject	<p>This book describes the PowerDesigner XML Model environment. It shows you how to do the following:</p> <ul style="list-style-type: none">◆ Build an XML model◆ Work on an XML model◆ Generate and reverse engineer a DTD file◆ Generate and reverse engineer an XSD file◆ Generate and reverse engineer an XDR file◆ Generate an annotated schema◆ Generate a DAD file◆ Generate an SQL/XML query◆ Generate an XML model from a PDM◆ Generate an XML model from an OOM◆ Generate an XML model from another XML model
Audience	<p>This book is for anyone who will be designing or building an XML model with PowerDesigner. It requires an understanding of XML. Some experience with XML Schema might also be helpful but not required. For more information, see the Bibliography section at the end of this chapter.</p>
Documentation primer	<p>The PowerDesigner modeling environment supports several types of models:</p> <ul style="list-style-type: none">◆ Conceptual Data Model (CDM) to model the overall logical structure of a data application, independent from any software or data storage structure considerations◆ Physical Data Model (PDM) to model the overall physical structure of a database, taking into account DBMS software or data storage structure considerations

- ◆ **Object Oriented Model (OOM)** to model a software system using an object-oriented approach for Java or other object languages
- ◆ **Business Process Model (BPM)** to model the means by which one or more processes are accomplished in operating business practices
- ◆ **XML Model (XSM)** to model the structure of an XML file using a DTD or an XML schema
- ◆ **Requirements Model (RQM)** to list and document the customer needs that must be satisfied during a development process
- ◆ **Information Liquidity Model (ILM)** to model the replication of information from a source database to one or several remote databases using replication engines
- ◆ **Free Model (FEM)** to create any kind of chart diagram, in a context-free environment

This book only explains how to use the XML Model. For information on other models or aspects of PowerDesigner, consult the following books:

General Features Guide To get familiar with the PowerDesigner interface before learning how to use any of the models.

Conceptual Data Model Getting Started To learn the basics of the CDM.

Conceptual Data Model User's Guide To work with the CDM.

Physical Data Model Getting Started To learn the basics of the PDM.

Physical Data Model User's Guide To work with the PDM.

Object Oriented Model Getting Started To learn the basics of the OOM.

Object Oriented Model User's Guide To work with the OOM.

Business Process Model Getting Started To learn the basics of the BPM.

Business Process Model User's Guide To work with the BPM.

Requirements Model User's Guide To work with the RQM.

Information Liquidity Model User's Guide To work with the ILM.

Reports User's Guide To create reports for any or all models.

Repository Getting Started To learn the basics of the Repository.

Repository User's Guide To work in a multi-user environment using a central repository.

Typographic conventions

PowerDesigner documentation uses specific typefaces to help you readily identify specific items:

- ◆ monospace text (normal and **bold**)
Used for: Code samples, commands, compiled functions and files, references to variables.
Example: declare user_defined..., the **BeforeInsertTrigger** template.
- ◆ UPPER CASE
Object codes, reversed objects, file names + extension.
Example: The AUTHOR table appears in the Browser. Open the file OOMAFter.OOM.
- ◆ **bold text**
Any new term.
Example: A **shortcut** has a target object.
- ◆ SMALL CAPS
Any key name.
Example: Press the ENTER key.

Bibliography

W3C XML Recommendation – <http://www.w3.org/TR/REC-xml>

W3C DTD Recommendation – <http://www.w3.org/TR/REC-xml#dt-doctype>

W3C XML Schema Recommendation –
<http://www.w3.org/XML/Schema#dev>

W3C XML-Data Note – <http://www.w3.org/TR/1998/NOTE-XML-data-0105/>

CHAPTER 1

XML Model Basics

About this chapter This chapter presents PowerDesigner XML Model. It provides you with an introduction to the basic notions of XML modeling.

Contents

Topic	Page
Functional overview	2
What is an XML model?	3
Defining the XML model environment	10
Defining an XML model	17

Functional overview

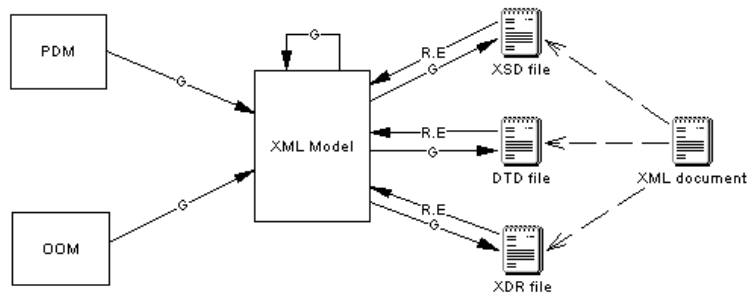
With the graphical interface and the Browser tree view of PowerDesigner XML Model, you can design an XML diagram which represents the content of an XML Schema Definition file (.XSD), a Document Type Definition file (.DTD) or an XML-Data Reduced file (.XDR). Since XML structures can be very complex, it is much easier to visualize them through comprehensive and explicit diagrams, than to read XML-coded pages.

Once you have created an XML diagram, you can generate an XSD, a DTD or an XDR file to share the structure of an XML document via an ordinary plain text file.

Conversely, you will be able to reverse engineer an XSD, a DTD or an XDR file into an XML model, with its corresponding diagram.

The XML Model allows you to:

- ◆ Build an XML model
- ◆ Check an XML model
- ◆ Map objects in an XML model
- ◆ Edit a report of an XML model
- ◆ Generate and reverse engineer an XSD, a DTD or an XDR file
- ◆ Generate an XML model from a Physical Data Model (PDM)
- ◆ Generate an XML model from an Object Oriented Model (OOM)
- ◆ Generate an XML model from an XML model



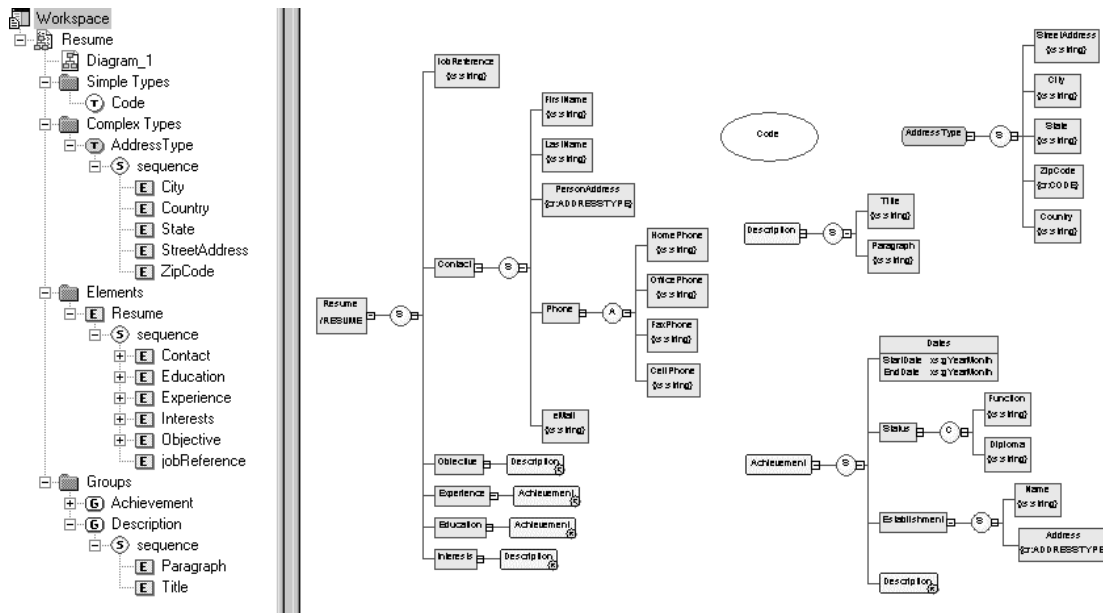
R.E: Reverse Engineering
G: Generation

What is an XML model?

An XML model is a graphical representation of an XML Schema Definition file (.XSD), a Document Type Definition file (.DTD) or an XML-Data Reduced file (.XDR).

With its Browser tree view and diagram, an XML model gives you a global and schematic view of all the elements composing an XSD, a DTD or an XDR file. This is very helpful when you need to understand, check or modify the complex structure of an XSD, a DTD or an XDR file.

Example of an XML model (Browser and diagram):



Demo models

Demo XML models are available in the Examples directory.

About XML

Why use XML?

The eXtensible Markup Language is used for different reasons:

- ◆ It describes and structures data, whereas HTML only displays data
- ◆ It uses a self-describing and personalized syntax
- ◆ It can be exchanged between incompatible systems, since data is stored in plain text format

DTD, XSD or XDR

The structure of an XML model is described by a DTD, an XSD or an XDR file:

- ◆ A DTD file is a basic way to describe the structure of an XML document. It is a raw list of all the legal elements making up an XML document

Extract of a DTD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Project Management -->

<!ELEMENT Database (DIVISION,EMPLOYEE,CUSTOMER,PROJECT,
TEAM,MATERIAL,PARTICIPATE,MEMBER,USED,COMPOSE)>
<!ELEMENT DIVISION EMPTY>
<!ATTLIST DIVISION
          DIVNUM          CDATA
          DIVNAME         CDATA
          DIVADDR         CDATA>
<!ELEMENT EMPLOYEE EMPTY>
<!ATTLIST EMPLOYEE
          EMPNUM          CDATA
          EMP_EMPNUM     CDATA
          DIVNUM         CDATA
          EMPFNAM        CDATA
          EMPLNAM        CDATA
          EMPFUNC        CDATA
          EMPSAL         CDATA>
```

- ◆ An XSD file (or schema) is an elaborated way to describe the structure of an XML document. It can support namespaces, derivations, keys, simple and complex user-defined data types and a robust collection of predefined data types

Extract of an XSD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Project Management
-->
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="Database">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DIVISION">
          <xs:complexType>
            <xs:attribute name="DIVNUM">
              <xs:simpleType>
                <xs:restriction base="ID">
                  <xs:minInclusive value="1"/>
                  <xs:pattern value="00000"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="DIVNAME" type="NAME">
            </xs:attribute>
            <xs:attribute name="DIVADDR" type="SHORT_TEXT">
            </xs:attribute>
          </xs:complexType>
        </xs:element>
```

An XSD file always starts with the <schema> tag (root element). All objects created in the model will appear in the XSD file between the schema start-tag and end-tag

- ◆ An XDR file is a simplified XSD file (or schema). It does not support simple and complex user-defined data types

Extract of an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="PROJECT"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <description>Project Management</description>
  <ElementType name="DIVISION" content="empty">
    <AttributeType name="DIVNUM"/>
    <attribute type="DIVNUM"/>
    <AttributeType name="DIVNAME" dt:type="string"/>
    <attribute type="DIVNAME"/>
    <AttributeType name="DIVADDR" dt:type="string"/>
    <attribute type="DIVADDR"/>
  </ElementType>
```








An XDR file always starts with the <schema> tag (root element). All objects created in the model will appear in the XDR file between the schema start-tag and end-tag

Objects in an XML model

An XML model represents the structure of a potential or existing XML document through an XSD, a DTD or an XDR file.

An XML model is a tree structure of child elements attached to parent elements. Elements are the basic describing items of an XML model. They can be made of other elements combined in different ways through group particles. Elements are specified by attributes and data types which can be predefined or user-defined. Simple and complex data types can be defined as global (directly linked to the <schema> tag) or local (embedded in an element declaration).

The following table displays the specific objects of an XML model:

Tool	Object	Description
	Element	The basic object of an XML model. An element can contain other elements or attributes
	Group	A group of elements arranged by a group particle. A group is defined once and reused elsewhere in the model through references
	Any	Any type of object. Any can only be attached to a sequence or a choice group particle
—	Attribute	Additional information about an element or a complex type. An attribute is defined by a built-in data type or a simple data type
—	Attribute Group	A group of attributes. An attribute group is defined once and reused elsewhere in the model through references
—	Simple Type	A simple data type. A simple type is used in the case of elements or attributes with text-only content. Only available in a model targeted with XSD
	Complex Type	A complex data type. A complex type is used to introduce elements or attributes within an element declaration. Only available in a model targeted with XSD
	Sequence	A group particle to arrange a set of elements. A sequence group particle indicates that elements must appear at least once in the order of their declaration
	Choice	A group particle to arrange a set of elements. A choice group particle indicates that one element must be chosen among all elements
	All	A group particle to arrange a set of elements. An all group particle indicates that each element can appear once or not, in any order

Tool	Object	Description
—	Notation	A notation is used to define and process non-XML objects within an XML model

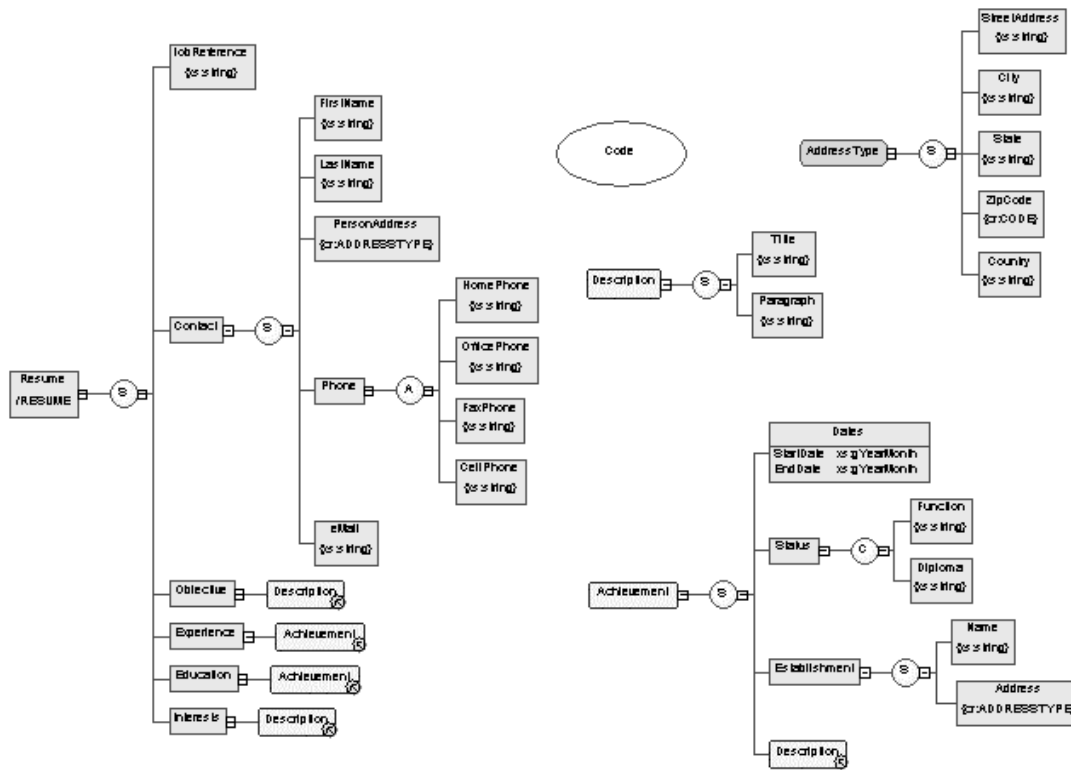
These tools are available in the palette of the diagram window.

The main objects of an XML model are represented by symbols in this diagram.

Objects in a diagram

Some objects may not appear in a diagram because they do not have a symbol or this symbol has been deleted or hidden. Always check the existence of objects in the Browser tree view.


























Example of an XML diagram:






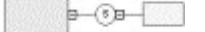


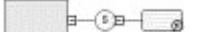








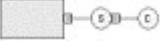
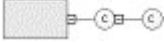

You can use several diagrams to have partial views of a complex diagram.

How to link a child object to a parent object in an XML model?

XML objects do not support standard link objects. To link a child object to a parent object, you must click the child object tool in the palette and then click the symbol of the parent object in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Element symbol	Group symbol	Complex type symbol
			
 Any			
			
	No link	No link	No link
			
			
 All			

Tool	Sequence symbol	Choice symbol	All symbol
			
 Any			No link
			No link
	No link	No link	No link

Tool	Sequence symbol	Choice symbol	All symbol
			No link
			No link
 All	No link	No link	No link

Caution

A group particle (sequence, choice, all) cannot be created from scratch in a diagram.

It must be the child element of an element, a group or a complex type.

For more information, see sections [How to link a child object to an element](#), [How to link a child object to a group particle](#), [How to link a child object to a group of elements](#), [How to link a child object to a complex type](#), in chapter [Building an XML model](#).

Defining the XML model environment

The XML model environment includes a set of parameters and configuration options that define various aspects of the model content and behavior. You can set these parameters:

- ◆ At model creation
- ◆ After creating a model with default options and parameters
- ◆ When creating a model template

Choosing the XML language of an XML model

An XML language contains specifications for a particular language. It provides PowerDesigner with the syntax and guidelines for implementing stereotypes, data types, scripts and constants for an XML language. You manage an XML language from the Resource Editor. The language displays a tree view with several categories that can be used to extend XML model objects (Profile category) or manage generation (Generation category).

Each XML model is by default attached to an XML language. When you create a new XML model, you must choose an XML language. You can create a new XML language or use the XML languages delivered with PowerDesigner.

The definition of an XML language is available from its property sheet. You can select and configure parameters used when defining objects or generating from an XML model.

🌀 For more information on XML languages, see chapter XML Languages Reference Guide in the *Advanced User Documentation*.

🌀 For more information on resource files, see chapter The Resource Editor in the *General Features Guide*.

Not certified resource file

Some resource files are delivered with "Not Certified" in their names. Sybase will perform all possible validation checks, however Sybase does not maintain specific environments to fully certify these resource files. Sybase will support the definition by accepting bug reports and will provide fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. Users are invited to assist Sybase by testing fixes of the definition provided by Sybase and report any continuing inconsistencies.

Changing the XML language of an XML model

If you change the XML language of an XML model, you have to define the status of the language:

XML language	Description
Share	To use the shared XML language stored in the XML Languages directory of your installation. Any changes made to the XML language are available to the linked XML model
Copy	To create a copy of the XML language in the model. The current XML language is independent from the original XML language, so any changes made in the XML language are not available to other models. The XML language is saved with the model and cannot be used by other models

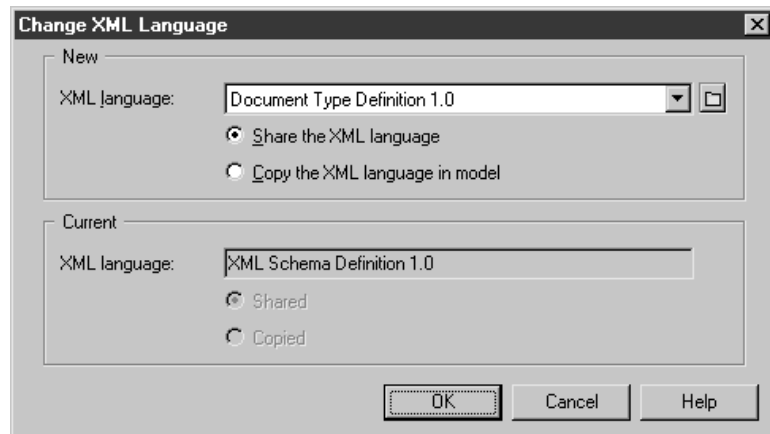
Caution

PowerDesigner is delivered with a set of XML languages. It is strongly advised to make a backup copy of each XML language before you start modifying them.

❖ To change the XML language of an XML model:

- 1 Select Language→Change Current Language.

The Change XML Language dialog box appears.



- 2 Select an XML language.
- 3 Select the Share or Copy radio button.

- 4 Click OK.

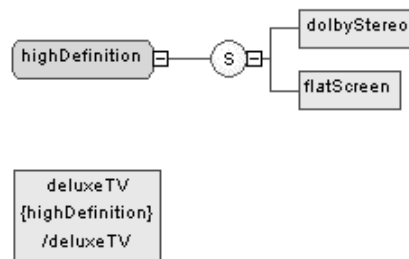
A message box informs you that the current XML language has been changed.

- 5 Click OK.

Changes concerning simple and complex types

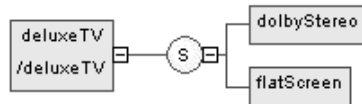
Simple types and complex types are only supported by XSDs (schemas). When changing an XSD into a DTD or an XDR, simple types and global complex types (directly linked to the <schema> tag) disappear from the diagram and the Browser tree view. Local complex types (within an element) are expanded in the diagram, beneath their containing element.

- ◆ Example of a complex type with XSD:



HighDefinition is a global complex type, reused as data type for the deluxeTV element.

- ◆ The same example with DTD or XDR:

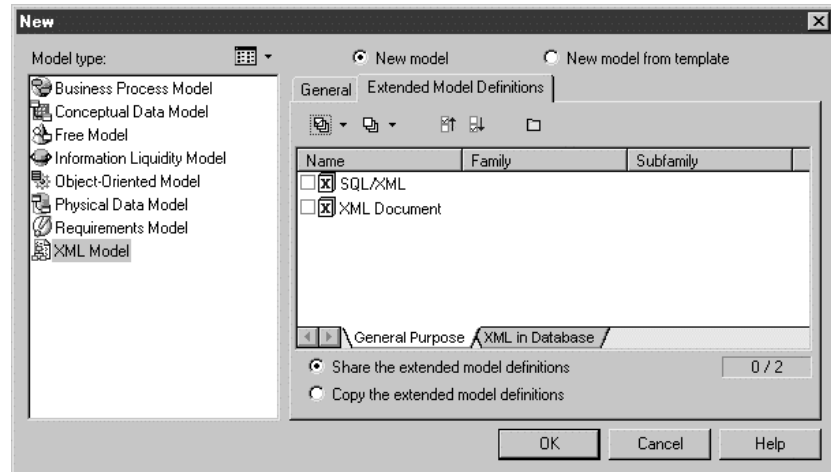


- ☞ For more information on simple and complex types, see sections Defining simple types and Defining complex types in chapter Building an XML model.

Selecting extended model definitions at model creation

Extended model definitions (.XEM files) provide means for customizing and extending PowerDesigner metaclasses, parameters and generation. Extended model definitions are typed like models in PowerDesigner. You create an extended model definition for a specific type of model and you cannot share these files between heterogeneous models.

When you create a new XML model, or when you reverse engineer into a new XML model, you can select one or several extended model definitions and attach them to the model from the New dialog box.



You can choose one of the following options:

Option	Description
Share	Current extended model definition constantly refers to the extended model definition stored in the Resource Files\Extended Model Definitions directory. Any changes made to the extended model definition are shared by all linked XEM
Copy	Current extended model definition is a unique copy of the extended model definition stored in the Resource Files\Extended Model Definitions directory. The current extended model definition is independent of the original one, so modifications made to the extended model definition in the Resource Files\Extended Model Definitions directory are not available to the copied XEM. This one is saved with the XML model and cannot be used without it

☞ For more information on extended model definitions, see chapter Extended Model Definitions Reference Guide, in the *Advanced User Documentation*.

Defining model options

Name/Code case sensitive

You can define the case sensitivity of names and codes for all objects in the current model. When this check box is selected, it implies that you can have two objects with identical name or code but different case in the same namespace.

Unlike other model options, you can modify the name and code case sensitivity during the design process. However, if you do so, make sure you run the check model feature to verify if the model does not contain any duplicate object.

Enable links to requirements

Requirements are descriptions of customer needs that must be satisfied during development processes.

You can enable links to requirements for all objects in the current model. When this check box is selected, it implies that the **Requirements** tab appears in the objects property sheet. The Requirements page allows you to attach requirements to objects; these requirements are defined in the Requirements models open in the workspace. Attached requirements and Requirements models are synchronized.

☞ For more information on requirements, see the *Requirements Model User's Guide*.

Naming conventions

You can also set naming conventions for each type of objects in your model.

☞ For information on naming conventions, see section Defining naming conventions, from chapter Managing Models, in the *General Features Guide*.

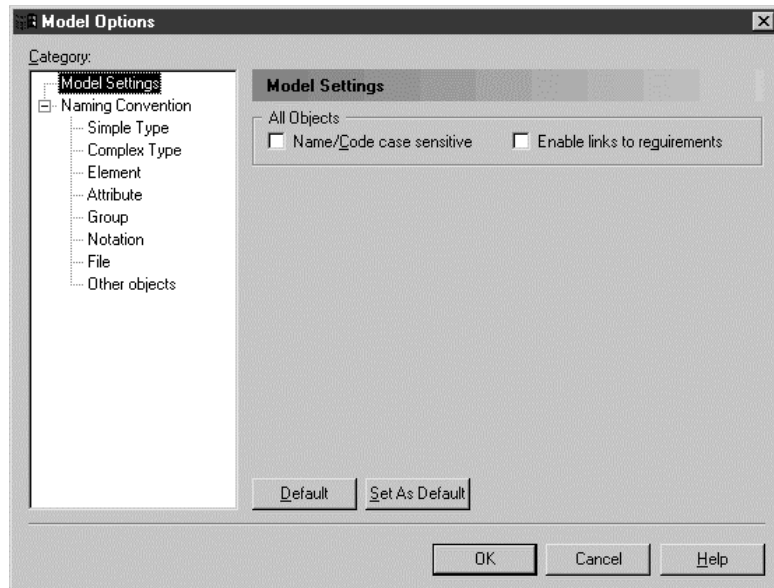
❖ **To define XML model options:**

- 1 Select Tools→Model Options.

or

Right-click the diagram background and select Model Options in the contextual menu.

The Model Options dialog box opens to the Model Settings pane.



- 2 Select or clear the Name/Code case sensitive check box in the All Objects groupbox.
- 3 Click OK.

XML model extended dependencies

Extended dependencies are links between objects of an XML model. These links help to make object relationships clearer but are not interpreted and checked by PowerDesigner as they are meant to be used for documentation purposes only.

You can complement these links by applying stereotypes. Stereotypes can be used to define extended dependencies between objects in an XML model.

You can type stereotypes directly in the Stereotype column of the object property sheet or select a value from the dropdown listbox if you have previously defined stereotypes in an embedded or imported extended model definition (.XEM).

↪ For more information on extended model definitions, see chapter Extended Model Definitions Reference Guide in the *Advanced User Documentation*.

Defining an XML model

This section presents the main operations you have to perform before starting to build or work on an XML model.

Defining model properties

The model property sheet displays the definition of the current model.

Only the specific pages of an XML model are explained in this section.

↪ For information on the generic pages of a model property sheet, see section Using property sheets in chapter Using the PowerDesigner Interface in the *General Features Guide*.

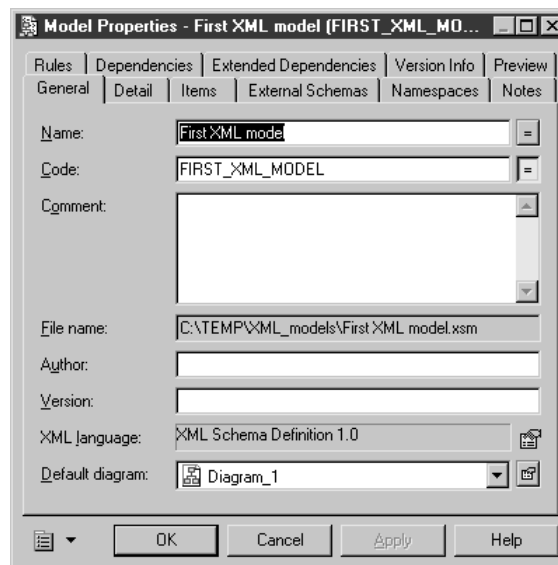
❖ To define the properties of an XML model:

- 1 Select Model→Model Properties.

or

Right-click the diagram background and select Properties from the contextual menu.

The model property sheet appears.



- 2 Type changes to model properties in the different pages.

If you want to display the XML language, click the Properties tool beside the XML language box in the General page to display the property sheet of the XML language.

- 3 Click OK.

Model General page


The General page of the model property sheet displays the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the model
File name	Location of the model file. This box is empty if the model has never been saved
Author	Author of the model. You can insert a name, a space or nothing. If you insert a space, the Author field in the title box remains empty. If you intentionally leave the box empty, the Author field in the title box displays the user name from the Version Info page of the model property sheet
Version	Version of the model. You can use this box to display the repository version or a user-defined version of the model. This parameter is defined in the display preferences of the Title node
XML language	Current XML language for the model
Default diagram	Diagram displayed by default when opening the model

Model Detail page







The Detail page of a model property sheet (only available in a model targeted with XSD) displays the following properties:

Property	Description
Target Namespace	Namespace of all the model objects. Its name is a URI which does not refer to any file but only to an assigned name. A prefix can be assigned to the namespace. All the schema elements with this prefix in their start-tag will be associated with the namespace. For example: http://www.mycompany.com/myproduct/XMLmodel
Language	Indicator of the language used in the model. For example: en, en-GB, en-US, de, fr
ID	ID of the model. Its value must be of type ID and unique within the file containing the model. For example: XMOD1
Element Form	Form of the elements declared in the target namespace. If you select Qualified , elements must be qualified with the namespace prefix. If you select Unqualified , elements are not required to be qualified with the namespace prefix. The value of Element Form is the global default value for all the elements declared in the target namespace. To override this setting, individual elements can use the Form attribute
Attribute Form	Form of the attributes declared in the target namespace. If you select Qualified , attributes must be qualified with the namespace prefix. If you select Unqualified , attributes are not required to be qualified with the namespace prefix. The value of Attribute Form is the global default value for all the attributes declared in the target namespace. To override this setting, individual attributes can use the Form attribute
Block	Default value for the Block property of elements and complex types in the target namespace. The Block property prevents an element or a complex type with a specified type of derivation from being used in place of the inherited element or complex type
Final	Default value for the Final property of elements, simple types and complex types in the target namespace. The Final property prevents the specified type of derivation for an element, a simple type or a complex type








 For more information on elements, attributes, simple and complex types, see chapter Building an XML model.


Model Items page


The Items page of the model property sheet displays the list of global objects (with no parent symbol in the diagram, directly linked to the <schema> tag) created in the model. This list reflects the order in which global objects are being declared in the schema. If you want to change this order of declaration, you must select an item in the list and use the arrowed buttons, at the bottom-left corner of the page, to move the selected item in the list:

Button	Moves item
	Top of the list
	Up one page
	Up one line
	Down one line
	Down one page
	Bottom of the list

You can also use the Items page to create global objects in the model:

Tool	Tooltip	Description
	Add Element	Adds an element to the model
	Add Group	Adds a group of elements to the model
	Add Attribute	Adds an attribute to the model
	Add Attribute Group	Adds a group of attributes to the model
	Add Simple Type	Adds a simple type to the model. Only available in a model targeted with XSD
	Add Complex Type	Adds a complex type to the model. Only available in a model targeted with XSD
	Add Notation	Adds a notation to the model, to describe the format of non-XML data





Tool	Tooltip	Description
	Add Annotation	Adds an annotation to the model, to provide documentation or application information. Only available in a model targeted with XSD


 For more information on these items, see chapter Building an XML model.

Model External Schemas page

A schema is an XML-written text defining the content and structure of an XML document. An XML model is a graphic representation of a schema.

You can use the following tools to reuse in your model global objects from other schemas:



Tool	Tooltip	Description
	Add Include	Adds a specified schema file to be included in the target namespace of the current schema
	Add Import	Adds a specified namespace whose schema components are referenced by the current schema
	Add Redefine	Adds a specified schema file whose simple and complex types, groups and attribute groups can be redefined in the current schema
	Add Annotation	Adds an annotation to the model to provide documentation or application information

 For more information on these items, see chapter Building an XML model.

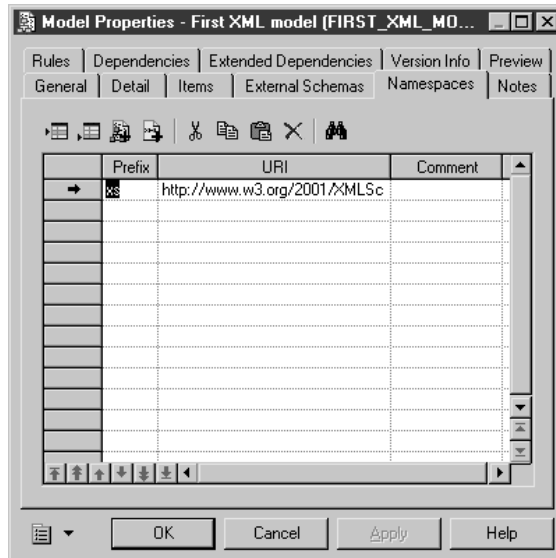
Model Namespaces page

A namespace is a URI indicating a location where objects are declared. The prefix of a namespace, followed by a colon (:) and the name of an object, indicates that this object is declared in that namespace. Namespaces are not supported by DTDs.

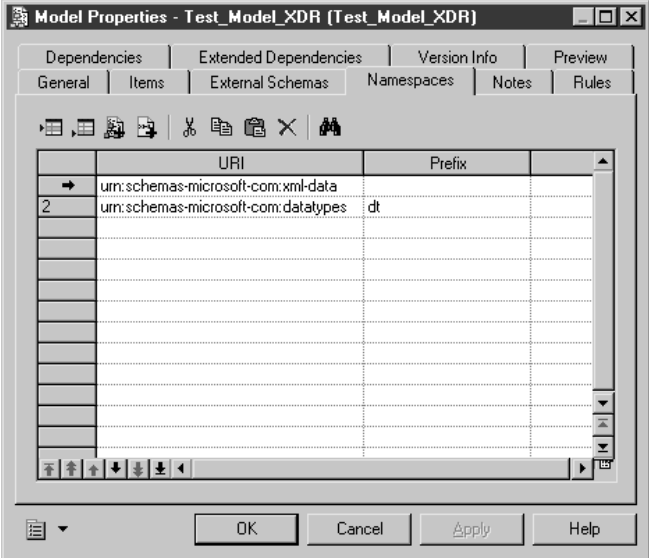
You can use the following tools to attach namespaces from other models or schema files to the current model:

Tool	Tooltip	Description
	Add Namespaces from XML Models	Adds namespaces of other XML models. These are source namespaces for the current model
	Add Namespaces from XML schema files	Adds namespaces of other schema files available on your machine. These are source namespaces for the current model

- ◆ In the case of a model targeted with XSD, the namespace of the W3C XML Schema Recommendation is predefined in the list of namespaces.



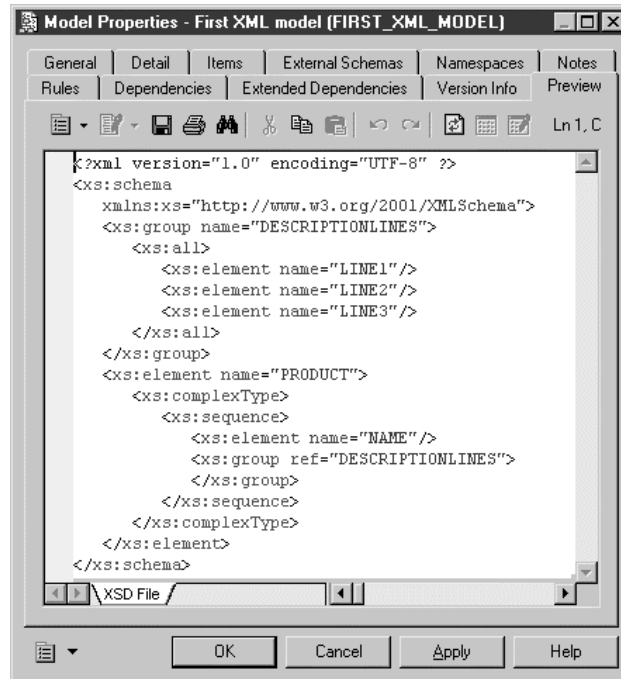
- ◆ In the case of a model targeted with XDR, two namespaces are predefined in the list of namespaces.



Model Preview page

The Preview page of the model property sheet displays a preview of the XSD, DTD or XDR file generated from the XML model.

Example of an XSD file (or schema file):



The schema file starts with the XML declaration followed by the <schema> (root element) declaration.

All objects created in the model will appear in the schema file between the schema start-tag and end-tag.

Creating an XML model

There are several ways to create an XML model:

- ◆ Create a new XML model
- ◆ Create a new XML model using a template
- ◆ Create an XML model using existing elements (reverse engineering an XSD, a DTD or an XDR file, generating from a PDM or an OOM)

Creating an XML model using the New model option

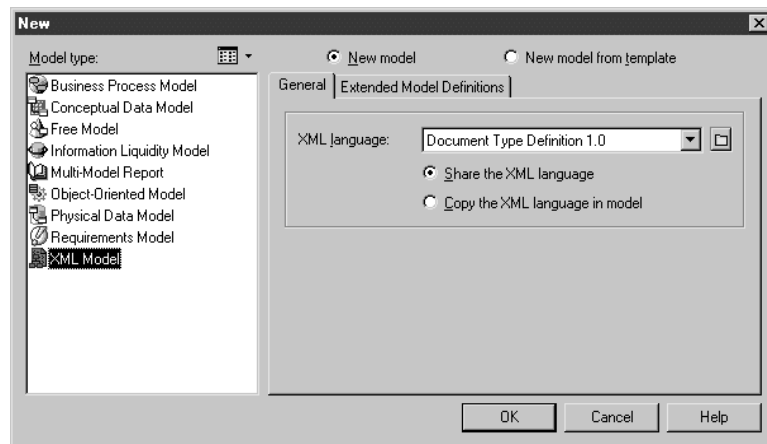
When you create a new XML model using the New model option, you have to select an XML language.

You can then select one of the following options:

Option	Description
Share	To use the shared XML language stored in the XML Languages directory of your installation. Any changes made to the XML language are available to the linked XML model
Copy	To create a copy of the XML language in the model. The current XML language is independent from the original XML language, so any changes made in the XML language are not available to the other models. The XML language is saved with the model and cannot be used by other models

❖ To create a new XML model using the New model option:

- 1 Select File→New to display the New dialog box.



- 2 Select XML Model in the list of model types.
- 3 Select the New model radio button in the upper right part of the dialog box.
- 4 Select an XML language from the XML language dropdown listbox of the General page.
- 5 Select either Share or Copy.

- <optional> If you want to attach one or more extended model definitions to the model, click the Extended Model Definitions tab, and select the extended model definitions of your choice.

🔗 For more information on attaching extended model definitions to a model, see section [Selecting extended model definitions at model creation](#).

- Click OK.

A new XML model is created in the Workspace.

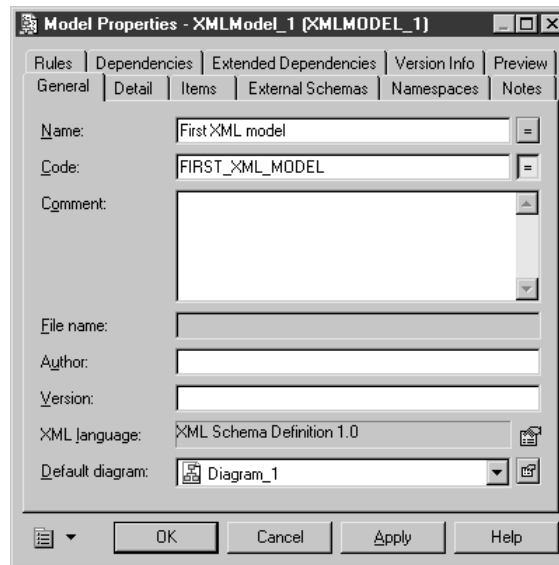
- Select Model→Model Properties.

or

Right-click any empty space in the diagram window and select Properties from the contextual menu.

The model property sheet appears.

- Type a name and a code for the model.



- Click OK.

Creating an XML model using the New model from template option

❖ To create a new XML model using the New model from template option:

- 1 Select File→New to display the New dialog box.
- 2 Select XML Model in the list of model types.
- 3 Select the New model from template radio button in the upper right part of the dialog box to display the Template page.
- 4 Select a model template from the list.

List of templates

You can select user-defined model templates (use the Change User-Defined Model Templates Folder tool to specify the user templates folder) and copy some existing models as model templates using the Copy Model to User-Defined Model Templates Folder tool.

🔗 For more information on model templates, see section Creating a model in chapter Managing Models, in the *General Features Guide*.

- 5 Click OK.

A new XML model is created in the Workspace.

- 6 Select Model→Model Properties.

or

Right-click any empty space in the diagram window and select Properties from the contextual menu.

The model property sheet appears.

- 7 Type a name and a code for the model.
- 8 Click OK.

Opening an existing XML model

An XML model has the file extension .XSM.

Choose XML language If PowerDesigner cannot find the XML language attached to the XML model you want to open, the Choose XML Language dialog box appears to let you select another XML language to attach to the model. You can select the XML language from the XML language dropdown listbox.

❖ **To open an existing XML model:**

- 1 Select File→Open.
or
Click the Open tool.

A standard Windows Open file dialog box appears.

- 2 Select a file with an .XSM extension.
- 3 Click Open.

The model opens in the diagram window and appears in the Browser.

Detaching an XML model from the workspace

When you detach an XML model from a workspace, its node is removed from the Browser and it is no longer defined in the workspace. Yet the file is not deleted from your operating environment.

❖ **To detach an XML model from a workspace:**

- 1 Right-click the XML model node in the Browser and select Detach from Workspace in the contextual menu.

A confirmation box asks if you want to save the XML model.

- 2 Click Yes if you want to save modifications to the XML model.
Select or browse to a directory.
Type a name for the file and click the Save button.

or

Click No if you do not want to save modifications to the file.

The XML model is removed from the workspace.

Saving and closing an XML model

Saving an XML model

To save an XML model, choose one of the following options:

- ◆ Select File→Save
- ◆ Click the Save tool in the standard toolbar
- ◆ Right-click the XML model in the Browser tree view and select Save in the contextual menu

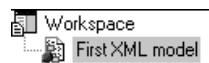
If it is the first time you save an XML model, a standard Windows Save As dialog box appears: Type a file name, choose a folder in your directory and click Save.

Closing an XML model

To close an XML model, choose one of the following options:

- ◆ Select File→Close
- ◆ Right-click the XML model in the Browser tree view and select Close in the contextual menu

When an XML model is closed, a red mark appears on its icon in the Browser tree view:



CHAPTER 2

Building an XML model

About this chapter

This chapter describes how to build an XML model (XSM). It explains the role of each object in an XML model and how to create and modify them.

Contents

Topic	Page
XML diagram basics	32
Defining elements	35
Defining identity constraints	64
Defining groups	76
Managing external shortcuts through references and data types	87
Defining simple types	89
Defining complex types	92
Defining derivations	102
Defining annotations	115
Defining notations	120
Defining entities	122
Defining import, include and redefine	125
Defining business rules	132

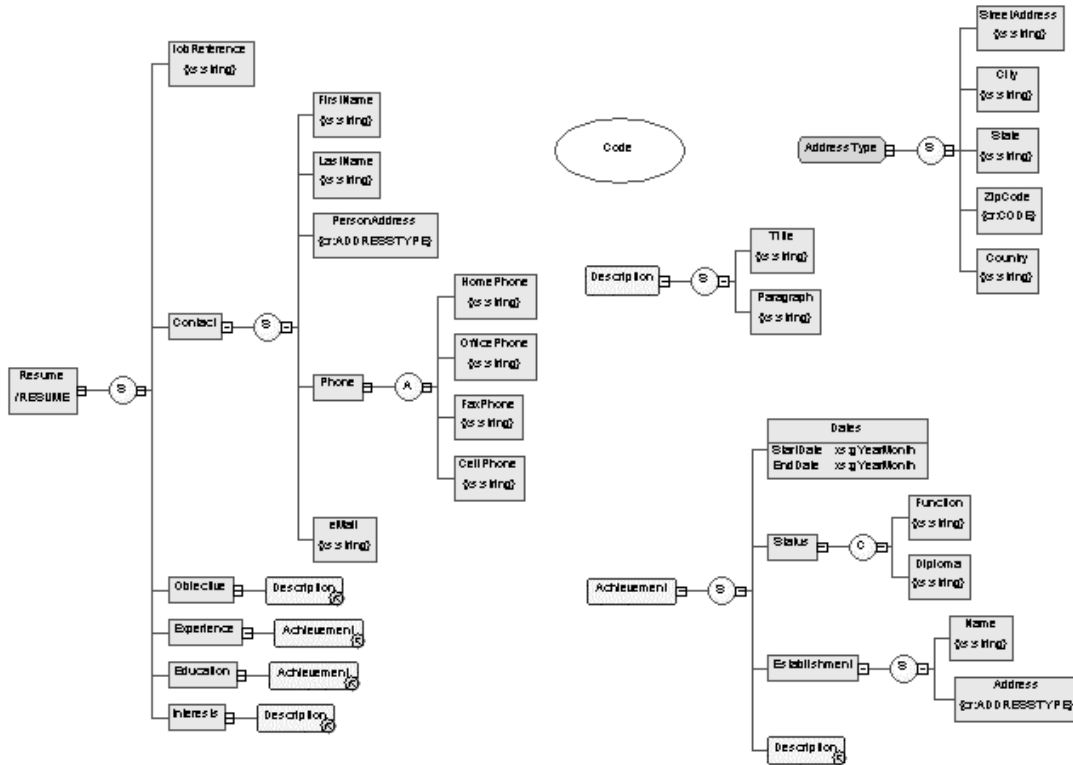
XML diagram basics

You can create XML diagrams in an XML model.

Defining an XML diagram

An XML diagram is a graphical view of an XML model.

The following example shows the diagram of the Resume XML model:



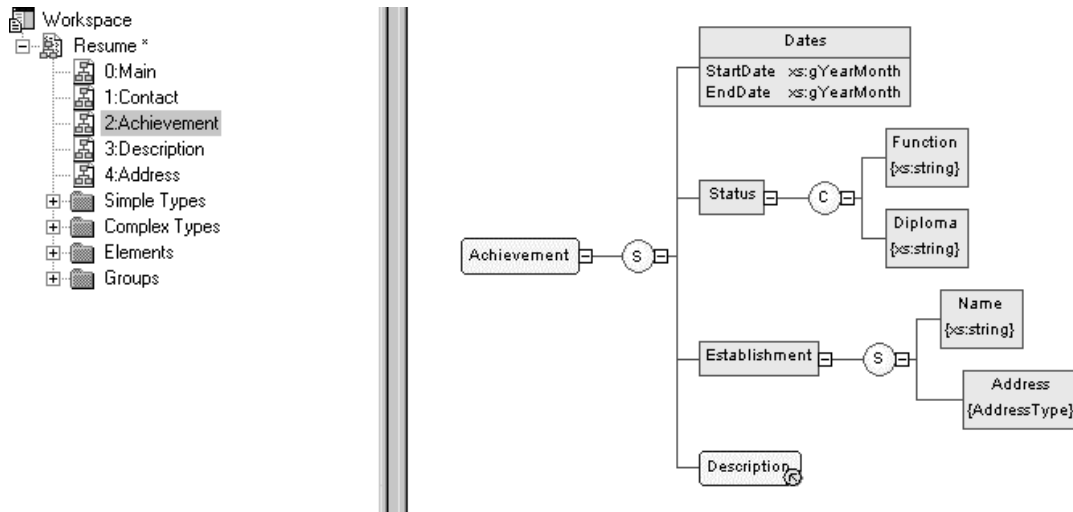
The main objects of an XML model are represented by symbols in its diagram.

Objects in a diagram

Some objects may not appear in a diagram because they do not have a symbol or this symbol has been deleted or hidden.
Always check the existence of objects in the Browser tree view.

If an XML model is too large or too complex, you can create several diagrams to have partial views of the model and focus on certain objects.

The following example displays the Achievement diagram of the Resume XML model:



The original diagram of the Resume XML model being too large (see first picture), it has been split into five diagrams (Main, Contact, Achievement, Description and Address) corresponding to the five main objects of the model.

Why build an XML diagram?

An XML diagram is the easiest way to define the structure and content of an XML document if you are not familiar with the syntax of XML Schema Definition (XSD), Document Type Definition (DTD) or XML-Data Reduced (XDR).

With the user-friendly graphical interface of PowerDesigner XML Model, you can build an XML diagram and then generate automatically an XSD, a DTD or an XDR file.

Creating an XML diagram

There are two ways to create an XML diagram:

- ◆ From a new XML model
- ◆ From an existing XML model

☞ For information on creating an XML diagram from a new XML model, see section Creating an XML model in chapter XML Model Basics.

☞ For information on creating an XML diagram from an existing XML model, see section Creating a new diagram in chapter Managing Models in the *General Features Guide*.

Group Symbols feature

The Symbol→Group Symbols feature is only available for free symbols in an XML diagram.

Expand/Expand All/Collapse/Arrange Symbols features

Right-click a symbol in an XML diagram and select one of these features in the contextual menu:

Expand: the hierarchy below a symbol is partially expanded (only the first level).

Expand All: the hierarchy below a symbol is fully expanded (all levels).

Collapse: the hierarchy below a symbol is hidden.

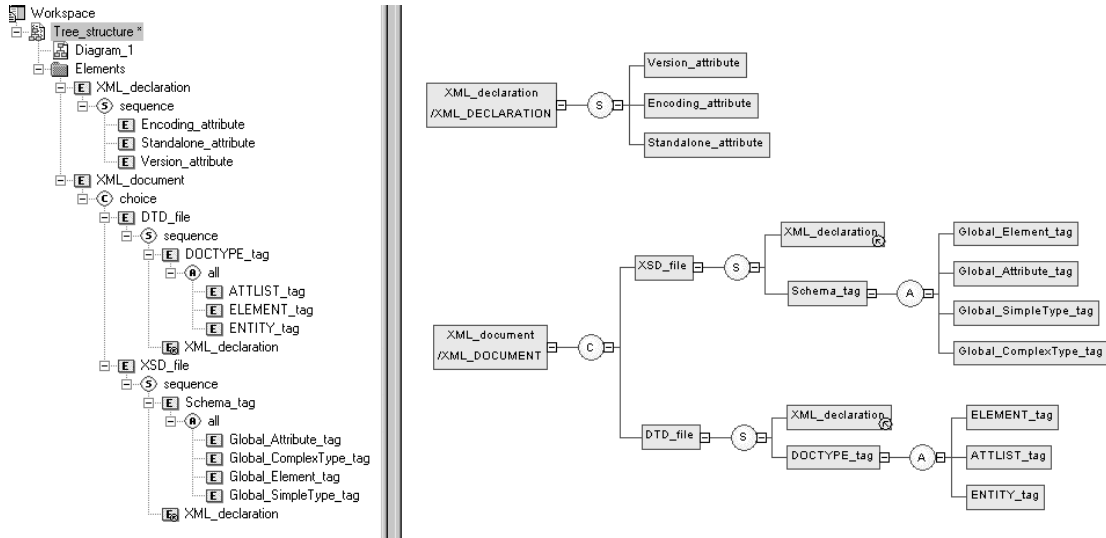
Arrange Symbols: the hierarchy below a symbol is properly displayed.

Defining elements

Elements are the basic building blocks of an XML model.

An XML model is a tree structure of elements where child elements are attached to parent elements.

For example (Browser tree view and diagram):



Generated schema:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="XML_DOCUMENT">
    <xs:complexType>
      <xs:choice>
        <xs:element name="XSD_FILE">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="XML_DECLARATION"/>
              <xs:element name="SCHEMA_TAG">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="GLOBAL_ELEMENT_TAG"/>
                    <xs:element name="GLOBAL_ATTRIBUTE_TAG"/>
                    <xs:element name="GLOBAL_SIMPLETYPE_TAG"/>
                    <xs:element name="GLOBAL_COMPLEXTYPE_TAG"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="DTD_FILE">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="XML_DECLARATION"/>
              <xs:element name="DOCTYPE_TAG">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="ELEMENT_TAG"/>
                    <xs:element name="ATTLIST_TAG"/>
                    <xs:element name="ENTITY_TAG"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="XML_DECLARATION">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="VERSION_ATTRIBUTE"/>
        <xs:element name="ENCODING_ATTRIBUTE"/>
        <xs:element name="STANDALONE_ATTRIBUTE"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

In a schema, elements are declared with <element> tags.

Defining element properties

To display an element property sheet, double-click its symbol in a diagram.

Element general properties

There are global and local elements:

- ◆ Global elements have no parent element in a diagram. They are directly linked to the <schema> tag (root element) in a schema. They can be reused in the model through referencing elements (See “XML_declaration” in the Defining elements example)
- ◆ Local elements have a parent element in a diagram. They are unique within their parent scope. They can be defined by reference to a global element (See Reference in the following table)

Global and local elements in XDR files

In a model targeted with the XML-Data Reduced language, local elements are first declared separately, like global elements (with the <ElementType> tag and a name attribute), then within their parent element (with the <element> tag and a type attribute).

Extract of an XDR file:

```
<ElementType name="localElement"
<ElementType name="globalElement"
  <element type="localElement"/>
</ElementType>
```


Parent elements are linked to their child elements through group particles (sequence, choice or all). A parent element can contain a group of child elements (See Group type in the following table)


You can derive an element data type to extend or restrict its values. (Only with a model targeted with XSD)

The General page of an element property sheet displays the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the element
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

Property	Description
Reference	Name of a global element. The current element will have the same properties as the global element. The Reference property is only available for child elements. Use the dropdown listbox to select a global element in the current model. Use the Browse tool to select a global element from any model opened in the current workspace. If you select a global element from another model, a shortcut is created with the referencing element. When you define a reference, name and code properties are grayed. Name and code are those of the global element
Group type	Indicator that specifies how child elements are to be used within the parent element. It can be a group particle (all , choice , sequence) or a group of elements (group). If you select group, a referencing group is directly linked to the current element (see Defining group properties)
Type	Element data type. Use the dropdown listbox to select a built-in data type. Use the Browse tool to select a simple or a complex type from any model opened in the current workspace. In the case of an XSD, selecting a data type will delete any group particle (and its child elements) or attribute previously defined in the element property sheet. Do not select a data type if you want to define attributes or child elements within the current element
Embedded type	Locally defined data type. It applies to the current element only. Automatically set to Complex if you define a derivation for the element data type. Only available in a model targeted with XSD
Content	Content type of the element. If you select Complex , the element can have child elements. If you select Simple , the element cannot have child elements. Only available in a model targeted with XSD
Derivation	Derivation method for the element data type. Used to extend or restrict the values of the element data type. When you define a derivation, the data type disappears. You must click Apply and then the Properties tool to select a base type in the derivation property sheet. Only available in a model targeted with XSD

 For more information on group particles, see section Linking child elements to a parent element.

 For more information on derivations, see section Defining derivations.

Once you have defined the reference of a referencing element, you can locate the referenced element in the diagram by right-clicking the referencing element symbol and selecting Find Referenced Element in the contextual menu. The referenced element appears with handles in the diagram.

Defining elements
in XDR files

In a model targeted with the XML-Data Reduced language, elements are defined by different attributes:

XDR attribute for an element	Description	Property or page in element property sheet
Model	To specify if a global element can contain new local elements. Set to closed by default. Set to open if an Any symbol is attached to the element symbol	—
Content	To specify the content of a global element. If a group particle and a data type are defined, the content value is mixed . If a group particle and no data type is defined, the content value is eltOnly . If no group particle and a data type is defined, the content value is textOnly . If no group particle or data type is defined, the content value is empty	Group type, Type
Order	To specify how local elements are organized within a global element. Set to seq for a sequence group particle. Set to one for a choice group particle. Set to many for an All group particle	Group type
dt:type	To specify a data type for a global element	Type
dt:values	To specify a list of available values for a global element	Values page
type	To specify the name of a global element as reference for a local element	Reference
minOccurs	To specify the minimum number of occurrences for a local element. Usually set to 0 or 1	Detail page in local element property sheet
maxOccurs	To specify the maximum number of occurrences for a local element. Usually set to 1 or * (unbounded)	Detail page in local element property sheet

Example of an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_elements"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="localElement" content="empty"/>
  <ElementType name="globalElement" model="closed" content="eltonly" order="seq" dt:values="0 1 2">
    <element type="localElement" minOccurs="0" maxOccurs="*" />
  </ElementType>
</Schema>
```

Element detail properties

The Detail page of an element property sheet displays the following properties:

Property	Description
Minimum	Minimum number of times the element can occur. To specify that the element is optional, set this attribute to zero
Maximum	Maximum number of times the element can occur. For an unlimited number of times, select unbounded
Substitution group	Name of a global element for which the current element can be substituted. It must have the same type or a derived type. Its value must be a qualified name (See Glossary)
Default	Default value of the element if its content is a simple type or text-only. Enter a default value only if there is no fixed value
Fixed	Predetermined, unchangeable value of the element if its content is a simple type or text-only. Enter a fixed value only if there is no default value
Block	Property to prevent another element with the same type of derivation from being used in place of the current element
Final	Property to prevent derivation of the current element. Prohibited if the element is not a global element
Form	Form of the element. Used to specify the target namespace of the element. If you select Qualified , a namespace prefix is required to qualify the element. If you select Unqualified , a namespace prefix is not required to qualify the element
ID	ID of the element. Its value must be of type ID and unique within the model containing the element
Abstract	Property defining if the element can appear in the instance document or not. If selected, the element cannot appear in the instance document






Property	Description
Nilable	Property defining if the element is null or not

In the case of a model targeted with XDR, the Detail page is only available for local elements.

Element attributes properties

Attributes give additional information about an element.

The Attributes page of an element property sheet allows you to add attributes to an element declaration:




Tool	Tooltip	Description
	Add Attribute	Creates a local attribute
	Add Attribute Group with Reference to Attribute Group	Adds an attribute group with a reference to an attribute group defined in the current model. Select a name in the Reference dropdown listbox. You can also type a new name in the Reference column and then define a new attribute group in the Attribute Groups list (See Model menu)
	Add Attribute with Reference to Attribute from a Selection	Adds one or several attributes with a reference to global attributes defined in the current model. Select one or several global attributes in the Selection dialog box
	Add Attribute Group with Reference to Attribute Group from a Selection	Adds one or several attribute groups with a reference to attribute groups defined in the current model. Select one or several attribute groups in the Selection dialog box
	Any Attribute	Adds any attribute of a specified namespace

You can access directly to the Attributes page of an element property sheet through the contextual menu. Right-click an element symbol in the diagram and select Attributes in the contextual menu.


Element constraints properties

Identity constraints allow you to indicate that element values must be unique within their specified scope.

You can use the Constraints page of an element property sheet to add the following constraints to an element declaration:

Tool	Tooltip	Description
	Key Constraint	The element value must be a key within the specified scope. The scope of a key is the containing element in an instance document. A key must be unique, not null, and always present
	Unique Constraint	The element value must be unique or null within the specified scope
	KeyRef Constraint	The element value corresponds to those of the specified key or unique constraint

You can access directly to the Constraints page of an element property sheet through the contextual menu. Right-click an element symbol in the diagram and select Constraints in the contextual menu.

 For more information on constraints, see section Defining identity constraints.

Element mapping properties

Object mapping is the ability to establish a correspondence between objects belonging to heterogeneous models and diagrams.

The Mapping page of an element property sheet allows you to map the current element and its attributes to PDM or OOM objects.

Select a data source in the Mapping for dropdown listbox. If it is the first time you define a mapping for an element, the Mapping for dropdown listbox is empty. Click the Add a Mapping for a Data Source tool and select a data source.




Element Sources page


The Element Sources page allows you to associate one or several PDM or OOM objects to the current element.

You can use the Add Objects tool to select objects from the PDMs or OOMs opened in the current workspace.

Attributes Mapping page

The Attributes Mapping page allows you to define the mapping between PDM columns or OOM class attributes and the element attributes.


Tool	Tooltip	Description
	Add Mapping	Use this tool to select the attributes in the current element that will be mapped to PDM columns or OOM class attributes. Once you have selected the attributes, you can use the dropdown listbox in the Mapped to column to select corresponding PDM columns or OOM class attributes
	Create from Sources	Use this tool to copy PDM columns or OOM class attributes in the data source to the current element attributes
	Generate Mapping	Use this tool to automatically generate a mapping between PDM columns or OOM class attributes and element attributes with the same name or code in the data source and the current model

 For more information on element mapping, see section Mapping objects in an XML model in chapter Working with an XML model.

Creating an element

You can create an element:

- ◆ From the palette
- ◆ From the Browser tree view
- ◆ From the List of Elements of the Model menu

 For more information on the different ways to create an element, see section Creating an object in chapter Managing objects of the *General Features Guide*.

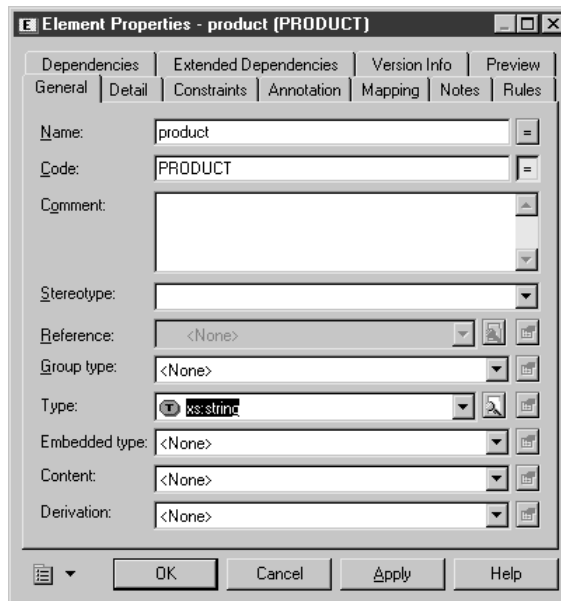
❖ To create an element from the palette:

- 1 Select the Element tool in the palette.
- 2 Click an empty space in the diagram.

An element symbol appears in the diagram at the click position.

Element_1

- 3 Click the Pointer tool in the palette.
or
Right-click to recover the Pointer.
- 4 Double-click the element symbol in the diagram.
The element property sheet appears.
- 5 Type a name and a code for the element.
- 6 Select a data type for the element. You can use the Type dropdown listbox or the Browse tool.








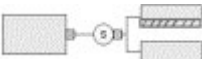

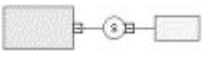

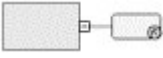









- 7 Click OK.
The element symbol appears in the diagram, with its data type (between brackets) right under its name.

product
{xs:string}

How to link a child object to an element?

XML objects do not support standard link objects. To link a child object to an element, you must click the child object tool in the palette and then click the element symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action	Result
	If you click a parent element symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet	
	If you click the upper part of a child element symbol with the Element tool, a brother element symbol appears above the child element symbol	
	If you click the middle part of a child element symbol with the Element tool, a sequence group particle and a grand child element symbol are created. You can modify the group particle via its property sheet	
	If you click the lower part of a child element symbol with the Element tool, a brother element symbol appears below the child element symbol	
	If you click an element symbol with the Any tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet	
	If you click an element symbol with the Group tool, a referencing group is created. You must now select a group for the reference	

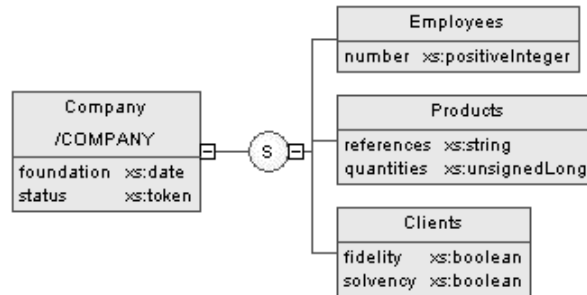
Tool	Action	Result
	If you click an element symbol with the Complex Type tool, a complex type symbol appears superposed, but not linked, to the element symbol. A global complex type cannot be the child of an element	No link
	If you click an element symbol with the Sequence tool, a sequence group particle appears linked to the element symbol	
	If you click an element symbol with the Choice tool, a choice group particle appears linked to the element symbol	
	If you click an element symbol with the All tool, an all group particle appears linked to the element symbol	

Pointer indications
 When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column).
 When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See elements in Tool column).

Defining the attributes of an element

Attributes are used to give additional information about elements.

For example:



Generated schema:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="COMPANY">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EMPLOYEES">
          <xs:complexType>
            <xs:attribute ref="NUMBER"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="PRODUCTS">
          <xs:complexType>
            <xs:attribute name="REFERENCES" type="xs:string"/>
            <xs:attribute name="QUANTITIES" type="xs:unsignedLong"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="CLIENTS">
          <xs:complexType>
            <xs:attribute name="FIDELITY" type="xs:string"/>
            <xs:attribute name="SOLVENCY" type="xs:string"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="FOUNDATION" type="xs:date"/>
      <xs:attribute name="STATUS" type="xs:token"/>
    </xs:complexType>
  </xs:element>
  <xs:attribute name="NUMBER" type="xs:positiveInteger"/>
</xs:schema>

```

In a schema, attributes are declared with <attribute> tags.

Attribute general properties

There are global and local attributes:

- ◆ Global attributes are defined with the Model menu. In a schema, they are directly linked to the <schema> tag (root element). They can be reused for any element in the model through references (See “NUMBER” attribute in the generated schema)
- ◆ Local attributes only apply to the elements in which they are created. They can be defined by reference to a global attribute (See Reference property)

Global and local attributes in XDR files

In a model targeted with the XML-Data Reduced language, local attributes are first declared separately, like global attributes (with the <AttributeType> tag and a name attribute), then within their parent element (with the <attribute> tag and a type attribute).

Extract of an XDR file:

```
<AttributeType name="globalAttribute"/>
<ElementType name="parentElement" content="empty">
  <AttributeType name="localAttribute" default="0"
  <attribute default="0" type="localAttribute"/>
</ElementType>
```

You can derive an attribute data type to extend or restrict its values. (Only with a model targeted with XSD)

To display an attribute property sheet, double-click its name or its icon in the Browser tree view.

The General page of an attribute property sheet displays the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the attribute
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

Property	Description
Reference	Name of an attribute in the current model or another model opened in the workspace. A reference allows you to reuse an attribute with all its properties without having to define it again. Use the dropdown listbox to select an attribute in the current model. Use the Browse tool to select an attribute from any model opened in the workspace. If you select an attribute from another model, a shortcut is created with the referencing attribute. When you define a reference, name and code properties are grayed. The name and code are those of the target attribute
Type	Attribute data type. It must be a qualified name (See Glossary). Use the dropdown listbox to select a built-in data type. Use the Browse tool to select a simple type defined in the current model or another model opened in the workspace
Embedded Type	If selected, the attribute data type disappears and a <simple type> tag is created in the schema within the <attribute> tag. Only available in a model targeted with XSD
Derivation	Derivation method for the attribute data type. Used to extend or restrict the values of the attribute data type. When you define a derivation, the data type disappears. You must click Apply and then the Properties tool to select a type, a base type or member types for the corresponding derivation (list, restriction or union). Only available in a model targeted with XSD

Defining attributes in XDR files

In a model targeted with the XML-Data Reduced language, attributes tags are defined by different attributes:

XDR attribute for an attribute	Description	Property or page in attribute property sheet
name	To specify the name of a global attribute	Name
default	To specify a default value for both global and local attributes	Detail page
dt:type	To specify a data type for a global attribute	Type
dt:values	To specify a list of available values for a global attribute	Values page
type	To specify the name of a global attribute as reference for a local attribute	Reference

Example of an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_attributes"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <AttributeType name="globalAttribute"/>
  <ElementType name="parentElement" content="empty">
    <AttributeType name="localAttribute" default="0" dt:values="1 2 3"/>
    <attribute default="0" type="localAttribute"/>
  </ElementType>
</Schema>
```

Attribute detail properties

The Detail page of an attribute property sheet displays the following properties:

Property	Description
Default	Default value. Enter a default value only if there is no fixed value
Fixed	Fixed value. Enter a fixed value only if there is no default value
Use	Indicator of how the attribute is used. If you select Optional , the attribute is optional and may have any value. If you select Prohibited , the attribute cannot be used. Use this value to prohibit the use of an existing attribute in the restriction of another complex type. If you select Required , the attribute must appear at least once and may have any value matching its data type
Form	Form of the attribute. If you select Qualified , Form must be qualified by combining the target namespace of the schema with the no-colon-name (See Glossary) of the attribute. If you select Unqualified , Form is not required to be qualified with the namespace prefix and is matched against the no-colon-name of the attribute
ID	ID of the attribute. Its value must be of type ID and unique within the model containing the attribute

Attribute values page

The Values page of an attribute property sheet is only available in a model targeted with DTD or XDR. You can set a list of predefined values for an attribute.

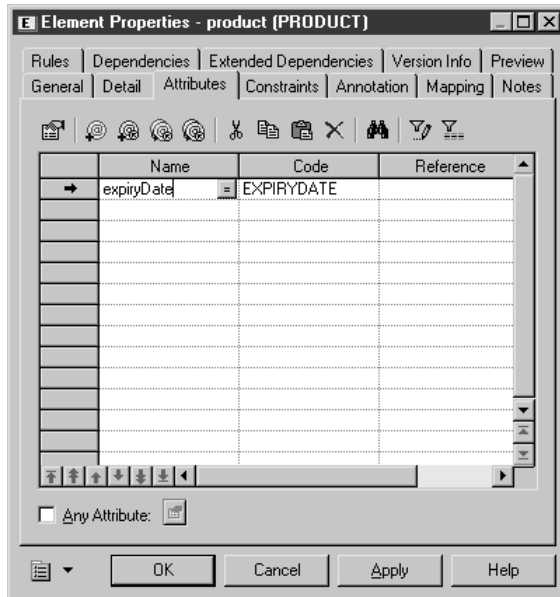
Element values with XDR

In a model targeted with the XML-Data Reduced language, there is also a Values page in the element property sheet.

Adding an attribute to an element

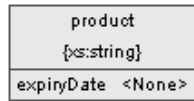
❖ To add an attribute to an element:

- 1 Double-click the element symbol in the diagram.
The element property sheet appears.
- 2 Click the Attributes tab to display the Attributes page.
- 3 Click the Add Attribute tool to add an attribute to the list.
or
Click an empty row in the list.
- 4 Type a name and a code for the attribute.



- 5 Click OK.

The element symbol appears in the diagram with the attribute name. In the example, <None> indicates that no data type has been defined for the attribute.



Modifying element display preferences

You can modify the following display preferences for an element by selecting Tools→Display Preferences:

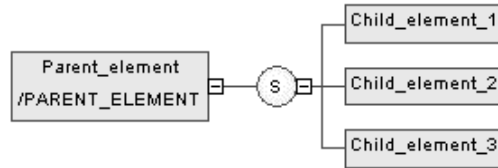
Preference	Description
Attributes	Displays attributes and attribute values of the element
Display limit	Maximum number of attributes displayed
Stereotype	Displays the stereotype of the element
Type	Displays the data type of the element
Show XPath	An XPath expression indicates the relationship between an element and the root element (<schema> tag). Select Never , if you do not want the elements XPath expressions to be displayed. Select Always , if you want all the elements to have their XPath expression displayed. Select Root symbol , if you only want the root symbols (global elements in the main diagram or parent elements in partial diagrams) to have their XPath expression displayed
Element Attributes	Select Type , if you want the attributes data types to be displayed

For more information on XPath expressions, see section Defining an identity constraint selector.

Linking child elements to a parent element

An element composed of other elements is a parent element with child elements.

Child elements are linked to their parent element through a group particle.



Defining group particles

Group particles indicate how child elements are related with their parent element.

You can choose a group particle from the following list:

Tool	Tooltip	Description
	Sequence	Child elements must appear at least once in the order of their declaration
	Choice	Only one child element can be linked to the parent element
	All	Child elements can appear in any order and each of them once or not

In a schema, a group particle is declared with its corresponding tag: <sequence>, <choice> or <all>.

Group particles in XDR files

In a model targeted with the XML-Data Reduced language, group particles are declared through the order attribute of an <ElementType> tag:

Group particle	Value of the order attribute in XDR
Sequence	seq
Choice	one
All	many

Extract of an XDR file:

```
<ElementType name="child1" content="empty"/>
<ElementType name="child2" content="empty"/>
<ElementType name="parent" model="closed" content="eltonly" order="many">
  <element type="child1"/>
  <element type="child2"/>
</ElementType>
```

Group particles properties

To display a group particle property sheet, double-click its symbol in a diagram.






General properties

The General page of a group particle property sheet displays the following properties:

Property	Description
Type	Type of the group particle. You can change its type by selecting a value in the dropdown listbox and clicking OK
Minimum	Minimum number of times the group particle can occur. To specify that the group particle is optional, set this property to zero
Maximum	Maximum number of times the group particle can occur. For an unlimited number of times, select unbounded
ID	ID of the group particle. Its value must be of type ID and unique within the model containing the group particle

Items list

The Items page of a group particle property sheet allows you to add (or order) the following objects to a list of child objects:

Tool	Tooltip	Description
	Add Element	Adds an element to the list
	Add Any	Adds an Any to the list. Only available with a choice or a sequence group particle
	Add Group Particle	Adds a group particle to the list
	Add Element with Reference to Element	Adds a referencing element to the list. Select a global element for the reference in the Selection dialog box. To use this tool, you must have previously defined a global element in the current model
	Add Group with Reference to Group	Adds a referencing group to the list. Select a group for the reference in the Selection dialog box. To use this tool, you must have previously defined a group in the current model

When you add an object to the list, an object is created and its symbol appears in the diagram linked to the group particle.

You can access directly to the Items page of a group particle property sheet through the contextual menu. Right-click a group particle symbol in the diagram and select Items in the contextual menu.

Creating a group particle

There are different ways to create a group particle.

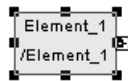
From an element property sheet

The same procedure applies for groups and complex types.

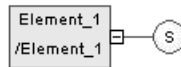
❖ To create a group particle from an element property sheet:

- 1 Select a group particle from the Group type dropdown listbox of the element property sheet.
- 2 Click OK.

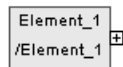
The element symbol appears selected, with an **Expand tab (+)** on its right side.



- 3 Click an empty space in the diagram, to deselect the element symbol, and click the Expand tab (+) to reveal the group particle symbol and its link with the element symbol.



Note the **Collapse tab (-)** on the right side of the element symbol. If you click it, the group particle symbol and its link are replaced by an Expand tab (+).

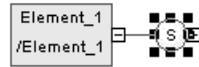


Click the Expand tab (+) to recover the group particle symbol and its link.

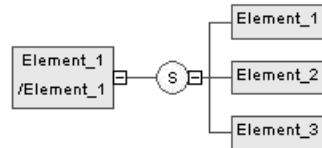
- 4 Double-click the group particle symbol to display the group particle property sheet.
- 5 Select the Items page to display a list of items.
- 6 Click the **Add Element** tool for each child element you create in the list.
or
Click an empty row in the list for each child element you want to create.

- 7 Click OK.

The group particle symbol appears selected, with an Expand tab on its right side.



- 8 Click an empty space in the diagram, to deselect the group particle symbol, and click the Expand tab to reveal the child element symbols and their link.



Name and code uniqueness

To respect the name and code uniqueness within a namespace, child elements are defined within an internal namespace which is their parent element. Therefore, there cannot be a conflict between a parent and a child name.

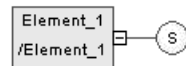
For more information on the namespace concept, see section Managing the namespace in models, in chapter Managing Models of the *General Features Guide*.

From the palette

❖ **To create a group particle from the palette:**

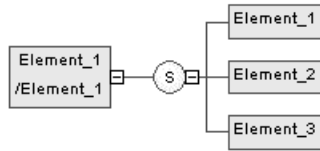
- 1 Select a group particle tool in the palette.
- 2 Click the element symbol in the diagram.

The group particle symbol appears in the diagram, linked to the element symbol.



- 3 Click the Element tool in the palette.
- 4 Click the group particle symbol in the diagram for each child element you need.

The child elements appear one by one in the diagram, linked to the group particle symbol.



- 5 Right-click to recover the Pointer.

Clicking an element symbol with the Element tool

When you click an element symbol with the Element tool, a sequence symbol (by default) appears in the diagram between the parent element and the child element.

To add other child elements, click the sequence symbol with the Element tool.

To change the group particle, double-click the sequence symbol to display its property sheet, then select another group particle in the Type dropdown listbox and click OK.


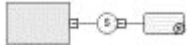
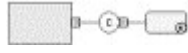


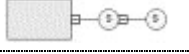



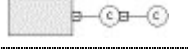

How to link a child object to a group particle?

XML objects do not support standard link objects. To link a child object to a group particle, you must click the child object tool in the palette and then click the group particle symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Caution

A group particle cannot be created from scratch in a diagram. It must be the child element of an element, a group or a complex type.

Tool	Sequence symbol	Choice symbol	All symbol
			No link

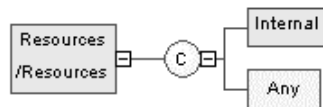
Tool	Sequence symbol	Choice symbol	All symbol
	 A referencing group is created. You must now select a group for the reference	 A referencing group is created. You must now select a group for the reference	No link
	No link	No link	No link
			No link
			No link
 All	No link	No link	No link

Pointer indications
 When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column).
 When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Defining Any properties

Any allows you to attach any type of object to a choice or a sequence group particle.

For example:



◆ Generated XSD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="RESOURCES">
    <xs:complexType>
      <xs:choice>
        <xs:element name="INTERNAL"/>
        <xs:any namespace="##local" processContents="lax"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In an XSD file, Any is declared with the <any> tag.

◆ Generated DTD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT Resources ANY>
<!ELEMENT Internal EMPTY>
```

In a DTD file, Any is declared within an <!ELEMENT> tag with the keyword “ANY”.

◆ Generated XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_Any"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="internal" content="empty"/>
  <ElementType name="resources" model="open" content="elonly" order="one">
    <element type="internal"/>
  </ElementType>
</Schema>
```

In an XDR file, Any is declared through of an <ElementType> tag (resources in the example) with its **model** attribute set to “**open**”. Although it appears in a diagram, Any is not considered as an object in an XDR file.

To display an Any property sheet, double-click its symbol in a diagram.

Any general properties

The General page of an Any property sheet displays the following properties:

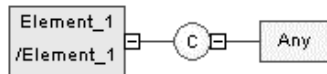
Property	Description
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Minimum	Minimum number of times the Any can occur. To specify that the Any is optional, set this attribute to zero
Maximum	Maximum number of times the Any can occur. For an unlimited number of times, select unbounded

Property	Description
ID	ID of the Any. Its value must be of type ID and unique within the model containing the Any. Only available in a model targeted with XSD
Namespace	Namespaces containing the objects that can be used. If you select ##any , objects from any namespace can be used. If you select ##other , objects from any namespace other than the target namespace of the schema can be used. If you select ##local , objects that are not qualified with a namespace can be used. If you select ##targetNamespace , objects from the target namespace of the schema can be used. If you type a combination of URI references, ##targetNamespace and ##local , provided they are separated by a white space, objects from this combination can be used. Only available in a model targeted with XSD
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the objects specified by the Any. If you select Strict , the XML processor must obtain the schema and validate any object of the specified namespaces. If you select Lax , the XML processor will try to obtain the schema and validate any object of the specified namespaces. If the schema cannot be found, no error will occur. If you select Skip , the XML processor will not try to validate the objects of the specified namespaces. Only available in a model targeted with XSD

❖ **To create an Any:**

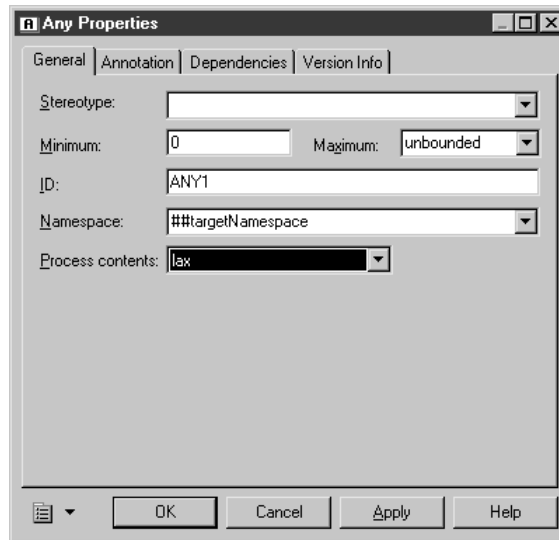
- 1 Select the Any tool in the palette.
- 2 Click a choice or a sequence symbol in the diagram.

The Any symbol appears in the diagram attached to the choice or sequence symbol.



- 3 Right-click to recover the Pointer.
or
Click the Pointer tool in the palette.
- 4 Double-click the Any symbol in the diagram.
The Any property sheet appears.

- Define the Any properties in the different pages of the property sheet.



- Click OK.

Modifying the Any display preference

You can modify the Stereotype display preference for Any by selecting Tools→Display Preferences.

Defining Any Attribute properties

The Any Attribute feature allows you to insert any attribute of specified namespaces into an element, a complex type or an attribute group declaration. It is only available in a model targeted with XSD.

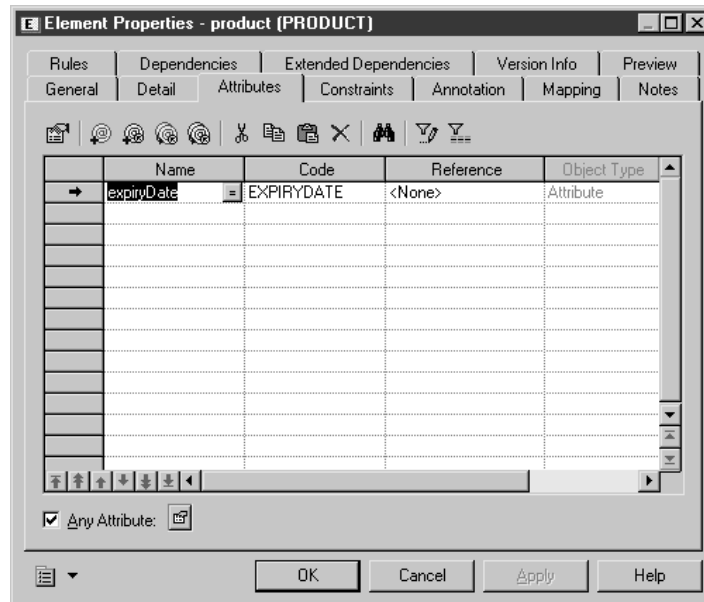
In a schema, Any Attribute is declared with the `<anyAttribute>` tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PRODUCT">
    <xs:complexType>
      <xs:attribute name="EXPIRYDATE" type="xs:date">
      </xs:attribute>
      <xs:anyAttribute namespace="##local" processContents="skip"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Caution
 Any Attribute only appears in a schema (see the Preview page of a model property sheet).

The Any Attribute feature is available via a check box in the bottom-left corner of an Attributes page.



To display an Any Attribute property sheet, select the Any Attribute check box and then click the Properties tool.

Any Attribute general properties

The General page of an Any Attribute property sheet displays the following properties:

Property	Description
ID	ID of the Any Attribute. Its value must be of type ID and unique within the model containing the Any Attribute
Namespace	Namespaces containing the attributes that can be used. If you select ##any , attributes from any namespace can be used. If you select ##other , attributes from any namespace other than the target namespace of the schema can be used. If you select ##local , attributes that are not qualified with a namespace can be used. If you select ##targetNamespace , attributes from the target namespace of the schema can be used. If you type a white space delimited list with URI references, ##targetNamespace and ##local , attributes from this list can be used
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the attributes specified by the Any Attribute. If you select Lax , the XML processor will try to obtain the schema and validate any attribute of the specified namespaces. If the schema cannot be found, no error will occur. If you select Skip , the XML processor will not try to validate the attributes of the specified namespaces. If you select Strict , the XML processor must obtain the schema and validate any attribute of the specified namespaces

Defining identity constraints

Identity constraints enable you to indicate that element values must be unique within their specified scope.

There are three kinds of identity constraints: unique, key and keyRef.

Each identity constraint has two specific attributes: selector and field.

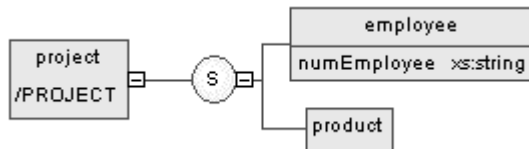
In a schema, an identity constraint is declared with its corresponding tag: <unique>, <key> or <keyRef>.

Identity constraints are only available in a model targeted with XSD.

Defining a unique constraint

A unique constraint specifies that an element or an attribute value (or set of values) must be unique or null within a specified scope.

For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PROJECT">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EMPLOYEE">
          <xs:complexType>
            <xs:attribute name="NUMEMPLOYEE" type="xs:string">
            </xs:attribute>
          </xs:complexType>
        </xs:element>
        <xs:element name="PRODUCT"/>
      </xs:sequence>
    </xs:complexType>
    <xs:unique name="UNIQUENUM">
      <xs:selector xpath="employee"/>
      <xs:field xpath="@numEmployee"/>
    </xs:unique>
  </xs:element>
</xs:schema>
```


The UNIQUENUM unique constraint, defined on the project element, specifies that the numEmployee attribute must be unique or null within the employee element

To display a unique property sheet, double-click its name or its icon in the Browser tree view.

Unique general properties

The General page of a unique property sheet displays the following properties:


Property	Description
Name	Name of the unique constraint. It must be a no-colon-name (See Glossary)
Code	Code of the unique constraint. It must be a no-colon-name
Comment	Descriptive label of the unique constraint
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the unique constraint. Its value must be of type ID and unique within the model containing the unique constraint
Selector (XPath)	An XML Path Language expression that selects a set of elements across which the values specified in the Fields page must be unique. There must be one and only one selector

 For more information on Selector or XPath, see section Defining an identity constraint selector.

Unique fields properties

The Fields page of a unique property sheet displays a list of XPath expressions.

If there is more than one field (or XPath expression), the combination of fields must be unique.

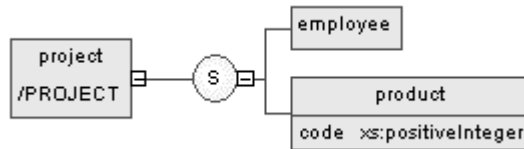
 For more information on XPath expressions, see section Defining an identity constraint selector.

Defining a key constraint

A key constraint specifies that an element or an attribute value (or set of values) must be a key within a specified scope.

A key means that data should be unique, not null and always present within a specified scope.

For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PROJECT">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EMPLOYEE"/>
        <xs:element name="PRODUCT">
          <xs:complexType>
            <xs:attribute name="CODE" type="xs:positiveInteger"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:key name="KEYCODE">
      <xs:selector xpath="product"/>
      <xs:field xpath="@code"/>
    </xs:key>
  </xs:element>
</xs:schema>
```


The KEYCODE key constraint, defined on the project element, specifies that the code attribute must be unique, not null and always present within the product element.

To display a key property sheet, double-click its name or its icon in the Browser tree view.

Key general properties

The General page of a key property sheet displays the following properties:


Property	Description
Name	Name of the key constraint. It must be a no-colon-name (See Glossary)
Code	Code of the key constraint. It must be a no-colon-name
Comment	Descriptive label of the key constraint
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the key constraint. Its value must be of type ID and unique within the model containing the key constraint
Selector (XPath)	An XML Path Language expression that selects a set of elements across which the values specified in the Fields page must be unique. There must be one and only one selector

 For more information on Selector or XPath, see section Defining an identity constraint selector.

Key fields properties

The Fields page of a key property sheet displays a list of XPath expressions.

If there is more than one field (or XPath expression), the combination of fields must be unique.

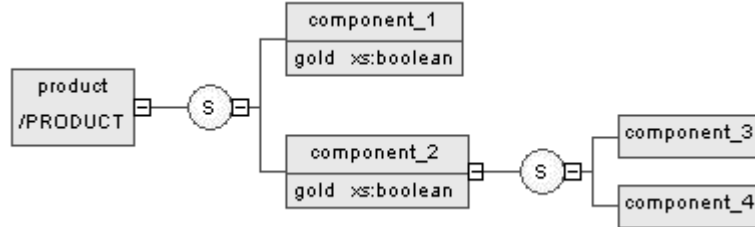
 For more information on XPath expressions, see section Defining an identity constraint selector.

Defining a keyRef constraint

A keyRef constraint specifies that an element or attribute value (or set of values) corresponds to the value of a specified key or unique constraint.

A keyRef is a reference to a key or a unique constraint.

For example:



Generated schema:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:k="keyRef.namespace">
  <xs:element name="PRODUCT">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="COMPONENT_1">
          <xs:complexType>
            <xs:attribute name="GOLD" type="xs:boolean">
            </xs:attribute>
          </xs:complexType>
        </xs:element>
        <xs:element name="COMPONENT_2">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="COMPONENT_3"/>
              <xs:element name="COMPONENT_4"/>
            </xs:sequence>
            <xs:attribute name="GOLD" type="xs:boolean">
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:unique name="UNIGOLD">
      <xs:selector xpath="k:COMPONENT_1"/>
      <xs:field xpath="@GOLD"/>
    </xs:unique>
    <xs:keyref name="KEYREF_UNIGOLD" refer="k:UNIGOLD">
      <xs:selector xpath="k:COMPONENT_2"/>
      <xs:field xpath="@GOLD"/>
    </xs:keyref>
  </xs:element>
</xs:schema>
  
```


The KEYREF_UNIGOLD keyRef, defined on the product element, by reference to the UNIGOLD unique constraint, specifies that the gold attribute must be unique or null within the component_2 element, as well as it must be unique or null within the component_1 element (See UNIGOLD).

To display a keyRef property sheet, double-click its name or its icon in the Browser tree view.

KeyRef general properties

The General page of a keyRef property sheet displays the following properties:


Property	Description
Name	Name of the keyRef constraint. It must be a no-colon-name (See Glossary)
Code	Code of the keyRef constraint. It must be a no-colon-name
Comment	Descriptive label of the keyRef constraint
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the keyRef constraint. Its value must be of type ID and unique within the model containing the keyRef constraint
Reference	Name of a key or a unique constraint defined in the current model (or another model with a specified namespace). The Reference value must be a qualified name (See Glossary)
Selector (XPath)	An XML Path Language expression that selects a set of elements across which the values specified in the Fields page must be unique. There must be one and only one selector

 For more information on Selector or XPath, see section Defining an identity constraint selector.

KeyRef fields properties

The Fields page of a keyRef property sheet displays a list of XPath expressions.

If there is more than one field (or XPath expression), the combination of fields must be unique.

 For information on XPath expressions, see section Defining an identity constraint selector.

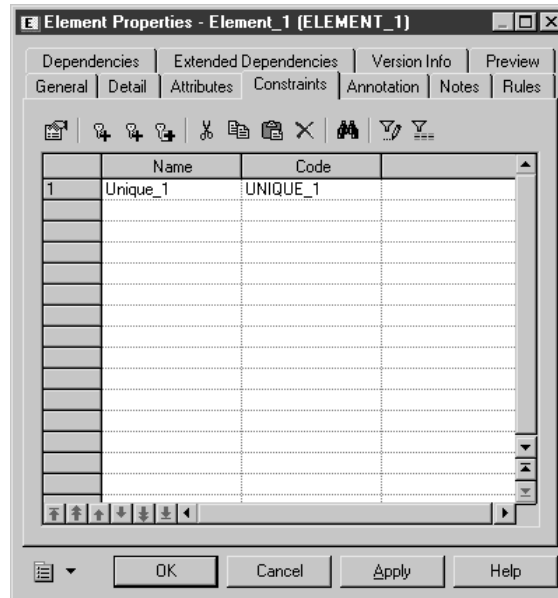
Creating an identity constraint

You create an identity constraint from an element property sheet.

❖ **To create an identity constraint from an element property sheet:**

- 1 Click the Constraints tab of an element property sheet.
- 2 Click the Add Unique, Add Key or Add KeyRef Constraint tool in the Constraints page.

An identity constraint appears in the list with predefined name and code (a unique constraint in the example).



- 3 Modify the name and the code of the identity constraint.
- 4 Click OK.

Defining an identity constraint selector

An identity constraint selector specifies an XPath expression that selects a set of elements for an identity constraint (unique, key or keyRef).

An XPath expression is limited to a subset of the full XPath language defined in the W3C Recommendation XML Path Language 1.0.

An XPath expression allows you to locate a node (an element with its ramifications) in the hierarchical tree structure of an XML document.

XPath abbreviated syntax

You can use the following abbreviated syntax to define an XPath expression:

Syntax	Description
/	Root node of the XML document. It is the root element with its ramifications
.	Selects the context node. It is the current element (on which an identity constraint is defined) with its ramifications
..	Selects the context node parent
*	Selects all the child elements of the context node
employee	Selects all the employee child elements of the context node
s:employee	Selects all the employee child elements of the context node, defined in the namespace with the “s” prefix
@numEmployee	Selects the numEmployee attribute of the context node
@*	Selects all the attributes of the context node
../@numEmployee	Selects the numEmployee attribute of the context node parent
employee[1]	Selects the first employee child element of the context node
employee[last()]	Selects the last employee child element of the context node
*/employee	Selects all the employee grandchildren of the context node
//employee	Selects all the employee descendants of the root node
./employee	Selects the employee descendants of the context node
company//employee	Selects the employee descendants of the company child elements of the context node
//company/employee	Selects all the employee elements with company as parent element in the context node

Syntax	Description
/book/chapter[2]/section[3]	Selects the third section in the second chapter of the book
employee[@dept="doc"]	Selects all the employee child elements of the context node with a dept attribute set to doc
employee[@dept="doc"][3]	Selects the third employee child element of the context node with a dept attribute set to doc
employee[3][@dept="doc"]	Selects the third employee child element of the context node only if it has a dept attribute set to doc
chapter[title]	Selects the chapter child elements of the context node with at least one title child element
chapter[title="About this book"]	Selects the chapter child elements of the context node with at least one title child element with a text content set to About this book
employee[@numEmployee and @dept]	Selects all the employee child elements of the context node with the numEmployee and dept attributes
text()	Selects all the child nodes of the text context node

Defining selector general properties

The General page of a selector property sheet displays the following properties:

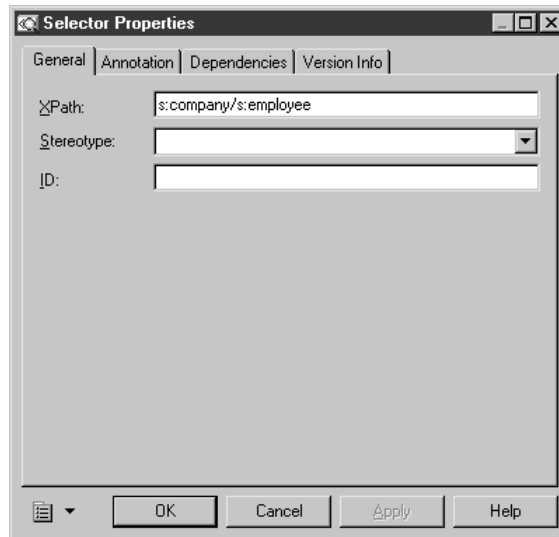
Property	Description
XPath	An XPath expression relative to the parent element being declared. It identifies the child elements to which the identity constraint applies
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the selector. Its value must be of type ID and unique within the model containing the selector

❖ **To display a selector property sheet:**

- 1 Type an XPath expression in the Selector groupbox of an identity constraint (unique, key or keyRef) property sheet.

- 2 Click Apply.
- 3 Click the Properties tool beside the XPath box.

The selector property sheet appears.



- 4 Define the selector properties in the different pages of the property sheet.
- 5 Click OK.

Defining an identity constraint field

The Fields page of an identity constraint lets you type one or more XPath expressions to specify the values used to define an identity constraint (unique, key or keyRef).

🔗 For information on XPath expressions, see section Defining an identity constraint selector.

Defining field general properties

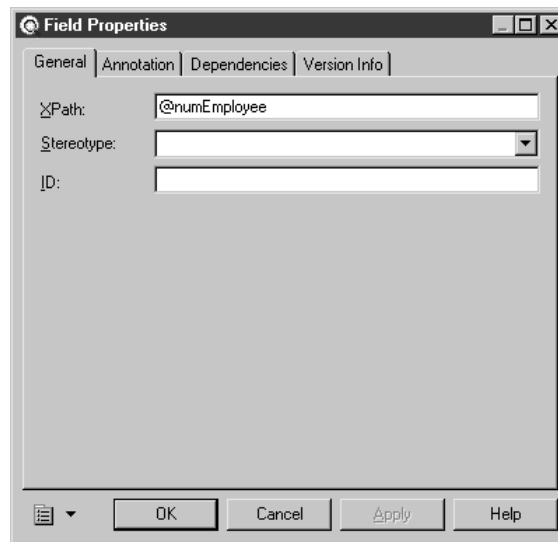
The General page of a field property sheet displays the following properties:

Property	Description
XPath	An XPath expression relative to each element selected by the selector of the identity constraint. It identifies a single element (with a simple type) whose content or value is used for the identity constraint
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the field. Its value must be of type ID and unique within the model containing the field

❖ To create an identity constraint field:

- 1 Click the Fields tab of an identity constraint (unique, key or keyRef) property sheet.
- 2 Click any empty line in the Fields page.
An arrow appears at the beginning of the line.
- 3 Type an XPath expression.
- 4 Click Apply.
- 5 Double-click the arrow left of the row.

The field property sheet appears.



- 6 Define the field properties in the different pages of the property sheet.
- 7 Click OK.

Defining groups

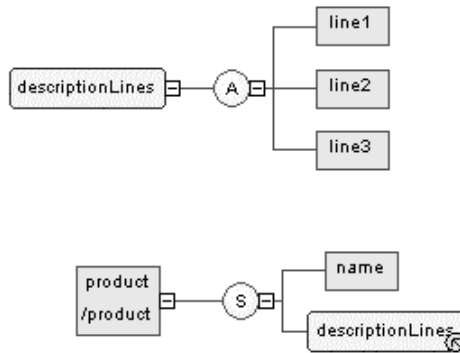
You can define groups of elements or attributes in an XML model.

Defining a group of elements

A group of elements is a set of elements arranged by a group particle (all, choice or sequence).

You create a group of elements when you need to reuse a set of elements in several parts of your model. Once you have defined a group, you can reuse it elsewhere in the model through referencing groups (see Reference property in Group property sheet).

For example:



The descriptionLines group is reused in the definition of the product element by clicking the sequence group particle (S) with the palette Group tool. The Reference property of the referencing group property sheet is then set to descriptionLines.

- ◆ In the generated XSD file, the group is first declared with the <group> tag and then reused through a reference (ref) set to descriptionLines:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:group name="descriptionLines">
    <xs:all>
      <xs:element name="line1"/>
      <xs:element name="line2"/>
      <xs:element name="line3"/>
    </xs:all>
  </xs:group>
  <xs:element name="product">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"/>
        <xs:group ref="descriptionLines">
          </xs:group>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

- ◆ In the generated DTD file, the group is expanded directly within its parent element:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT line1 EMPTY>
<!ELEMENT line2 EMPTY>
<!ELEMENT line3 EMPTY>
<!ELEMENT product (name, line1, line2, line3)>
<!ELEMENT name EMPTY>
```

- ◆ In the generated XDR file, the group is declared through a <group> tag, within an <ElementType> tag with its order attribute set to seq:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_group"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="line1" content="empty"/>
  <ElementType name="line2" content="empty"/>
  <ElementType name="line3" content="empty"/>
  <ElementType name="name" content="empty"/>
  <ElementType name="product" model="closed" content="eltonly" order="seq">
    <element type="name"/>
    <group>
      <element type="line1"/>
      <element type="line2"/>
      <element type="line3"/>
    </group>
  </ElementType>
</Schema>
```

Defining group properties

There are global and referencing groups:

- ◆ Global groups are directly created in a diagram without a parent symbol. In a schema, they are directly linked to the <schema> tag (root element). They can be reused within any element or complex type of the model through references (See descriptionLines in Defining a group of elements)

- ◆ Referencing groups are created in a diagram within an element, a complex type or a global group. You must select a global group for their Reference property (See the following table)

Groups in DTD and XDR files
 In a model targeted with DTD or XDR language, there are no global or referencing groups, although they appear on the diagram.
 Groups are expanded within their parent element and their child elements are declared individually as global elements. (See generated DTD and XDR files in Defining a group of elements)

To display a group property sheet, double-click its symbol in a diagram.

The General page of a group property sheet displays the following properties:

Property	Description
Name	Name of the group. It must be a no-colon-name (See Glossary). Required when the group is global
Code	Code of the group. It must be a no-colon-name. Required when the group is global
Comment	Descriptive label for the group
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	Name of a group in the current model or another model opened in the workspace. It must be a qualified name (See Glossary). A reference allows you to reuse a group with all its properties without having to define it again. Use the dropdown listbox to select a group in the current model. Use the Browse tool to select a group from any model opened in the workspace. If you select a group from another model, a shortcut is created with the referencing group. When a reference is defined, the name and code properties are grayed. The name and code are those of the target group
Group type	Indicator that specifies how child elements are to be used within the group. Select a group particle (all , choice or sequence)
Minimum	Minimum number of times the group can occur. To specify that the group is optional, set this attribute to zero
Maximum	Maximum number of times the group can occur. For an unlimited number of times, select unbounded
ID	ID of the group. Its value must be of type ID and unique within the model containing this group

Once you have defined the reference of a referencing group, you can locate the referenced group in the diagram by right-clicking the referencing group symbol and selecting Find Referenced Group in the contextual menu. The referenced group appears with handles in the diagram.

You can access directly to the Preview page of a group property sheet. Right-click a group (or a referencing group) symbol in the diagram and select Preview in the contextual menu.

Creating a group of elements

You can create a group of elements:

- ◆ From the palette
- ◆ From the Browser tree view
- ◆ From the List of Groups in the Model menu

↪ For more information on the different ways to create a group, see section Creating an object in chapter Managing objects of the *General Features Guide*.

❖ To create a group of elements from the palette:

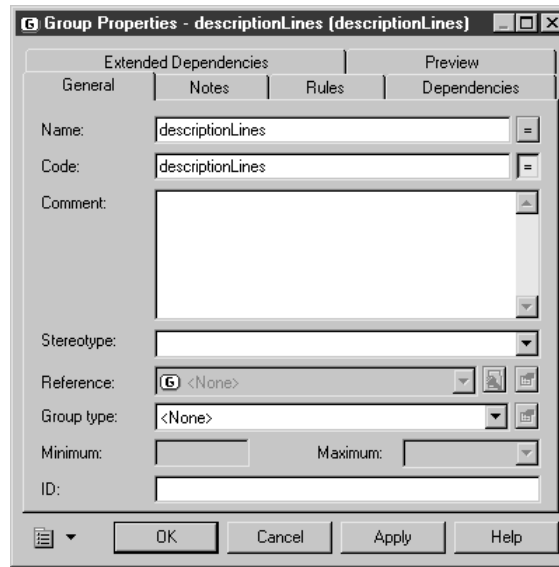
- 1 Click on the Group tool in the palette.
- 2 Click an empty space in the diagram.

A group symbol appears in the diagram at the click position.

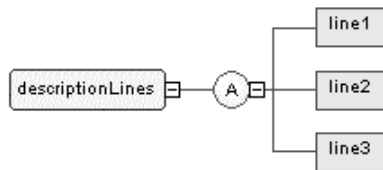


- 3 Click the Pointer tool in the palette.
or
Right-click to recover the Pointer.
- 4 Double-click the group symbol in the diagram.
The group property sheet appears.

- 5 Type a name and a code for the group.
















- 6 Click OK.
- 7 Select a group particle tool in the palette (Sequence, Choice or All).
- 8 Click the group symbol in the diagram.
The group particle symbol appears attached to the group symbol.
- 9 Select the Element tool in the palette.
- 10 Click the group particle symbol in the diagram for each child element you want to create.
The child element symbols appear attached to the group particle symbol.
- 11 Double-click a child element symbol to display its property sheet.
- 12 Type a name and a code for the child element.
- 13 Repeat steps 11 and 12 for each child element.
- 14 Click OK.



How to link a child object to a group of elements?

XML objects do not support standard link objects. To link a child object to a group, you must click the child object tool in the palette and then click the group symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action	Result
	If you click a group symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet	
	If you click a group symbol with the Any tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet	
	If you click a group symbol with the Group tool, a sequence group particle and a referencing group are created. You can modify the group particle via its property sheet. You must now select a group for the reference	
	If you click a group symbol with the Complex Type tool, a complex type symbol appears superposed, but not linked, to the group symbol. A global complex type cannot be the child of a group	No link
	If you click a group symbol with the Sequence tool, a sequence group particle appears linked to the group symbol	
	If you click a group symbol with the Choice tool, a choice group particle appears linked to the group symbol	
	If you click a group symbol with the All tool, an all group particle appears linked to the group symbol	

Pointer indications

When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column).
When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

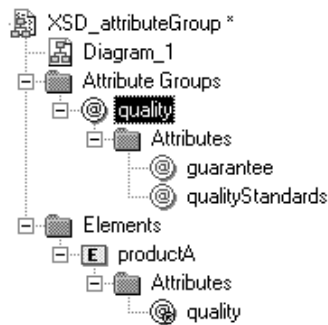
Modifying the group display preference

You can modify the Stereotype display preference for a group by selecting Tools→Display Preferences.

Defining a group of attributes

An attribute group is a set of attributes. You define an attribute group globally (with the Model menu) and then you reuse it locally (with only its reference) in the definition of an element, a complex type or another attribute group.

For example:



The quality attribute group is composed of the guarantee and qualityStandards attributes. The productA element reuses the quality attribute group via the Attributes page of its property sheet (See attribute group tools).

◆ Generated XSD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:attributeGroup name="quality">
    <xs:attribute name="guarantee">
    </xs:attribute>
    <xs:attribute name="qualityStandards">
    </xs:attribute>
  </xs:attributeGroup>
  <xs:element name="productA">
    <xs:complexType>
      <xs:attributeGroup ref="quality">
      </xs:attributeGroup>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In a schema, a group of attributes is declared with the <attributeGroup> tag. It can contain the following tags: <attribute>, <attributeGroup> or <anyAttribute>.

◆ Generated DTD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT productA EMPTY>
<!ATTLIST productA
  guarantee          CDATA
  qualityStandards  CDATA>
```

Attribute groups are not supported by XDR.

Defining attribute group properties

To display an attribute group property sheet, double-click its name or its icon in the Browser tree view.

Attribute group general properties

There are global and referencing attribute groups:




- ◆ Global attribute groups are defined with the Model menu. In a schema, they are directly linked to the <schema> tag (root element). They can be reused for any element, complex type or attribute group in the model through references (See quality attribute group in the generated schema)
- ◆ Referencing attribute groups are used within an element, a complex type or an attribute group definition. For this, you must use the attribute group tools in the Attributes page of an element, a complex type or an attribute group property sheet



The General page of an attribute group property sheet displays the following properties:

Property	Description
Name	Name of the attribute group. It must be a no-colon-name (See Glossary). Required when the attribute group is global
Code	Code of the attribute group. It must be a no-colon-name. Required when the attribute group is global
Comment	Descriptive label of the attribute group
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	Name of an attribute group from the current model or any model opened in the workspace. It must be a qualified name (See Glossary). If you select an attribute group from another model, a shortcut is created with the referencing attribute group. A reference allows you to reuse an attribute group with all its properties without having to define it again. When a reference is defined, the name and code properties are grayed. The name and code are those of the target attribute group
ID	ID of the attribute group. Its value must be of type ID and unique within the model containing this attribute group

Attribute group attributes properties

The Attributes page of an attribute group property sheet allows you to add attributes to an attribute group declaration:


Tool	Tooltip	Description
	Add Attribute	Creates a local attribute
	Add Attribute Group with Reference to Attribute Group	Adds an attribute group with a reference to an attribute group defined in the current model. Select a name in the Reference dropdown listbox. You can also type a new name in the Reference column and then define a new attribute group in the Attribute Groups list (See Model menu)
	Add Attribute with Reference to Attribute from a Selection	Adds one or several attributes with a reference to global attributes defined in the current model. Select one or several global attributes in the Selection dialog box

Tool	Tooltip	Description
	Add Attribute Group with Reference to Attribute Group from a Selection	Adds one or several attribute groups with a reference to attribute groups defined in the current model. Select one or several attribute groups in the Selection dialog box
	Any Attribute	Adds any attribute of a specified namespace

Creating an attribute group

You can create an attribute group:

- ◆ From the Browser tree view
- ◆ From the List of Attribute Groups in the Model menu

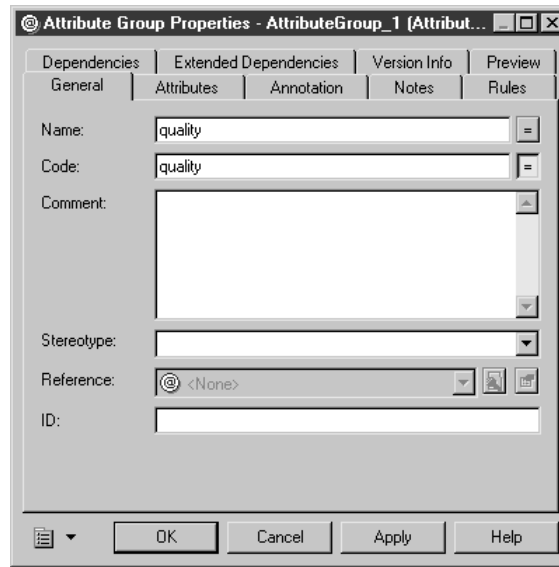
 For more information on the different ways to create an attribute group, see section Creating an object in chapter Managing objects of the *General Features Guide*.

❖ To create an attribute group from the Browser tree view:

- 1 Right-click the name or the icon of the XML model.
- 2 Select New→Attribute Group in the contextual menu.

The attribute group property sheet appears.

- 3 Type a name and a code for the attribute group.



- 4 Click OK.
- 5 Select the Attributes page.
- 6 Select different tools to add items to the attribute group.
- 7 Click Apply.
- 8 Double-click on the left of a row to display an item property sheet.
- 9 Type a name and a code for the item.
- 10 Repeat steps 8 and 9 for each item.
- 11 Click OK.

Managing external shortcuts through references and data types

External shortcuts allow you to share objects between different models. You can define external shortcuts in an XML model, but you cannot use them directly in the model, except as substitution groups for elements (see Detail page in element property sheet).

You can define external shortcuts for any global object (with no parent object in the diagram), except for imports, includes, redefines and annotations.

Internal shortcuts allow you to share objects between packages of a same model. You cannot define internal shortcuts since an XML model does not support packages.

External shortcuts are automatically generated in the following situations:

References

When you use the Reference property to define an element, an attribute, a group or an attribute group, by reference to a similar object in another model opened in the workspace, a shortcut is created between the referencing object and the target object.

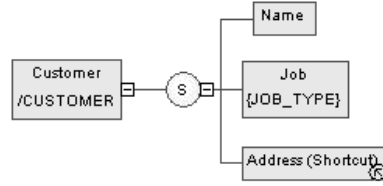
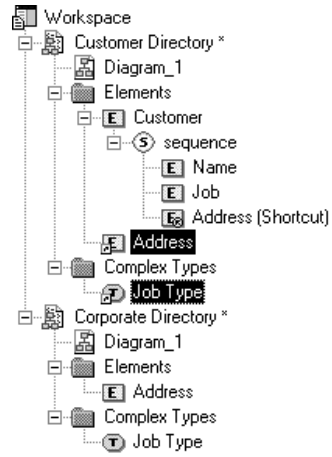
The shortcut appears in the current model with a specific item in the Browser tree view and the “(Shortcut)” expression in the reference symbol and item. The target object keeps track of the referencing object in the Reference tab of the Dependencies page of its property sheet.

Data types

When you define the data type of an element by selecting a simple or a complex type from another model (using the Browse tool beside the Type dropdown listbox), a shortcut is created between the current element type and the target data type.

The shortcut appears in the current model with a specific item in the Browser tree view.

Example of shortcuts through a reference and a data type:



Defining simple types

You can only define simple types in a model targeted with XSD.

What is a simple type?

A simple type is a data type definition for elements or attributes with text-only content.

A simple type cannot contain elements or attributes.

A simple type is defined by derivation of an existing simple type (built-in data type or derived simple type).

There are three kinds of derivation for a simple type:

Type of derivation	Description
List	The simple type contains a white space-separated list of values of an inherited simple type
Restriction	The simple type has a range of values restricted to a subset of those of an inherited simple type
Union	The simple type contains a union of values of two or more inherited simple types

For more information on simple type derivations, see section Defining derivations.

Once defined in a model, a simple type can be reused in the definition of an attribute, an element or a complex type.

Example of a simple type in a schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="BARCODE">
    <xs:restriction base="xs:nonNegativeInteger" id="STR1">
      <xs:length value="13"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="PRODUCTA" type="BARCODE"/>
  <xs:element name="PRODUCTB" type="BARCODE"/>
</xs:schema>
```

Defining simple type properties

To display a simple type property sheet, double-click its name or its icon in the Browser tree view.

The General page of a simple type property sheet displays the following properties:

Property	Description
Name	Name of the simple type. It must be a no-colon-name (See Glossary). If specified, it must be unique among all simple types and complex types
Code	Code of the simple type. It must be a no-colon-name. If specified, it must be unique among all simple types and complex types
Comment	Descriptive label of the simple type
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Derivation	Derivation method for the simple type. Enabled and required when the simple type is defined
Final	Property to prevent derivation of the current simple type
ID	ID of the simple type. Its value must be of type ID and unique within the model containing the simple type

Modifying the simple type display preference

You can modify the Stereotype display preference for a simple type by selecting Tools→Display Preferences.

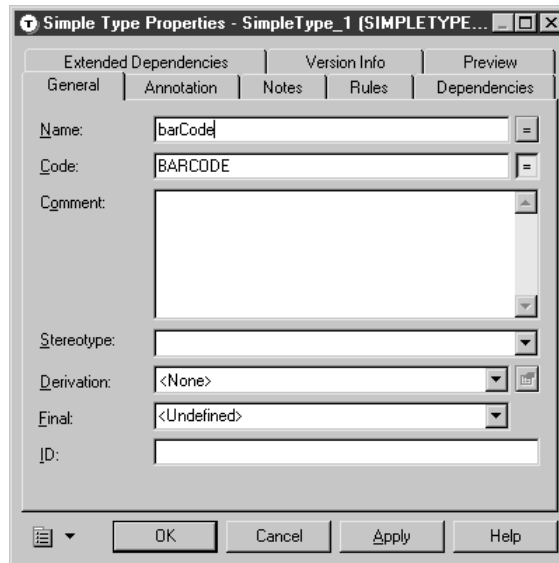
Creating a simple type

You can create a simple type:

- ◆ From the Browser tree view
 - ◆ From the Model menu
- ❖ **To create a simple type from the Browser tree view:**
- 1 Right-click the name or the icon of the model.
 - 2 Select New→Simple Type in the contextual menu.

The simple type property sheet appears.

- 3 Type a name and a code for the simple type.



Simple Type Properties - SimpleType_1 [SIMPLETYPE...]

Extended Dependencies | Version Info | Preview

General | Annotation | Notes | Rules | Dependencies

Name: barCode

Code: BARCODE

Comment:

Stereotype:

Derivation: <None>

Final: <Undefined>

ID:

OK Cancel Apply Help

- 4 Click OK.

The simple type symbol appears in the diagram.



Caution

If the simple type symbol does not appear in the diagram, select Show Symbols in the Symbol menu, then click the Simple Type tab and select the simple type box to display its symbol in the diagram.

Defining complex types

You can only define complex types in a model targeted with XSD.

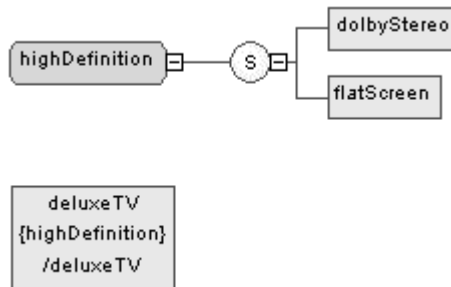
What is a complex type?

A complex type is a data type definition used to define attributes and child elements of a parent element. It is a template for a data type definition that can be reused and derived by extension or restriction.

A complex type has a global scope when it has no parent element in the diagram and when it is directly linked to the <schema> tag. It can then be reused or derived, by extension or restriction, in other parts of the schema.

It has a local scope when integrated into an <element> tag. It applies only to its containing element.

Example of a complex type in a diagram:



HighDefinition is a global complex type, reused as data type for the deluxeTV element.

Generated schema:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="highDefinition">
    <xs:sequence>
      <xs:element name="dolbyStereo"/>
      <xs:element name="flatScreen"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deluxeTV" type="highDefinition"/>
</xs:schema>

```


Caution

*Global complex types appear in the model as objects, with their corresponding symbol in the diagram.
Local complex types only appear in the schema (see Preview page of an element property sheet).*

Defining complex type properties

To display a complex type property sheet, double-click its symbol in a diagram.

Complex type general properties

The General page of a complex type property sheet displays the following properties:

Property	Description
Name	Name of the complex type. It must be a no-colon-name (See Glossary) and unique among all simple types and complex types
Code	Code of the complex type. It must be a no-colon-name and unique among all simple types and complex types
Comment	Descriptive label of the complex type
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Group type	Indicator that specifies how child elements are to be used within the complex type. It can be a group particle (all , choice , sequence) or a group of elements (group). If you select group, a referencing group is directly linked to the complex type (See Defining group properties)
Content	Content type of the complex type. If you select Complex , the complex type can contain child elements. If you select Simple , the complex type cannot contain child elements
Derivation	Derivation method for the complex type. Once you have selected a derivation method, you must define a base type. Click the Properties tool beside the derivation box to display the derivation property sheet. In the General page, select a base type in the Base Type dropdown listbox

Complex type detail properties




The Detail page of a complex type property sheet displays the following properties:



Property	Description
Final	Property to prevent derivation of the current complex type
Block	Property to prevent another complex type with the specified type of derivation from being used in place of the current complex type
Mixed	If selected, this property indicates that character data is allowed to appear between the child elements of the current complex type. Select Mixed only if the current complex type has a complex content (See general properties)
Abstract	If selected, this property indicates that the complex type can be used in the instance document
ID	ID of the complex type. Its value must be of type ID and unique within the model containing this complex type

Complex type attributes properties

Attributes are used to give additional information about complex types.

The Attributes page of a complex type property sheet allows you to add attributes to a complex type declaration:

Tool	Tooltip	Description
	Add Attribute	Creates a local attribute
	Add Attribute Group with Reference to Attribute Group	Adds an attribute group with a reference to an attribute group defined in the current model. Select a name in the Reference dropdown listbox. You can also type a new name in the Reference column and then define a new attribute group in the Attribute Groups list (See Model menu)
	Add Attribute with Reference to Attribute from a Selection	Adds one or several attributes with a reference to global attributes defined in the current model. Select one or several global attributes in the Selection dialog box

Tool	Tooltip	Description
	Add Attribute Group with Reference to Attribute Group from a Selection	Adds one or several attribute groups with a reference to attribute groups defined in the current model. Select one or several attribute groups in the Selection dialog box
	Any Attribute	Adds any attribute of a specified namespace

You can access directly to the Attributes page of a complex type property sheet through the contextual menu. Right-click a complex type symbol in the diagram and select Attributes in the contextual menu.

Complex type mapping properties

Object mapping is the ability to establish a correspondence between objects belonging to heterogeneous models and diagrams.

The Mapping page of a complex type property sheet allows you to map the current complex type to PDM or OOM objects.

Select a data source in the Mapping for dropdown listbox. If it is the first time you define a mapping for a complex type, the Mapping for dropdown listbox is empty. Click the Add a Mapping for a Data Source tool and select a data source.


Complex Type Sources page



The Complex Type Sources page allows you to associate one or several abstract data types (in PDMs) or classes (in OOMs) to the current complex type.

You can use the Add Objects tool to select abstract data types or classes from the PDMs or OOMs opened in the current workspace.

Attributes Mapping page

The Attributes Mapping page allows you to define the mapping between abstract data type attributes (in PDMs) or class attributes (in OOMs) and attributes in the current complex type.

Icon	Tool	Description
	Add Mapping	To select the attributes in the current complex type that will be mapped to abstract data type attributes or class attributes. Once you have selected the attributes, you can use the dropdown listbox in the Mapped to column to select corresponding abstract data type attributes or class attributes

Icon	Tool	Description
	Create from Sources	To copy abstract data type attributes or class attributes to the current complex type attributes
	Generate Mapping	To automatically generate a mapping between abstract data type attributes or class attributes and complex type attributes with same name or code in the data source and the current model

For more information on complex type mapping, see section Mapping objects in an XML model in chapter Working with an XML model.

Modifying complex type display preferences

You can modify the following display preferences for a complex type by selecting Tools→Display Preferences:

Preference	Description
Attributes	Displays attributes and attribute values of the complex type
Display limit	Maximum number of attributes displayed
Stereotype	Displays the stereotype of the complex type
Type	Displays the data type of the complex type
Complex Type Attributes	Select Type, if you want the complex type attributes data types to be displayed

Creating a complex type

You can create a complex type:

- ◆ From the palette
- ◆ From the Browser tree view
- ◆ From the List of Complex Types in the Model menu

For more information on the different ways to create a complex type, see section Creating an object in chapter Managing objects of the *General Features Guide*.

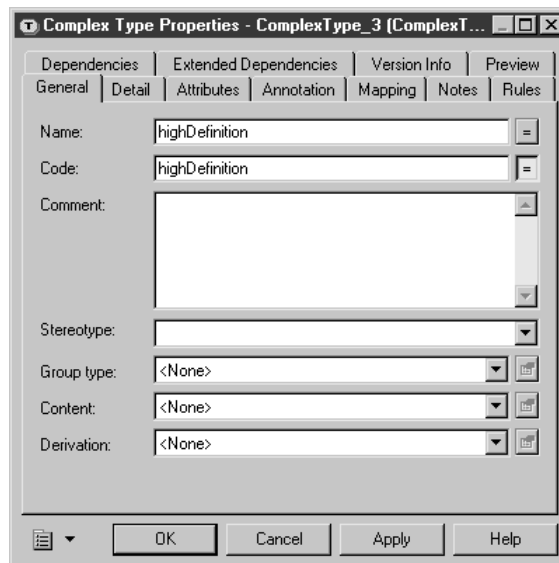
❖ **To create a complex type from the palette:**

- 1 Select the Complex Type tool in the palette.
- 2 Click an empty space in the diagram.

A complex type symbol appears in the diagram at the click position.



- 3 Click the Pointer tool in the palette.
or
Right-click to recover the Pointer.
- 4 Double-click the complex type symbol in the diagram.
The complex type property sheet appears.
- 5 Type a name and a code for the complex type.

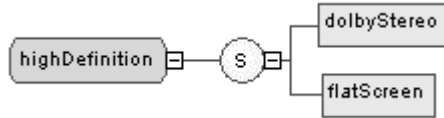


- 6 Click OK.
- 7 Select a group particle tool in the palette (Sequence, Choice or All).
- 8 Click the complex type symbol in the diagram.
The group particle symbol appears attached to the complex type symbol.
- 9 Select the Element tool in the palette.

- 10 Click the group particle symbol in the diagram for each child element you want to create.


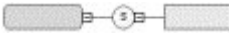


The child element symbols appear attached to the group particle symbol.










- 11 Double-click a child element symbol to display its property sheet.
- 12 Type a name and a code for the child element.
- 13 Repeat steps 11 and 12 for each child element.
- 14 Click OK.



How to link a child object to a complex type?

XML objects do not support standard link objects. To link a child object to a complex type, you must click the child object tool in the palette and then click the complex type symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action	Result
	<p>If you click a complex type symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet</p>	
	<p>If you click a complex type symbol with the Any tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet</p>	

Tool	Action	Result
	If you click a complex type symbol with the Group tool, a referencing group is created. You can modify the group particle via its property sheet. You must now select a group for the reference	
	If you click a complex type symbol with the Complex Type tool, a second complex type symbol appears superposed, but not linked, to the first complex type symbol. A complex type cannot be the child of another complex type	No link
	If you click a complex type symbol with the Sequence tool, a sequence group particle appears linked to the complex type symbol	
	If you click a complex type symbol with the Choice tool, a choice group particle appears linked to the complex type symbol	
	If you click a complex type symbol with the All tool, an all group particle appears linked to the complex type symbol	

Pointer indications

When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column).
When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Defining simple content properties

Simple content general property

A simple content allows you to extend or restrict the values of a complex type supporting character data or a simple type.

It cannot contain elements.

The General page of a simple content property sheet allows you to define an id for the simple content.

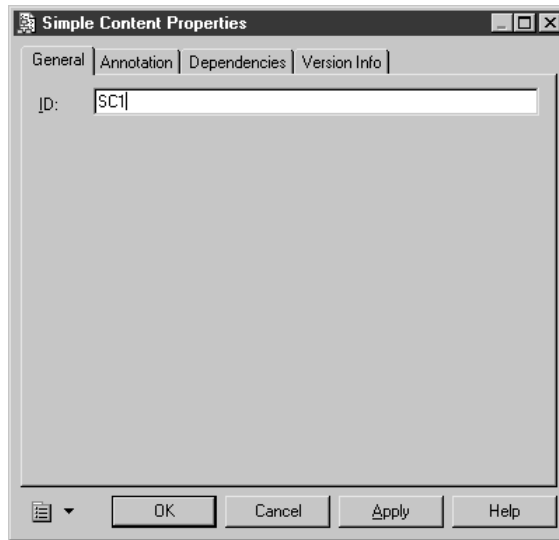
The id value must be of type ID and unique within the model containing the simple content.

❖ To create a simple content:

- 1 Select Simple in the Content dropdown listbox of a complex type property sheet.
- 2 Click Apply.
- 3 Click the Properties tool beside the Content dropdown listbox.

The simple content property sheet appears.

- 4 Type an ID for the simple content.



- 5 Click OK.

Defining complex content properties

Complex content general properties

A complex content allows you to extend or restrict the values of a complex type with mixed content (character data and elements) or elements only.

The General page of a complex content property sheet displays the following properties:

Property	Description
ID	ID for the complex content. Its value must be of type ID and unique within the model containing the complex content
Mixed	An indicator of whether character data is allowed to appear between child elements of the complex type

❖ To create a complex content:

- 1 Select Complex in the Content dropdown listbox of a complex type property sheet.
- 2 Click Apply.
- 3 Click the Properties tool beside the Content dropdown listbox.

The complex content property sheet appears.

- 4 Type an ID for the complex content and check the Mixed box, if needed.



- 5 Click OK.

Defining derivations

You use derivations when you want to extend or restrict the values of simple and complex types.

An XML model allows you to derive:

- ◆ Simple types by restriction, list or union
- ◆ Complex types by extension or restriction

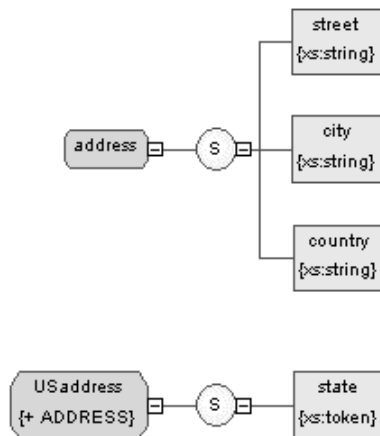
Derivation in element property sheet

When you define a derivation in an element property sheet, a simple or a complex type is automatically created within the element declaration (See Preview page). The Embedded type property is automatically set to Simple or Complex, and the Content property to Simple or Complex in the case of an embedded complex type.

Deriving by extension

You derive a complex type by extension when you want to extend the values of its base type.

For example:



USaddress is a derivation by extension of the address complex type.

Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="ADDRESS">
    <xs:sequence>
      <xs:element name="STREET" type="xs:string"/>
      <xs:element name="CITY" type="xs:string"/>
      <xs:element name="COUNTRY" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="USADDRESS">
    <xs:complexContent>
      <xs:extension base="ADDRESS">
        <xs:sequence>
          <xs:element name="STATE" type="xs:token"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Defining extension
general properties

The General page of an extension property sheet displays the following properties:

Property	Description
ID	ID of the extension. Its value must be of type ID and unique within the model containing the extension
Base Type	Data type on which the extension is based

❖ **To define a derivation by extension:**

- 1 Select Extension in the Derivation dropdown listbox of an element or a complex type property sheet.

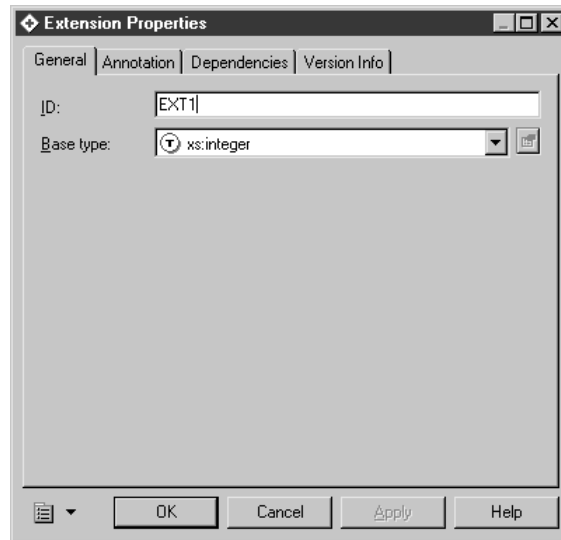
In the case of an element property sheet, the Embedded type and Content boxes are set to Complex.

In the case of a complex type property sheet, the Content box is set to Complex.

- 2 Click Apply.
- 3 Click the Properties tool beside the Derivation box.

The extension property sheet appears.

- 4 Type an ID and select a base type.



- 5 Click OK.

Deriving by restriction

You derive a simple type or a complex type by restriction when you want to restrict the values of their base type.

Restriction on a simple type

The property sheet of a simple type restriction displays several specific pages:

- ◆ General
- ◆ Detail
- ◆ Enumerations
- ◆ Patterns

Detail, Enumerations and Patterns pages are described in the next section, Defining restriction facets.

Restriction general properties

The General page of a simple type restriction property sheet displays the following properties:

Property	Description
ID	ID of the simple type restriction. Its value must be of type ID and unique within the model containing the simple type restriction
Base type	Data type on which the restriction is based. Select a data type in the Base type dropdown listbox or with the Browse tool
Embedded type	If selected, the base type disappears and a simple type is created in the schema within the current simple type









Defining restriction facets



Facets are the constraints on the set of values of a simple type.

You can find these facets in the Detail, Enumerations and Patterns pages of a simple type restriction property sheet.

Restriction detail properties

The Detail page of a simple type restriction property sheet displays a range of facets:

Icon	Facet	Description
	Length	Exact number of characters or list items allowed. It must be equal to or greater than zero
	Minimum length	Minimum number of characters or list items allowed. It must be equal to or greater than zero
	Maximum length	Maximum number of characters or list items allowed. It must be equal to or greater than zero
	Minimum exclusive	Lower bound for numeric values. All values are greater than this value
	Maximum exclusive	Upper bound for numeric values. All values are lower than this value
	Minimum inclusive	Minimum value allowed for data type
	Maximum inclusive	Maximum value allowed for data type
	Total digits	Exact number of decimal digits allowed. It must be greater than zero

Icon	Facet	Description
	Fraction digits	Maximum number of decimal digits in the fractional part
	Whitespace	Way of handling white spaces. If the value is Preserve , white spaces are unchanged. If the value is Replace , tabs, line feeds and carriage returns are replaced with spaces. If the value is Collapse , contiguous sequences of spaces are collapsed to a single space. Leading and trailing spaces are removed

A facet icon appears in the title bar of a facet property sheet.

Caution

Facets only appear in the schema, within a simple type declaration (see Preview tab in the model property sheet).

Facet general properties

The General page of a facet property sheet displays the following properties:

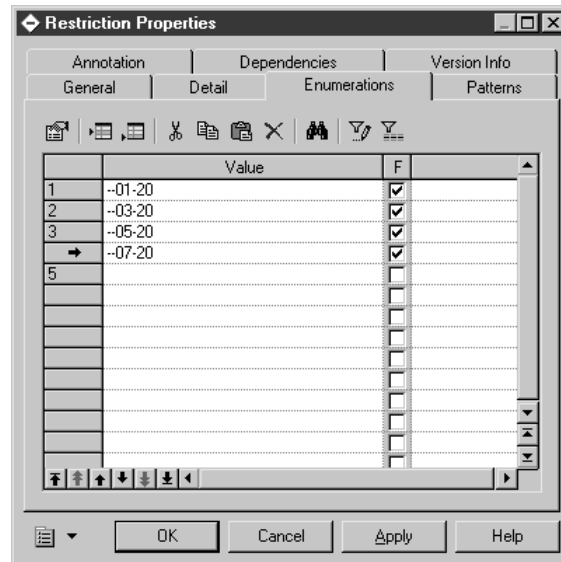
Property	Description
ID	ID of the facet. Its value must be of type ID and unique within the model containing the facet
Value	Value(s) of the facet
Fixed (check box)	To prevent a modification of the facet value(s), select the Fixed property

Restriction enumerations properties

The Enumerations page of a simple type restriction property sheet allows you to enter a set of acceptable values for the simple type restriction.

Select F (for Fixed) at the end of a row if you want to prevent the modification of a value.

For example: the meetings simple type, based on the xs:gMonthDay data type, is restricted to the following dates: 01/20, 03/20, 05/20 and 07/20.



Generated schema:

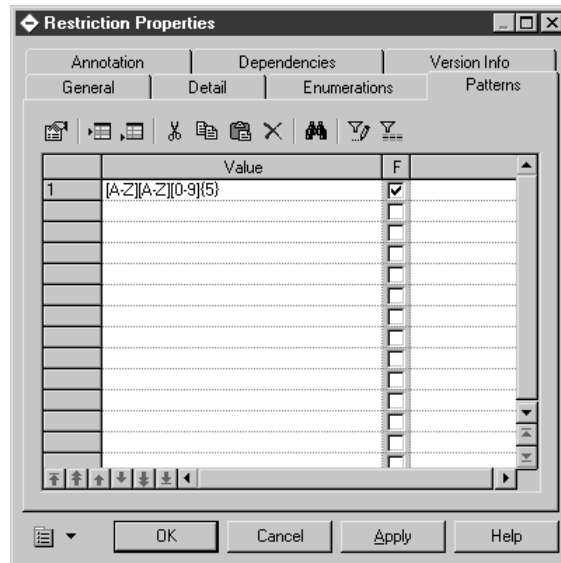
```
<xs:simpletype name="MEETINGS">
  <xs:restriction base="xs:gMonthDay">
    <xs:enumeration value="--01-20"/>
    <xs:enumeration value="--03-20"/>
    <xs:enumeration value="--05-20"/>
    <xs:enumeration value="--07-20"/>
  </xs:restriction>
</xs:simpletype>
```

Restriction patterns properties

The Patterns page of a simple type restriction property sheet allows you to enter the exact sequence of acceptable values for the simple type restriction.

Select F (for Fixed) at the end of a row if you want to prevent the modification of a value.

For example: the zipCode simple type, based on the xs:string data type, is restricted to the following pattern: two uppercase letters, from A to Z, followed by a five-digit number, each digit ranging from 0 to 9.



Generated schema:

```
<xs:simpleType name="ZIPCODE">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Z][A-Z][0-9]{5}"/>
  </xs:restriction>
</xs:simpleType>
```

Creating a restriction on a simple type

You derive a simple type by restriction when you want to restrict the values of its base type.

❖ **To create a restriction on a simple type:**

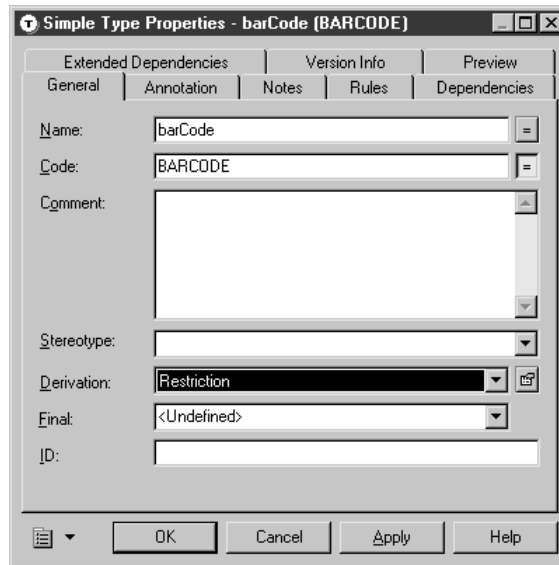
- 1 Double-click the name or the icon of a simple type in the Browser tree view.

or

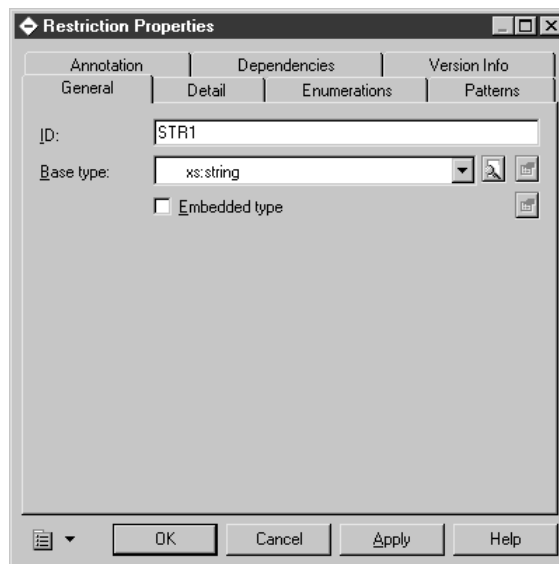
Double-click the symbol of a simple type in the diagram.

The simple type property sheet appears.

- 2 Select Restriction in the dropdown listbox of the Derivation box.

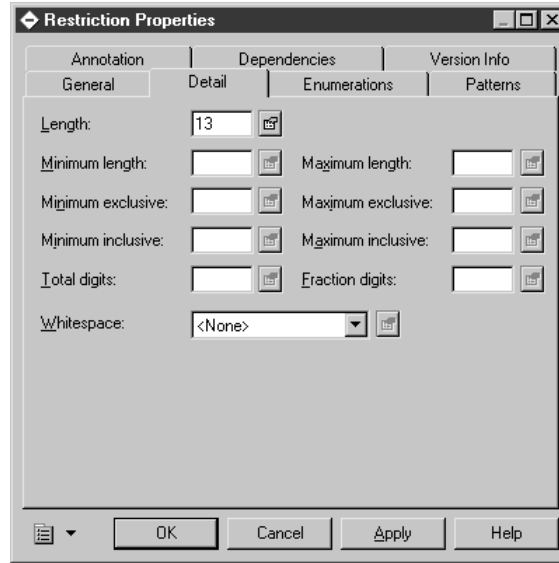


- 3 Click Apply.
- 4 Click the Properties tool beside the Derivation box.
The restriction property sheet appears.
- 5 Type an ID and a base type for the simple type restriction.



If you select Embedded type, the base type disappears and a simple type is created in the schema within the current simple type. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a base type for the embedded simple type.

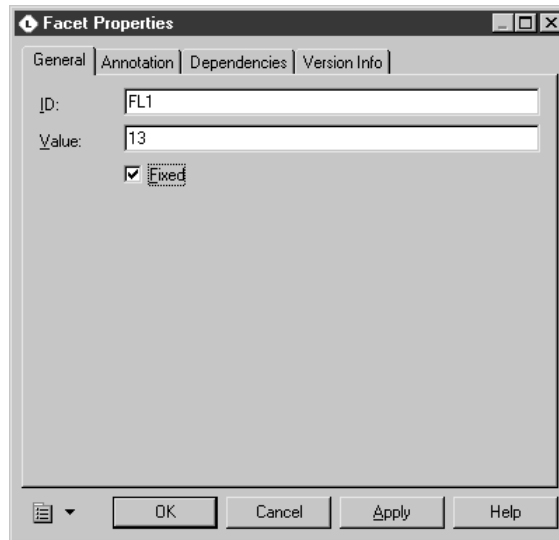
- 6 Click the Detail tab to display the Detail page.
- 7 Type a value in a facet box.



- 8 Click Apply.
- 9 Click the Properties tool beside the facet box.

The facet property sheet appears (the length property sheet in the example).

- 10 Type an ID and select the Fixed box, if needed.



- 11 Click OK.

Enumeration and Pattern facets

For Enumeration and Pattern facets, click their corresponding tabs in the restriction property sheet and double-click the arrow left of an enumeration or a pattern value to display its property sheet.

Restriction on a complex type

The General page of a complex type restriction property sheet displays the following properties:

Property	Description
ID	ID of the complex type restriction. Its value must be of type ID and unique within the model containing the complex type restriction
Base Type	Data type on which the restriction is based

❖ To create a restriction on a complex type:

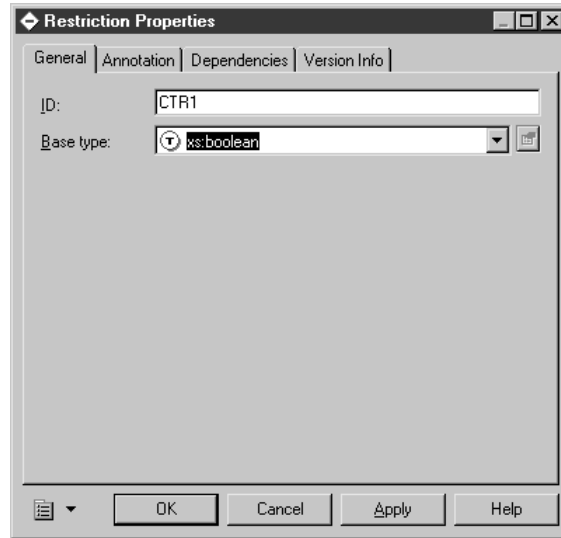
- 1 Select Restriction in the Derivation dropdown listbox of a complex type property sheet.

The Content box is set to Complex.

- 2 Click Apply.
- 3 Click the Properties tool beside the Derivation box.

The restriction property sheet appears.

- 4 Type an ID and a base type for the complex type restriction.



- 5 Click OK.

Deriving by list

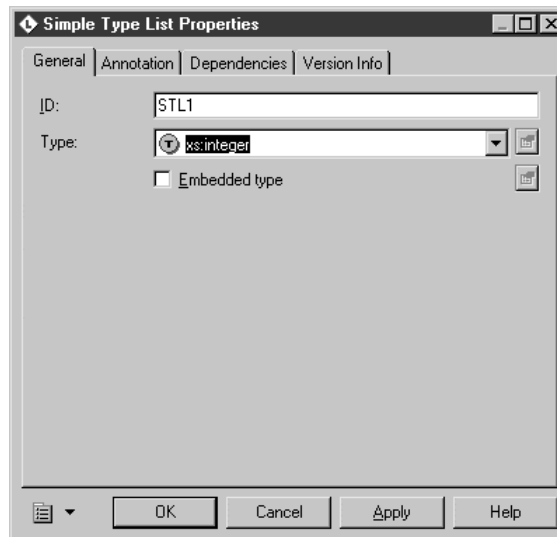
You derive a simple type by list when you want to define it as a list of values of a specified data type.

The General page of a simple type list property sheet displays the following properties:

Property	Description
ID	ID of the simple type list. Its value must be of type ID and unique within the model containing the simple type restriction
Type	Data type for the list of values
Embedded Type	If selected, the type disappears and a simple type is created in the schema within the current simple type

❖ **To define a simple type derivation by list:**

- 1 Select List in the Derivation dropdown listbox of a simple type property sheet.
- 2 Click Apply.
- 3 Click the Properties tool beside the Derivation box.
The simple type list property sheet appears.
- 4 Enter an ID and a type for the simple type list.



If you select Embedded type, the type disappears and a simple type is created in the schema within the current simple type. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a type for the embedded simple type.

- 5 Click OK.

Deriving by union

You derive a simple type by union when you want to define it as a collection of built-in and simple data types.

Union general properties

The General page of a simple type union property sheet displays the following properties:

Property	Description
ID	ID of the simple type union. Its value must be of type ID and unique within the model containing the simple type union
Member Types	White space separated list of built-in data types. Values must be qualified names (See Glossary)

Union types properties

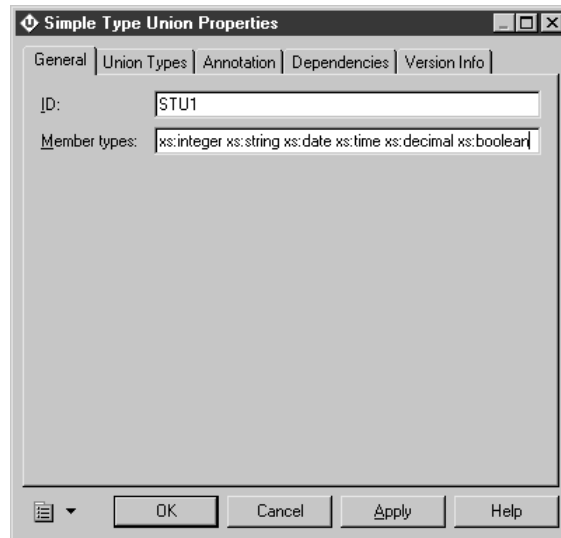
The Union Types page of a simple type union property sheet displays a list where you can add simple types using the Add Simple Type tool.

❖ **To define a simple type derivation by union:**

- 1 Select Union in the Derivation dropdown listbox of a simple type property sheet.
- 2 Click Apply.
- 3 Click the Properties tool beside the Derivation box.

The simple type union property sheet appears.

- 4 Type an ID and member types for the simple type union.



- 5 Click OK.

Defining annotations

You define annotations when you want to add information about an XML model.

Annotations are made of Documentation and Application Information:

- ◆ Documentation allows you to write a URI reference or any well-formed XML content that will give extra information about XML objects or documents
- ◆ Application Information allows you to write a URI reference or any well-formed XML content that will be used by applications for processing instructions

There are two kinds of annotations:

- ◆ Local annotations, for extra information on XML objects. You can define them through the Annotation page of an object property sheet. They have no property sheet
- ◆ Global annotations, for extra information on XML models or schemas. You can define them through the Items page or the External Schemas page of a model property sheet. They have a property sheet

Generated schema of a global annotation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:annotation id="ANNOT1">
    <xs:documentation source="attributes.dtd" xml:lang="en-US">
      <?xml version="1.0" encoding="UTF-8" ?>
      <!ELEMENT COMPANY (EMPLOYEES,PRODUCTS,CLIENTS)>
      <!-- -->
      <!ATTLIST COMPANY
        FOUNDATION          CDATA
        STATUS               CDATA>
      <!ELEMENT EMPLOYEES EMPTY>
      <!-- -->
      <!ATTLIST EMPLOYEES
        NUMBER              CDATA>
      <!ELEMENT PRODUCTS EMPTY>
      <!-- -->
      <!ATTLIST PRODUCTS
        REFERENCES          CDATA
        QUANTITIES         CDATA>
      <!ELEMENT CLIENTS EMPTY>
      <!-- -->
      <!ATTLIST CLIENTS
        FIDELITY           CDATA
        SOLVENCY           CDATA>
    </xs:documentation>
    <xs:appinfo source="http://www.parsersandco.com"></xs:appinfo>
  </xs:annotation>
</xs:schema>
```

This global annotation is composed of a documentation, with a well-formed XML content (extract of a DTD file), and an application information.

Annotations are only available in models targeted with XSD.

Defining annotation properties

To display an annotation property sheet, double-click its name or icon in the Browser tree view.



Annotation general property

The General page of an annotation property sheet displays the ID of the annotation.

The id value must be of type ID and unique within the model containing the annotation.

Annotation items properties

The Items page of an annotation property sheet displays a list where you can add the following items:

Tool	Tooltip	Description
	Documentation	Adds a comment or a document reference to be read by users
	Application Information	Adds an information to be used by applications for processing instructions

Local annotations

The Annotation page of an object property sheet displays the same items list.

The ID box allows you to give an id to the local annotation. Its value must be of type ID and unique within the model containing the local annotation.

Defining documentation properties

Documentation allows you to write a URI reference or any well-formed XML content that will give extra information about XML objects or documents.

You can display a documentation property sheet:

- ◆ By double-clicking its name or icon in the Browser tree view
- ◆ By double-clicking left of its row in the Items page of an annotation property sheet

Documentation general properties

The General page of a documentation property sheet displays the following properties:

Property	Description
Source	Source of the documentation. It must be a URI reference
Language	Language used in the documentation. For example: en, en-GB, en-US, de, fr

Documentation content

The Content page of a documentation (or an application information) property sheet allows you to write or paste any well-formed XML content.

Defining application information properties

Application Information allows you to write a URI reference or any well-formed XML content that will be used by applications for processing instructions.

You can display an application information property sheet:

- ◆ By double-clicking its name or icon in the Browser tree view
- ◆ By double-clicking left of its row in the Items page of an annotation property sheet

Application information general properties

The General page of an application information property sheet displays the source of the application information.

The source value must be a URI reference.

Application information content

The Content page of an application information property sheet allows you to write or paste any well-formed XML content.

Creating an annotation

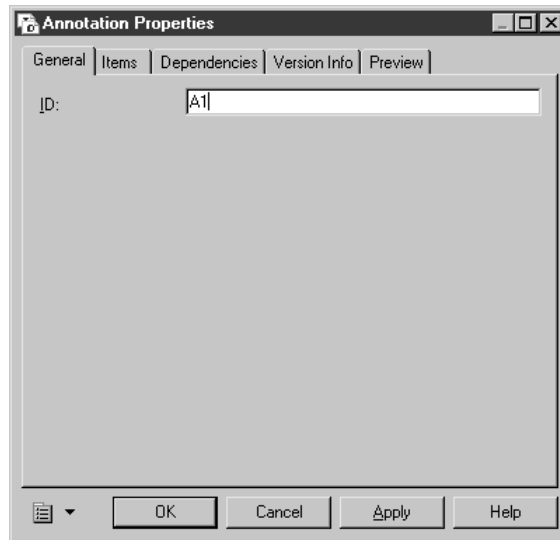
You can create a global annotation from the Items page or the External Schemas page of a model property sheet.

❖ To create a global annotation from the Items page of a model property sheet:

- 1 In the Browser tree view, double-click the name or the icon of a model to display its property sheet.
or
Select Model→Model Properties to display the current model property sheet.
- 2 Select the Items page or the External Schemas page.
- 3 Click the Add Annotation tool.
- 4 Click Apply.
- 5 Double-click the arrow left of the annotation line.

The annotation property sheet opens at the General page.

- 6 Type an ID for the annotation.



- 7 Click OK.

Defining notations

Notations allow you to define and process non-XML objects within an XML model.

For example: picture files with a .GIF extension.

Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!--
    Integrating GIF files in your XML model
  -->
  <xs:notation name="PICTURES" public="pictures/gif"
    system="user/local/pictureViewer"/>
</xs:schema>
```

Notations are not available on models targeted with XDR.

Defining notation properties

To display a notation property sheet, double-click its name or its icon in the Browser tree view.

The General page of a notation property sheet displays the following properties:

Property	Description
Name	Name of the notation. It must be a no-colon-name (See Glossary)
Code	Code of the notation. It must be a no-colon-name
Comment	Descriptive label of the notation
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the notation. Its value must be of type ID and unique within the model containing the notation
Public	URI reference identifying the non-XML object. For example: pictures/gif
System	URI reference identifying the application that will process the non-XML object. For example: user/local/pictureViewer

Creating a notation

You can create a notation:

- ◆ From the Browser tree view
- ◆ From the List of Notations in the Model menu

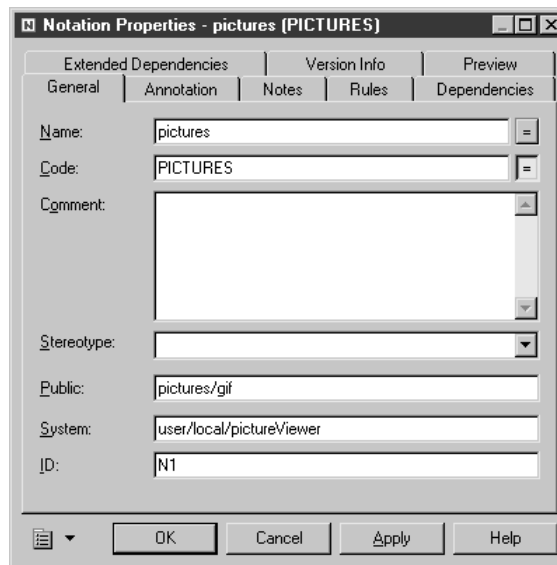
❖ **To create a notation from the Browser tree view:**

1 Right-click the name or the icon of the model to display the contextual menu.

2 Select New→Notation in the contextual menu.

The notation property sheet appears.

3 Type a name, a code and other properties for the notation.



The screenshot shows a dialog box titled "Notation Properties - pictures [PICTURES]". It has three tabs: "Extended Dependencies", "Version Info", and "Preview". The "General" tab is selected, and it contains several fields: "Name" with the value "pictures", "Code" with "PICTURES", "Comment" (empty), "Stereotype" (empty), "Public" with "pictures/gif", "System" with "user/local/pictureViewer", and "ID" with "N1". At the bottom, there are buttons for "OK", "Cancel", "Apply", and "Help".

4 Click OK.

Defining entities

Entities enable you to include predefined values, external XML or non-XML files in an XML model targeted with a DTD.

When an XML processor reads an entity reference in an XML document, it will replace this entity reference by its value defined in the DTD file of the XML document.

An entity reference is the entity name preceded by an ampersand and followed by a semicolon.

For example: `&glossary;` will be replaced by See Glossary.

The W3C has predefined five entities for XML tags:

Entity name	Reference	Value
Less than	<code>&lt;</code>	<
Greater than	<code>&gt;</code>	>
Ampersand	<code>&amp;</code>	&
Apostrophe	<code>&apos;</code>	'
Quotation	<code>&quot;</code>	"

In an XML model, you just need to type the name and the value of an entity.

Defining entity properties

To display an entity property sheet, double-click its name or its icon in the Browser tree view.

The General page of an entity property sheet displays the following properties:

Property	Description
Name	Name of the entity. It must be a no-colon-name (See Glossary)
Code	Code of the entity. It must be a no-colon-name
Comment	Descriptive label of the entity
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

Property	Description
Value	Value of the entity. A string of characters in the case of a predefined value. A URI in the case of an XML or a non-XML file. For example: <code>http://something.com/pictures/logo.gif</code>
Public	URI reference identifying the non-XML object. For example: <code>pictures/gif</code>
System	URI reference identifying the application that will process the non-XML object. For example: <code>user/local/pictureViewer</code>
Notation	Used to define and process non-XML objects within an XML model
Parameter	If selected, the entity is parsed within the DTD, and not within the XML document as for a general entity. A parameter entity allows you to predefine a value within a DTD. This predefined value can then be easily changed within the DTD

Creating an entity

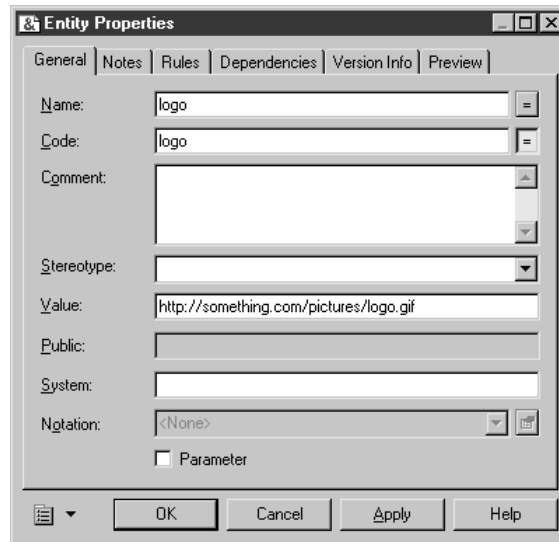
You can create an entity:

- ◆ From the Browser tree view
- ◆ From the List of Entities in the Model menu

❖ To create an entity from the Browser tree view:

- 1 Right-click the name or the icon of the model to display the contextual menu.
- 2 Select New→Entity in the contextual menu.
The entity property sheet appears.
- 3 Type a name and a code for the entity.

- 4 Type a value.



- 5 Click OK.

Defining import, include and redefine

Import, Include and Redefine allow you to enrich your XML model with external namespaces, schema files or schema components.

Import, Include and Redefine are only available in a model targeted with XSD.

Defining an import

An import identifies a namespace whose schema components are referenced by the current schema.

With an import, you can use schema components from any schema with different target namespace than the current schema.

In a schema, an import is declared with the `<import>` tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import id="IMP1" namespace="xml.ordering"
    schemaLocation="ORDER.xsd"/>
</xs:schema>
```

Defining import properties

To display an import property sheet, double-click its name or its icon in the Browser tree view.

Import general properties

The General page of an import property sheet displays the following properties:

Property	Description
Schema location	URI reference for the location of a schema file with an external namespace. You can use the Browse tool beside the Properties tool to select a schema file among those opened in the current workspace. For example: ORDER.xsd
ID	ID of the import. Its value must be of type ID and unique within the schema containing the import
Namespace	URI reference for the namespace to import. For example: xml.ordering

Property	Description
Comment	Descriptive label of the import
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

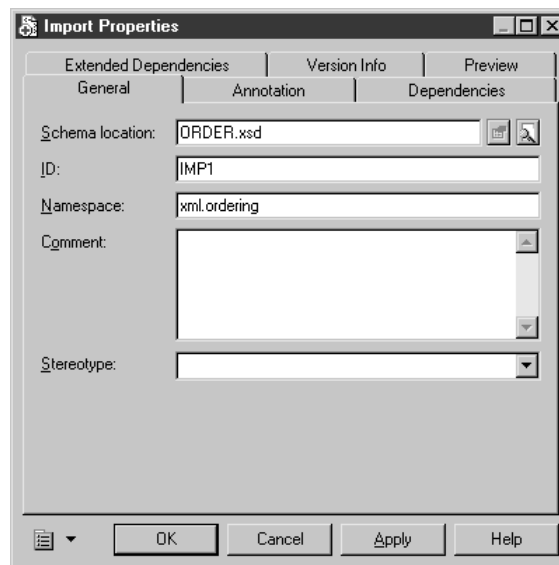
Creating an import

You can create an import:

- ◆ From the Browser tree view
- ◆ From the List of Imports in the Model menu

❖ To create an import from the Browser tree view:

- 1 Right-click the name or the icon of the model to display the contextual menu.
- 2 Select New→Import in the contextual menu.
The import property sheet appears.
- 3 Type a schema location, an ID and a namespace.



- 4 Click OK.

Defining an include

An include allows you to include a specified schema file in the target namespace of the current schema.

With an include, you can use schema components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, an include is declared with the <include> tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include id="INCL" schemaLocation="PROFORMA.xsd"/>
</xs:schema>
```

Defining include properties

To display an include property sheet, double-click its name or its icon in the Browser tree view.

Include general properties

The General page of an include property sheet displays the following properties:

Property	Description
Schema location	URI reference for the location of a schema file with the same target namespace or no specified target namespace. You can use the Browse tool beside the Properties tool to select a schema file among those opened in the current workspace. For example: PROFORMA.xsd
ID	ID of the include. Its value must be of type ID and unique within the schema containing the include
Comment	Descriptive label of the include
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

Creating an include

You can create an include:

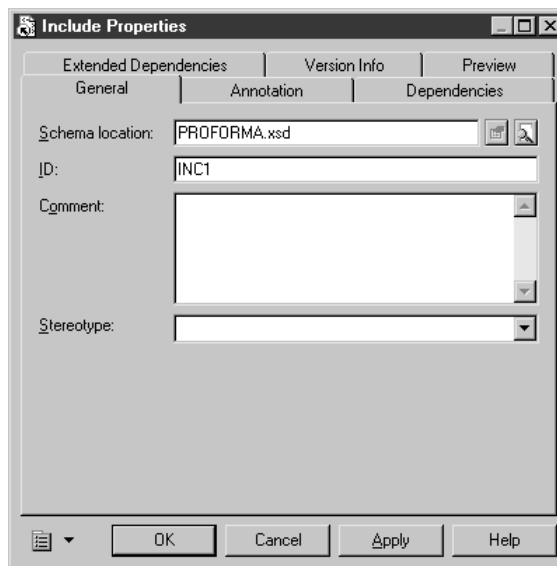
- ◆ From the Browser tree view
- ◆ From the List of Includes in the Model menu

❖ To create an include from the Browser tree view:

- 1 Right-click the name or the icon of the model to display the contextual menu.
- 2 Select New→Include in the contextual menu.

The include property sheet appears.

- 3 Type a schema location and an ID.



- 4 Click OK.

Defining a redefine

A redefine allows you to redefine simple and complex types, groups and attribute groups from an external schema file in the current schema.

With a redefine, you can use schema components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, a redefine is declared with the <redefine> tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:redefine id="REDEF1" schemaLocation="CUSTOMERS.xsd">
    <xs:group name="PRIVILEGED_CUSTOMERS">
      <xs:all>
        <xs:element name="CUSTOMER_1"/>
        <xs:element name="CUSTOMER_2"/>
        <xs:element name="CUSTOMER_3"/>
      </xs:all>
    </xs:group>
    <xs:complexType name="PRIVILEGE">
      </xs:complexType>
    </xs:redefine>
  </xs:schema>
```

Defining redefine properties

To display a redefine property sheet, double-click its name or its icon in the Browser tree view.





Redefine general properties

The General page of a redefine property sheet displays the following properties:

Property	Description
Schema location	URI reference for the location of a schema file with the same target namespace or no specified target namespace. You can use the Browse tool beside the Properties tool to select a schema file among those opened in the current workspace. For example: customers.xsd
ID	ID of the redefine. Its value must be of type ID and unique within the schema containing the redefine
Comment	Descriptive label of the redefine
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined

Redefine items properties

The Items page of a redefine property sheet displays a list of items to be redefined.

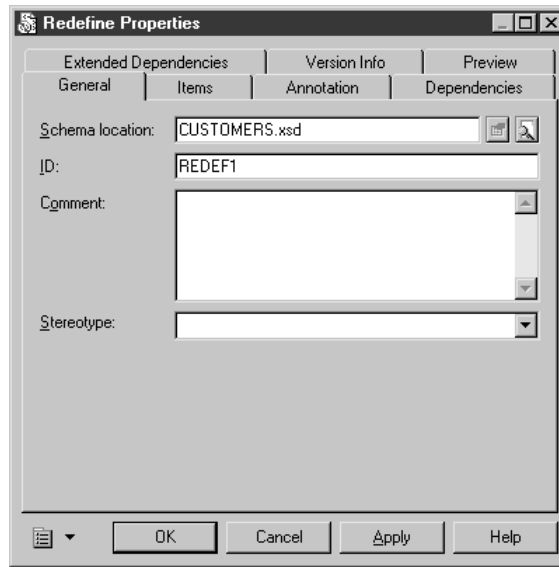
Tool	Tooltip	Description
	Group	Adds a group of elements to be redefined
	Attribute Group	Adds a group of attributes to be redefined
	Simple Type	Adds a simple type to be redefined
	Complex Type	Adds a complex type to be redefined

Creating a redefine

You can create a redefine:

- ◆ From the Browser tree view
- ◆ From the List of Redefines in the Model menu
- ❖ **To create a redefine from the Browser tree view:**
 - 1 Right-click the name or the icon of the model to display the contextual menu.
 - 2 Select New→Redefine in the contextual menu.
The redefine property sheet appears.

- 3 Define redefine properties in the different pages.



- 4 Click OK.

Defining business rules

You can define business rules in an XML model and attach them to objects.

What is a business rule?

A business rule is a written statement specifying what an XML model must contain or how it must be structured to support business needs.

A business rule is a rule that your business follows. It could be a government-imposed law, a customer requirement, or an internal guideline.

Modeling guides

Business rules guide and document the creation of an XML model.

Graphics complements

Business rules complement model graphics with information that is not easily represented graphically.

Defining business rule properties

To display a business rule property sheet, double-click its name or its icon in the Browser tree view.

Business rule general properties

The General page of a business rule property sheet displays the following properties:

Property	Description
Name	Name of the rule. It must be a no-colon-name (See Glossary)
Code	Code of the rule. It must be a no-colon-name
Comment	Descriptive label of the rule
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Type	Constraint, definition, fact, formula, requirement or validation

You can define several types of business rules in an XML model:

Type	Description	Example
Constraint	Additional check constraint on a value	The start date should be inferior to the end date of a project
Definition	Properties of an element in the XML model	A customer is a person identified by a name and an address
Fact	Certainty, existence in the XML model	A client may place one or more orders
Formula	Calculation used in the XML model	The total order is the sum of all the order line costs
Requirement	Functional specification in the XML model	The model is designed so that total losses do not exceed 10% of total sales
Validation	Constraint on a value in the XML model	The sum of all orders for a client must not be greater than the client's allowance

Business rule expression property

The Expression page of a business property sheet allows you to complete a rule by adding a technical expression.

There are two types of expression for a business rule:

- ◆ Server that can be generated to a database
- ◆ Client that is used mainly for documentation purposes

Expressions are mainly used in a CDM or a PDM.

Creating a business rule

You can create a business rule in different ways:

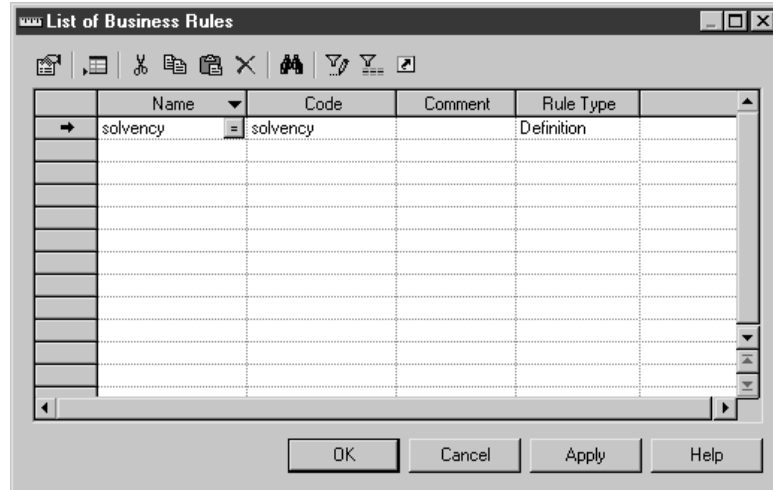
- ◆ From the Browser tree view
- ◆ From the List of Business Rules

❖ To create a business rule from the List of Business Rules:

- 1 Select Model→Business Rules to display the List of Business Rules.
- 2 Click the Add a Row tool.
or
Click a blank line in the list.

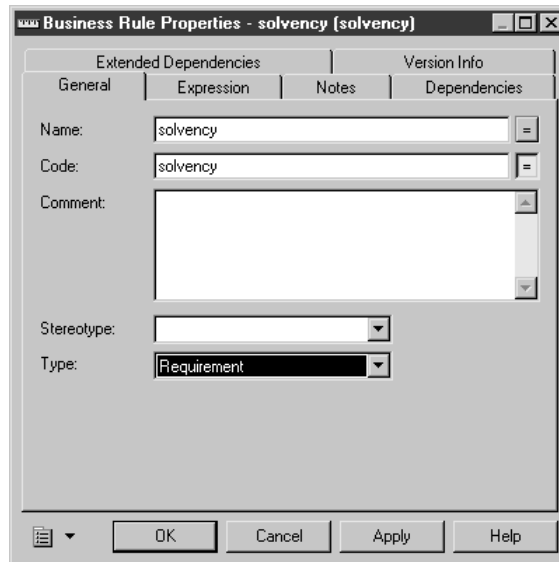
A rule appears with generic name, code and rule type (Definition, by default).

- 3 Type a name and a code for the business rule.



- 4 Click Apply.
- 5 Double click the arrow at the beginning of the line.
The business rule property sheet appears.

- 6 In the General page, select a business rule type in the Type dropdown listbox.



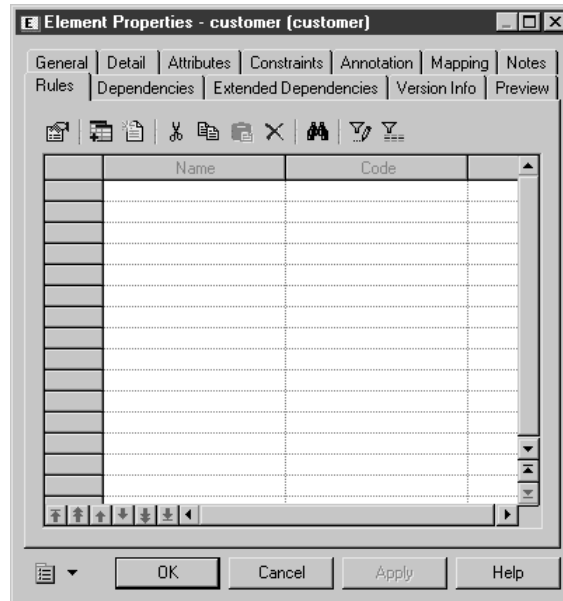
- 7 Click OK.

Applying a business rule to an XML object

Once you have defined a business rule in the List of Business Rules, you can apply this business rule to an XML object.

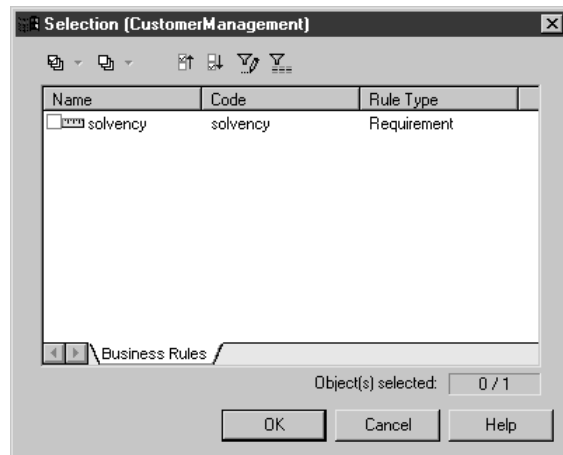
❖ **To apply a business rule to an XML object:**

- 1 Double-click an object in the model to display the object property sheet.
- 2 Click the Rules tab to display the Rules page.



- 3 Click the Add Rules tool to display the list of business rules.

The Selection window appears. It contains a list of all the business rules of the model, with the exception of those that already belong to the object.



- 4 Select the business rules you want to add to the object.
- 5 Click OK.

The business rules are added to the object and appear in the list of business rules for the object.

- 6 Click OK.

U Column in the List of business rules

When you apply a business rule to an object, the U (Used) column beside this business rule is automatically checked in the List of business rules to indicate that the business rule is used by at least one object in the model. The U column allows you to visualize unused business rules, you can then delete them if necessary.

CHAPTER 3

Working with an XML model

About this chapter This chapter describes how you can work with an XML model and how you can generate an XML model from a PDM or an OOM.

When building an XML model, you can check the validity of your model at any time, with the Check Model feature. You can also map XML objects to PDM or OOM objects, merge XML models and edit reports.

Contents

Topic	Page
Checking an XML model	140
XML Model objects verified by Check Model	147
Mapping objects in an XML model	166
Manipulating XML objects graphically	176
Comparing and merging XML models	179
Generating an XML model from a Physical Data Model	180
Generating an XML model from an Object-Oriented Model	188
Editing an XML model report	196

Checking an XML model

The XML Model is a very flexible tool. It should allow you to build your model without any controls or constraints on data exchange and consistency of the system.

However, you can use the Check Model feature at any time to control the consistency and correctness of the model you are building.

You can:

- ◆ Define check options, such as level of problem severity and automatic correction
- ◆ Select objects to be verified
- ◆ Check an XML model
- ◆ Reuse check options previously defined
- ◆ Make corrections based on XML model check results



XML model check options

When you check an XML model, if a parameter is found to be invalid, it will be displayed as an error or a warning in the Check Model window.

You can define levels of severity for problems that Check Model finds and you can have certain problems automatically corrected.

Levels of problem severity

You can identify the level of problem severity with the following tool:


Tool	Indicates	Description
	Error	Major problem that produces an invalid XML model
	Warning	Minor problem or recommendation

To display the severity level options, select Tools→Check Model, then expand the nodes of the tree view in the options tab.

These messages represent two different levels of problem severity. You can modify the level of problem severity for each object parameter verified by the Check model. This severity level can depend on the degree of normalization you want to achieve in your model.


Automatic correction

You can specify if you want PowerDesigner to automatically correct an error using the Automatic Correction feature.

Tool	Indicates	Description
	Automatic correction	PowerDesigner will correct the problem automatically

However, before using automatic correction, make sure you understand how it will affect your model.

Automatic correction is not available for all object parameters. Problems that cannot be corrected automatically must be corrected manually.

 For more information on objects available for automatic correction, see section XML Model objects verified by Check Model.

XML model object selection in the Check Model

You select objects to be checked from the Selection page of the Check Model Parameters dialog box (Tools→Check Model).

You can list all objects, including composite objects created in the current model, by selecting the Include Sub-Objects tool.

Selecting objects in the diagram

If you graphically select objects in your diagram before starting the Check Model, they can be automatically selected for verification by the Check Model by clicking the Use Graphical Selection tool in the Selection page toolbar.

Checking an XML model

You can check the validity of an XML model at any time.

❖ To check an XML model:

- 1 Select Tools→Check Model.

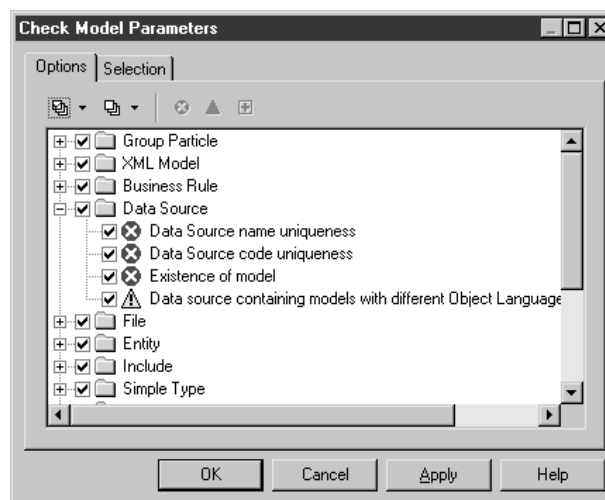
or

Right-click the diagram background and select Check Model from the contextual menu.

The Check Model Parameters dialog box opens to the Options page.

- 2 Expand an object parameter node.

The object parameters verified by the Check Model are displayed with the symbols indicating a degree of problem severity.



- 3 If you want to change a degree of problem severity, select the object parameter and then select either the Error or Warning tool.

The symbol changes to the appropriate severity level.

- 4 If you want PowerDesigner to automatically correct a problem, select the object parameter and then select the Automatic Correction tool.

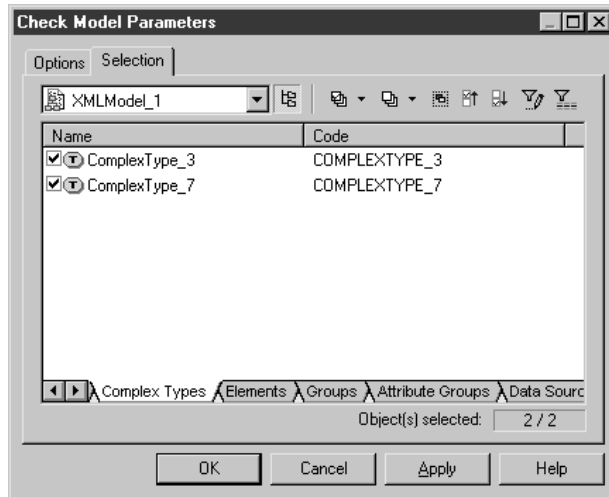
The Automatic Correction symbol appears superimposed on the Error or Warning symbol for that object parameter.

- 5 Click the Selection tab to display the Selection page.

- 6 Select a model from the dropdown listbox at the top of the dialog box.

- 7 Click an object tab at the bottom of the Selection page.

The corresponding object page displays all the objects in the current XML model.



- 8 Select check boxes only for objects you want to be checked.
- 9 Clear check boxes for objects that you do not want to be checked.

Selecting all or clearing all check boxes

You can select all object check boxes by clicking the Select All tool. You can clear all object check boxes by clicking the Deselect All tool.

- 10 Click OK.

The Check Model Result List displays errors and warnings based on the check options you have defined.

Category	Check	Object	Location
Entity	Undefined entity	Entity 'Entity_1'	<Model>
Complex Type	Existence of attr...	Complex Type "	<Model::Eleme...
Complex Type	Existence of attr...	Complex Type 'ComplexTyp...	<Model>
Element	Existence of attr...	Element 'Element_1'	<Model::Eleme...
Element	Existence of attr...	Element 'Element_1'	<Model::Compl...
Element	Existence of attr...	Element 'Element_2'	
Element	Existence of attr...	Element 'Element_2'	<Model::Compl...
Element	Existence of attr...	Element 'Element_2'	<Model::Eleme...
Element	Existence of attr...	Element 'Element_3'	<Model::Eleme...
Notation	Undefined notat...	Notation 'Notation_1'	<Model>
Attribute Group	Existence of attr...	Attribute Group 'AttributeGr...	<Model>
Redefine	Existence of co...	Redefine 'redefine 1'	<Model>

Dockable result window

When you right-click an object parameter in the Result List, a contextual menu appears listing correction options. Among these, you can also select options to clear, dock or hide the result window.

Displaying previously applied check options in an XML model

If you click the Apply button in the Check Model Parameters dialog box, all error and warning selections are stored in memory.

❖ **To display errors and warnings selected in the last check:**





- ◆ Select Tools→Check Model.

The Check Model Parameters dialog box opens to the Options page. The object parameters are displayed with the Error or Warning icons selected in the last check.

Making corrections based on XML model check results

You use the Check Model feature to locate and correct problems in an XML model.

Using the Check toolbar generally located in the upper part of the PowerDesigner main window, you have access to the following correction options when you select an error in the Result List:

Symbol	Option	Description
	Correct error	Displays property sheet of the problem object
	Display details	Displays description of the error and suggestion for correction
	Re-check model	Checks selected object parameter, normally after a correction has been done
	Automatic correction	PowerDesigner automatically corrects non-unique names and codes

Display the Check toolbar

If the Check toolbar is not displayed, select Tools→Customize Toolbars and select the Check check box.

Navigating in the error list

The Check toolbar contains navigation tools to move to the first, previous, next, or last error listed. You can also navigate in the Result list by right-clicking an object parameter and selecting Go To First error, Previous error, Next error, or Last error from the contextual menu.

Contextual menu

When you right click an object parameter, a menu appears listing the correction options: Manual Correction, Help, Check again, and Automatic Correction. You can also select options to clear, dock and hide the result window.

Making manual corrections to an XML model

Some errors cannot be corrected automatically, they have to be corrected manually.

❖ To make manual corrections to an XML model:

- 1 Select an object parameter from the Result List.
- 2 Right-click the object parameter and select Correct from the contextual menu to display the object property sheet.
- 3 Select the appropriate tab and make the necessary correction.
- 4 Close the property sheet.
- 5 Right-click the object parameter and select Check again from the contextual menu.

Verify that the problem has been corrected by running Check Model again.

Making automatic corrections to an XML model

PowerDesigner can perform automatic corrections on non-unique names and codes.

❖ To make automatic corrections to an XML model:

- 1 Select an object parameter from the Result List.
- 2 Right-click the object parameter and select Automatic Correction from the contextual menu.
- 3 Right-click the object parameter and select Re-check from the contextual menu.

Verify that the problem has been corrected by running Check Model again.

XML Model objects verified by Check Model

The Check Model verifies the validity of XML model objects.

When errors are encountered during a check model, corrections can be made manually or automatically. Manual corrections depend on how you are using your model.

Some checks are not available if the object is not supported in the model.

Use the Help command to select object control options

When you right click an XML model object control in the Check Model Parameters page, a menu appears listing several options. The Help command opens a contextual help page explaining the checks performed for the selected object type.

Group particle check

During a group particle check, the following object control is made.

Existence of particle

A group particle must contain elements, groups, group particles and/or Any.

Manual correction	Automatic correction
Add items to the group particle or delete it	—

Invalid cardinality

You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence.

This check is only available in a model targeted with XDR.

Manual correction	Automatic correction
Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties	—

Model check

This check only applies to models built on a schema.

During a model check, the following object controls are made.

Identifier uniqueness

Two or more objects cannot have the same identifier (ID).

Manual correction	Automatic correction
Give a unique identifier to each object	—

Undefined identifier

You must define an identifier (ID) for each object in the model.

Manual correction	Automatic correction
Define an identifier for each object	—

Shortcut code uniqueness

Two shortcuts with the same code cannot be in the same namespace.

Manual correction	Automatic correction
Change the code of one of the shortcuts	—

Undefined target namespace

You should define a target namespace to your model.

Manual correction	Automatic correction
Type a URI for the Target Namespace property in the Detail page of the model property sheet	—

Missing namespaces

There should be at least one namespace defined for the model.

Manual correction	Automatic correction
Type a URI and a prefix in the Namespaces page of the model property sheet	Adds the target namespace URI and a prefix “ns” followed by a number (e.g. “ns1”)

Business rule check in an XSM

During a business rule check, the following object controls are made.

Business rule name and code uniqueness

A model cannot contain two business rules with identical names and/or codes.

Manual correction	Automatic correction
Modify the duplicated business rule name/code	Modifies the business rule name/code of a selected object by appending a number to its current name

Unused business rule

The business rule you have created should be used in the model.

Manual correction	Automatic correction
Apply the business rule to an object in the model	—

Data source check

During a data source check, the following object controls are made.

Data source name and code uniqueness

There cannot be two data sources with identical names and/or codes in a model.

Manual correction	Automatic correction
Rename one of the data sources	Renames the data source by appending a number to its name or code

Existence of model

A data source must have at least one model in its definition.

Manual correction	Automatic correction
Add a model from the Models page of the data source property sheet	Deletes data source without a model

Data source containing models with different Object Language or DBMS types

The models in a data source represent a single set of information. This is why the models in the data source should share the same DBMS or object language.

Manual correction	Automatic correction
Delete models with different DBMS or object language, or modify the DBMS or object language of models in the data source	—

File check

During a file check, the following object controls are made.

Embedded file name uniqueness

A model cannot contain two embedded files with identical names.

Manual correction	Automatic correction
Rename one of the embedded files	Renames the file by appending a number to its name

Existence of external file location

An external file should have a valid path location.

Manual correction	Automatic correction
Define a valid path location	—

Entity check

During an entity check, the following object controls are made.

Entity name and code uniqueness

A model cannot contain two entities with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the entities	Renames the entity by appending a number to its name or code

Undefined entity

You must define an entity. In the entity property sheet, you must either type a value (string of characters or URI) in the Value box, or a URI in the Public or System boxes.

Manual correction	Automatic correction
Type a value in the Value box or a URI in the Public or System boxes	—

Include check

During an include check, the following object control is made.

Undefined schema location

You must define a schema location for an include.

Manual correction	Automatic correction
Define a URI or select a schema file for the schema location. For example: proforma.xsd	—

Simple type check

During a simple type check, the following object control is made.

Simple type name and code uniqueness

A model cannot contain two simple types with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the simple types	Renames the simple type by appending a number to its name or code

Complex type check

During a complex type check, the following object controls are made.

Complex type name and code uniqueness

A model cannot contain two complex types with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the complex types	Renames the complex type by appending a number to its name or code

Existence of attribute

A complex type should have at least one attribute.

Manual correction	Automatic correction
Define an attribute for the complex type	—

Existence of particle

A complex type must contain elements, groups, group particles and/or Any.

Manual correction	Automatic correction
Add items to the complex type or delete complex type	—

Element check

During an element check, the following object controls are made.

Element name and code uniqueness

A model cannot contain two elements with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the elements	Renames the element by appending a number to its name or code

Undefined type

An element without a reference should have a defined data type.

Manual correction	Automatic correction
In the element property sheet, define a data type with the Type dropdown listbox or the Browse tool	—

Undefined reference

An element without a defined data type must have a reference.

Manual correction	Automatic correction
In the element property sheet, define a reference with the Reference dropdown listbox or the Browse tool	—

Existence of attribute

An element without a reference, a data type or a substitution group should have at least one attribute.

Manual correction	Automatic correction
Define an attribute for the element	—

Existence of particle

An element with an embedded complex type must contain child elements, groups, group particles and/or Any.

Manual correction	Automatic correction
Add items to complex element or delete complex element	—

Invalid cardinality

You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence.

This check is only available in a model targeted with XDR.

Manual correction	Automatic correction
Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties	—

Group check

During a group check, the following object controls are made.

Group name and code uniqueness

A model cannot contain two groups with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the groups	Renames the group by appending a number to its name or code

Undefined reference

A group without a name or a code must have a reference.

Manual correction	Automatic correction
In the group property sheet, define a reference with the Reference dropdown listbox or the Browse tool	—

Existence of particle

A group must contain elements, groups, group particles and/or Any.

Manual correction	Automatic correction
Add items to group or delete group	—

Invalid cardinality

You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence.

This check is only available in a model targeted with XDR.

Manual correction	Automatic correction
Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties	—

Attribute check

During an attribute check, the following object controls are made.

Attribute name and code uniqueness

The parent of an attribute cannot contain two attributes with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the attributes	Renames the attribute by appending a number to its name or code

Undefined reference

An attribute without a name or a code must have a reference.

Manual correction	Automatic correction
In the attribute property sheet, define a reference with the Reference dropdown listbox or the Browse tool	—

Undefined type

You must define a data type for an attribute.

Manual correction	Automatic correction
In the attribute property sheet, define a data type with the Type dropdown listbox or the Browse tool	—

Notation check

During a notation check, the following object controls are made.

Notation name and code uniqueness

A model cannot contain two notations with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the notations	Renames the notation by appending a number to its name or code

Undefined notation

A notation must have at least one URI defined for Public or System properties.

Manual correction	Automatic correction
In the notation property sheet, define a URI in the Public or System boxes	—

Attribute group check

During an attribute group check, the following object controls are made.

Attribute group name and code uniqueness

A model cannot contain two attribute groups with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the attribute groups	Renames the attribute group by appending a number to its name or code

Undefined reference

An attribute group without a name or a code must have a reference.

Manual correction	Automatic correction
In the attribute group property sheet, define a reference with the Reference dropdown listbox or the Browse tool	—

Existence of attributes

An attribute group must contain at least one attribute.

Manual correction	Automatic correction
Add attributes to attribute group or delete attribute group	Deletes unassigned attribute group

Import check

During an import check, the following object control is made.

Undefined schema location and namespace

An import must have at least a schema location or a namespace defined.

Manual correction	Automatic correction
In the import property sheet, define a URI for the schema location and/or the namespace. Example for a schema location: http://www.something.org/schemas/order.xsd Example for a namespace: http://www.something.org/xml/ordering	—

Redefine check

During a redefine check, the following object controls are made.

Undefined schema location

You must define a schema location for a redefine.

Manual correction	Automatic correction
In the redefine property sheet, define a URI or select a schema file for the schema location. For example: customers.xsd	—

Existence of component

A redefine must contain at least one of the following items: simple type, complex type, group or attribute group.

Manual correction	Automatic correction
Add items to the redefine	—

Key check

During a key check, the following object controls are made.

Key name and code uniqueness


An element cannot contain two keys with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the keys	Renames the key by appending a number to its name or code

Existence of fields

A key must contain at least one field.

Manual correction	Automatic correction
Add at least one field to the key or delete the key. For example: @numEmployee	Deletes unassigned key

 For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.

Undefined selector

You must define an XPath expression for a key selector attribute.

Manual correction	Automatic correction
In the key property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee	—

🔗 For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

KeyRef check

During a keyRef check, the following object controls are made.

KeyRef name and code uniqueness

An element cannot contain two keyRefs with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the keyRefs	Renames the keyRef by appending a number to its name or code

Undefined reference

A keyRef must contain a reference to a key or a unique constraint.

Manual correction	Automatic correction
In the keyRef property sheet, define a reference to a key or a unique constraint with the Reference dropdown listbox	—

Existence of fields

A keyRef must contain at least one field.


Manual correction	Automatic correction
Add at least one field to the keyRef or delete the keyRef. For example: @numEmployee	Deletes unassigned keyRef

🔗 For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.

Undefined selector

You must define an XPath expression for a keyRef selector attribute.

Manual correction	Automatic correction
In the keyRef property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee	—

 For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

Unique check

During a unique constraint check, the following object controls are made.

Unique name and code uniqueness


An element cannot contain two unique constraints with identical names and/or codes.

Manual correction	Automatic correction
Rename one of the unique constraints	Renames the unique constraint by appending a number to its name or code

Existence of fields

A unique constraint must contain at least one field.


Manual correction	Automatic correction
Add at least one field to the unique constraint or delete the unique constraint. For example: @numEmployee	Deletes unassigned unique constraint

 For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.

Undefined Selector

You must define an XPath expression for a unique constraint selector attribute.

Manual correction	Automatic correction
In the unique constraint property sheet, define an XPath expression for the unique constraint selector attribute. For example: s:company/s:employee	—

 For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

Extended object check

During an extended object check, the following object control is made.

Extended object name and code uniqueness

Extended object names and codes must be unique in the namespace.

Manual correction	Automatic correction
Modify the duplicate extended object name/code	Modifies the extended object name or code of a selected object by appending a number to its current name or code

Extended link check

During an extended link check, the following object control is made.

Extended link name and code uniqueness

Extended link names and codes must be unique in the namespace.

Manual correction	Automatic correction
Modify the duplicate extended link name/code	Modifies the extended link name or code of a selected object by appending a number to its current name or code

Replication check

During a replication check, the following object control is made.

Partial object replication

A replica object is partially synchronized with its replicated object.

Manual correction	Automatic correction
Modify the list of replicated attributes from the replication property sheet	Enforces the replication of desynchronized attributes of the replica object in the replication property sheet

Extension check

During an extension check, the following object control is made.

Undefined base type

You must define a base type when you derive a complex type by extension.

Manual correction	Automatic correction
In the complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type dropdown listbox or the Browse tool	—

Restriction check

During a restriction check, the following object controls are made.

Undefined base type

You must define a base type when you derive a simple or a complex type by restriction.

Manual correction	Automatic correction
In the simple or complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type dropdown listbox or the Browse tool	—

Existence of facet

A simple type restriction must have at least one facet defined. Facets are defined in the Detail, Enumerations and Patterns pages of a simple type restriction property sheet.

Manual correction	Automatic correction
Define one or more facets in the simple type restriction property sheet	—

Simple type list check

During a simple type list check, the following object control is made.

Undefined base type

You must define a base type when you derive a simple type by list.

Manual correction	Automatic correction
In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type list property sheet and select a data type with the Type dropdown listbox or the Browse tool	—

Simple type union check

During a simple type union check, the following object control is made.

Undefined base type

You must define at least two data types when you derive a simple type by union.

Manual correction	Automatic correction
In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type union property sheet and type a white space separated list of at least two data types (qualified names) in the Member types box	—

Annotation check

During an annotation check, the following object control is made.

Existence of items

An annotation must contain at least one URI for a Documentation or an Application Information.

Manual correction	Automatic correction
Define a URI for a Documentation or an Application Information	—

Mapping objects in an XML model

Object mapping is the ability to establish a correspondence between objects belonging to heterogeneous models and diagrams.

The following table shows all the allowed mappings between XML Model objects and PDM or OOM objects:

XML Model objects	PDM objects	OOM objects
Element	Table, view, column, view column	Class, class attribute
Element attribute	Column, view column	Class attribute
Complex type	Abstract data type	Class
Complex type attribute	Abstract data type attribute	Class attribute

Understanding object mapping

You create a mapping between XML Model objects and PDM or OOM objects to setup a structure for data movement and transformation. Data comes from a data source and is loaded in an XML model.

In an XML model, the data source can be a PDM representing a database or an OOM representing classes.

Object mapping consists in linking objects in a PDM or an OOM data source to objects in an XML model.

Defining data sources in an XML model

You create a data source to define where (in which PDM or OOM) data should be extracted to be transferred to an XML model.

When you define a data source, you have to declare PDMs or OOMs in the list of data source models. A data source can contain several models, you can select the source models among a list of models opened in the current workspace.

A model can contain several data sources.

Data source properties in an XML model

To display a data source property sheet, double-click its name or its icon in the Browser tree view.

Data source general properties The General page of a data source property sheet displays the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the data source
Model Type	Type of model(s) being used as data source

Data source models properties The Models page of a data source property sheet displays an empty list of models. Click the Add Models tool to display a selection dialog box and select models among those opened in the current workspace.

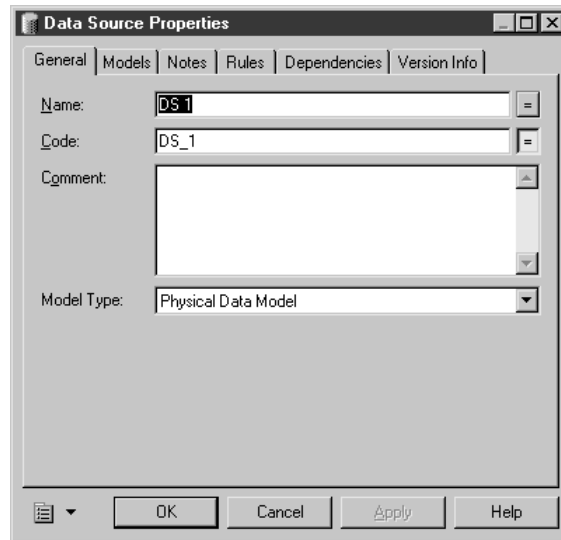
Creating a data source in an XML model

You can create as many data sources as you need in a model. Each data source defines the list of models used as source of data in the current model.

❖ **To create a data source in an XML model:**

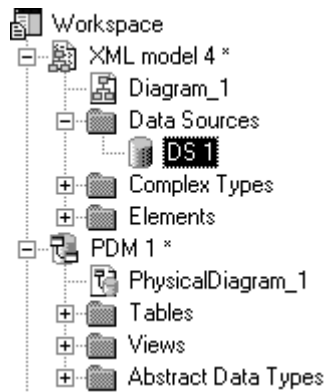
- 1 Select Model→Data Sources to display the List of Data Sources.
- 2 Click a blank line in the list.
or
Click the Add a Row tool.
An arrow appears at the beginning of the line.
- 3 Type a name and a code for the data source.
- 4 Click Apply.

- 5 Click the Properties tool to display the data source property sheet.



- 6 In the General page, select a model type in the Model Type dropdown listbox.
- 7 In the Models page, click the Add Models tool.
A model selection dialog box appears. Select the model(s) for the current data source.
- 8 Click OK in each of the dialog boxes.

The data source appears in the Browser tree view.



Mapping XML Model objects to PDM objects

You can map elements or complex types and their corresponding attributes to PDM objects.

The following table shows all the allowed mappings between XML Model objects and PDM objects:

XML Model object	PDM object
Element	Table, view, column, view column
Element attribute	Column, view column
Complex type	Abstract data type
Complex type attribute	Abstract data type attribute

In this section, we will talk about tables but the functionalities also apply to views, columns and/or view columns.

Mapping elements

You can map elements to PDM objects.

Once the tables are identified and mapped with the elements, you can define mappings between columns and attributes.

But columns can also be mapped to elements. Mapping a column either to an attribute or an element depends on the schema presentation:

Case 1:

```
<customer name="c"/>
```

A column should be mapped to the name attribute.




Case 2:

```
<customer>
  <name>c</name>
</customer>
```

A column should be mapped to the name element.

You have to use the Mapping page of an element property sheet to define element mappings.

In the Mapping page, you can use the following tools to select a data source for the current element:

Tool	Description
	Adds a mapping between the current element and an existing data source. The first time you define a mapping for an element, the Mapping for dropdown listbox is empty. You have to click the Add a Mapping for a Data Source tool before you can map the elements to tables in the data source
	Deletes the mapping for the data source
	Edits properties of the selected data source




Element Sources page

The Element Sources page allows you to associate one or several tables in the data source to the current element.

You can use the Add Objects tool to select tables from the PDMs opened in the current workspace.

Attributes Mapping page

The Attributes Mapping page allows you to define the mapping between columns in the source tables and attributes in the current element.


Tool	Tooltip	Description
	Add Mapping	To select the attributes in the current element that will be mapped to columns in the source table. Once you have selected the attributes, you can use the dropdown listbox in the Mapped to column to select corresponding columns in the source table
	Create from Sources	To copy columns from the tables in the data source to the current element
	Generate Mapping	To automatically generate a mapping between columns and attributes with same name or code in the data source and the current model

Mapping complex types

You can map complex types to PDM objects.

Once the abstract data types are identified and mapped with the complex types, you can define mappings between abstract data type attributes and complex type attributes.

You have to use the Mapping page of a complex type property sheet to define complex type mappings.

 For more information on how to select a data source, see section Mapping elements.


Complex Type Sources page

The Complex Type Sources page allows you to associate one or several abstract data types in the data source to the current complex type.

You can use the Add Objects tool to select abstract data types from the PDMs opened in the current workspace.

Attributes Mapping page

The Attributes Mapping page allows you to define the mapping between attributes in the source abstract data types and attributes in the current complex type.

 For more information on the Attributes Mapping page, see section Mapping elements.

Mapping XML Model objects to OOM objects

You can map elements or complex types and their corresponding attributes to OOM objects.

The following table lists object mapping in an XML Model to OOM mapping:

XML Model object	OOM object
Element	Class, class attribute
Element attribute	Class attribute
Complex type	Class
Complex type attribute	Class attribute



In this section, we will talk about classes but the functionalities also apply to class attributes.

Mapping elements



You can map elements to OOM objects.

Once the classes are identified and mapped with the elements, you can define mappings between class attributes and element attributes.

You have to use the Mapping page of an element property sheet to define element mappings.

	<p> For more information on how to select a data source, see section Mapping elements.</p>
Element Sources page	<p>The Element Sources page allows you to associate one or several classes in the data source to the current element.</p> <p>You can use the Add Objects tool to select classes from the OOMs opened in the current workspace.</p>
Attributes Mapping page	<p>The Attributes Mapping page allows you to define the mapping between class attributes in the source classes and attributes in the current element.</p> <p> For more information on the Attributes Mapping page, see section Mapping elements.</p>

Mapping complex types

	<p>You can map complex types to OOM objects.</p> <p>Once the classes are identified and mapped with the complex types, you can define mappings between class attributes and complex type attributes.</p> <p>You have to use the Mapping page of a complex type property sheet to define complex type mappings.</p> <p> For more information on how to select a data source, see section Mapping elements.</p>
Complex Type Sources page	<p>The Complex Type Sources page allows you to associate one or several classes in the data source to the current complex type.</p> <p>You can use the Add Objects tool to select classes from the OOMs opened in the current workspace.</p>
Attributes Mapping page	<p>The Attributes Mapping page allows you to define the mapping between attributes in the source classes and attributes in the current complex type.</p> <p> For more information on the Attributes Mapping page, see section Mapping elements.</p>

Creating a mapping for an XML object

Here we choose to map an element to PDM tables, but the same procedure applies for other XML objects.

❖ **To create a mapping for an element:**

- 1 Double-click an element to display its property sheet.

- 2 Select the Mapping tab to display the Mapping page.
- 3 Click the Add a Mapping for a Data Source tool to select a data source for the mapping.

or

Select a data source in the Mapping For dropdown listbox if you have already selected one or several data sources.

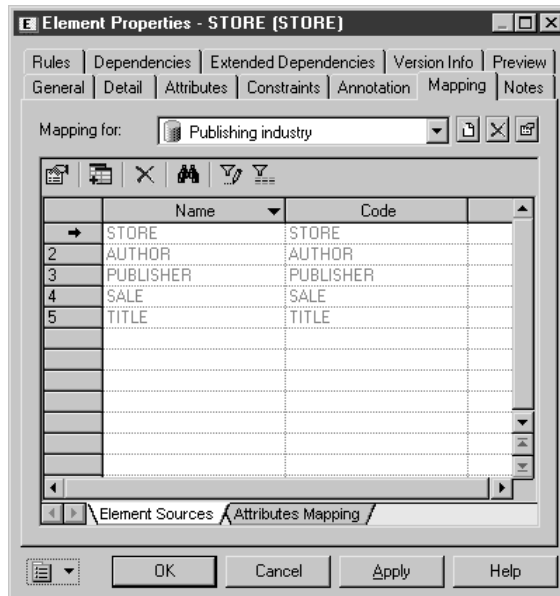
Caution

You cannot create a mapping for an XML object if you have not previously defined a data source containing at least one source model.

*To define a data source, see section *Defining data sources in an XML model*.*

- 4 Click OK.
- 5 In the Element Sources page, click the Add Objects tool to display the Add Objects dialog box.
- 6 In the Table page, select the tables you want to map to the current element.
- 7 Click OK.

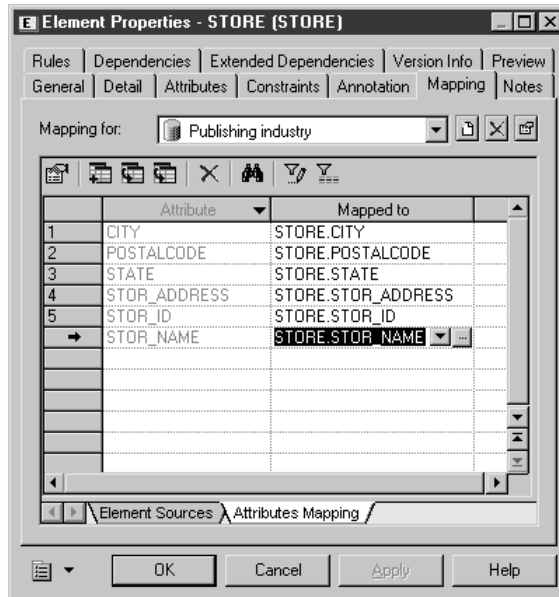
The selected tables appear in the Element Sources list.



- 8 Select the Attributes Mapping tab in the Mapping page.
- 9 Click the Add Mapping tool to display an attribute selection box.

Create From Sources and Generate Mapping tools
Click the Create From Sources tool to automatically create attributes from the columns selected in the data source.
Click the Generate Mapping tool to automatically create the mapping between columns and attributes with identical name or code in the data source and the current model

- 10 Select the attributes that will be mapped to columns in the selected data source.
- 11 Click OK.
The selected attributes appear in the Attribute column of the list.
- 12 Click the Mapped to column in front of each attribute to select, with the down arrow, the column in the data source table to which the selected attribute will be mapped.



- 13 Click OK.

Modifying the mapping of an attribute

You can modify the default mapping created for an attribute. The Attribute Mappings property sheet can be used to fine tune the mapping between an attribute in the current element and columns or attributes in data source tables or classes.

In the **Mapped To** box, you can see the column or attribute expression. By default, columns or attributes used in this expression are prefixed by their tables or classes. You can customize the content of the Mapped To box by inserting comments manually. You can also click the Ellipsis button and use the SQL Editor to modify the default mapping syntax. To recover the default column expression click the User-defined tool.

In the **Sources** page it is possible to select several columns or attributes and map them to the current attribute. To do so, you have to use the Add Sources tool to select columns or attributes from the list of columns or attributes belonging to the tables or classes mapped to the current element. When you add columns or attributes from the Sources page, and when the column or attribute expression has not been modified by the user, the content of the Mapped To box in the General page is updated.

❖ To modify the mapping of an attribute:

- 1 In the Mapping page, select the Attributes Mapping tab to display the corresponding page.
- 2 Select an attribute in the list and click the Properties tool.
The Attribute Mappings Properties appears.
- 3 Click the Sources tab to display the corresponding page.
- 4 Click the Add Sources tool to display a Selection dialog box.
- 5 Select source columns or attributes and click OK.
The columns or attributes appear in the Sources list.
- 6 Click the General tab to display the corresponding page. The selected columns or attributes appear in the Mapped To box.
- 7 Click OK.

Manipulating XML objects graphically

The graphical interface of PowerDesigner allows you to manipulate XML objects within or between the Browser tree view and the diagram window.

A global object is right under the model item in the Browser tree view. It has no parent symbol in the diagram.

A local object is under a group particle item in the Browser tree view. It has a parent symbol in the diagram.

The following sections explain all the graphical operations allowed in PowerDesigner.

Local objects

You can move local objects within or between the Browser tree view and the diagram window.

Move

Select Tools→General Options to make sure that Move is the Default action of the Drag & Drop option.

Within Browser

You can move a local object within the Browser to convert it into a global object. If the new global object does not appear in the diagram, select Symbol→Show Symbols and click the corresponding tab to select the new global object

Within diagram

You can move a local object within the diagram to another group particle. It remains a local object, but with a new parent object.

You cannot move a local object within the diagram to convert it into a global object. It remains attached to its group particle.

From Browser to diagram

When you move a local object from the Browser to the diagram, a synonym is created, attached to the same group particle as the original symbol.

From diagram to Browser

You can move a local object from the diagram to the Browser, and convert it into a global object. If the new global object does not appear in the diagram, select Symbol→Show Symbols and click the corresponding tab to select the new global object

Global objects

You can move global objects within or between the Browser tree view and the diagram window.

Move

Select Tools→General Options to make sure that Move is the Default action of the Drag & Drop option.

Within Browser	You can move a global object within the Browser to convert it into a local object (under a group particle item), but you cannot move a global object within its own structure (as a child of itself).
Within diagram	You can move a global object within the diagram to convert it into a local object. Just move the global object symbol to a group particle symbol.
From Browser to diagram	When you move a global object from the Browser to the diagram, a synonym symbol is created in the diagram.
From diagram to Browser	You can move a global object from the diagram to the Browser, and convert it into a local object (under a group particle item). If the new local object does not appear in the diagram, double-click the Collapse node of the group particle symbol under which the former global object has been attached.

Example: converting a local object into a global object

The following procedure explains how to convert a local object into a global object.

❖ To convert a local object into a global object:

- 1 Click a local object (child element or attribute) in the Browser or the diagram.
- 2 Drag and drop the local object in the space just beneath the Diagram item or the model item. A thick horizontal line indicates that you can drop the object.

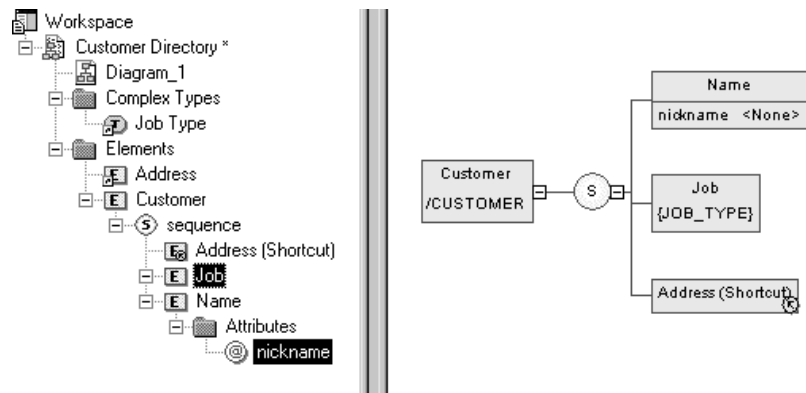
Drag and drop

Select Tools→General Options to make sure that Move is the Default action of the Drag & Drop option.

The new global object appears in a global objects category (one level beneath the model item).

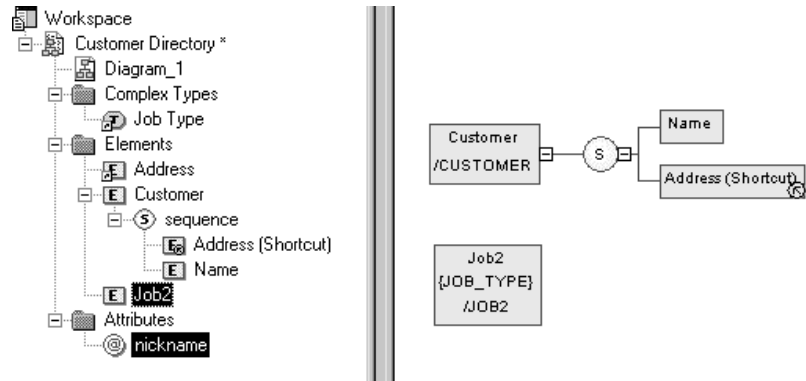
If the new global object does not appear in the diagram, select Symbol→Show Symbols and click the corresponding tab to select the object symbol.

Example before local to global conversions within the Browser



Job is a child element of the Customer element.
Nickname is the attribute of the Name element.

After conversions



Job became Job2, a global element.
Nickname became a global attribute.


Comparing and merging XML models

You can compare and merge two XML models with the same XML language.

The comparison process allows you to highlight the differences between two XML models.

The merge process allows you to form a single model that combines design efforts performed independently by several team members.

Merge is performed from left to right, the model in the right pane is compared to the model in the left pane, differences are highlighted and merge actions are proposed in the model to be merged.

 For more information on comparing and merging models, see chapter Comparing and Merging Models in the *General Features Guide*.

Generating an XML model from a Physical Data Model

This section explains how to generate an XML model from a Physical Data Model.

Generating XML Model objects

When you generate an XML model from a PDM, PowerDesigner translates PDM objects into specified XML Model objects as follows:

PDM object	Generated object in an XML model
Domain	Simple type
Table	Element. A table is generated as a child element when it has one outgoing reference link with the Mandatory parent property selected
Column	Element or element attribute (See generation options)
View	Element
View column	Element attribute
Key	Key
Index	Unique
Abstract data type	Complex type
Abstract data type attribute	Complex type attribute

PDM packages are generated into XML diagrams.

A PDM is generated into several XML diagrams:

- ◆ A global diagram displaying a hierarchical view of all the generated elements
- ◆ One diagram for each physical diagram in order to keep the PDM design. The generated diagram displays the same object symbols as the source diagram

The Global Diagram displays the symbol of a root element called Database. This Database element (which has no equivalent in the source PDM) groups together all the identity constraints (see Constraints page of the Database property sheet) generated from the PDM keys, indexes and references.

Select Symbol→Show Symbols to reveal the other generated XML Model objects (simple types and complex types).

XML model naming conventions

If the code of the generated XML Model objects does not correspond to the target language naming conventions, you can define a code naming convention script to convert object names into codes.

For more information on conversion scripts, see section `.convert_name & .convert_code` macros in chapter Managing Models of the *General Features Guide*.

Generating and updating an XML model


The General page of the XML Model Generation Options dialog box displays the following options:

- ◆ Generate new XML model
- ◆ Update existing XML model

Generate new XML model

You must indicate the following parameters when you generate a new XML model:

Parameter	Description
XML language	Target XML language
Share	XML language for the resulting XML model. It uses a shared XML language definition file stored in the XML Languages library
Copy	XML language for the resulting XML model. It uses a copy of the XML language definition file stored in the XML Languages library
Name	File name for the resulting XML model
Code	Reference code for the resulting XML model
Configure Model Options	It lets you define the model options for the new XML model. You can select the naming conventions corresponding to the target language for example

 For more information on model options when generating an XML model from a PDM, see Defining XML model generation options.

Update existing XML model

You can generate a PDM into an existing XML model.

If you choose to update an existing XML model when the current PDM has never been generated, the Select model dropdown listbox is empty by default. You have to click the Ellipsis button to display the Select a model dialog box in which you can select a model. The Workspace Location and Physical Path columns provide more information about the model location in the workspace and the file path to help you choose the XML model. You can also use the tooltip in the dropdown listbox to verify the location and path of the model.

To update an existing XML model, you must indicate the following parameters:

Parameter	Description
Select model	Existing XML model. The PDM is generated into an XML model. This XML model is merged with the existing XML model to create an updated XML model. The dropdown listbox displays already generated models. The Ellipsis button lets you select models opened in the workspace
XML language	Current XML language definition for the XML model
Preserve modifications	Allows a comparison and merge of the newly generated XML model (default XML model) with the currently selected XML model


Clearing the Preserve modifications check box
 If the Preserve modifications check box is not selected, PowerDesigner automatically replaces the existing XML model with the newly generated XML model.
 If you want to choose which objects to add or delete from the model to be merged, you must select Preserve modifications to compare and merge the two XML models.

Defining XML model generation options

The Detail page of the XML Model Generation Options dialog box displays the following options:

Option	Description
Check model	When selected, it verifies the model before generating the XML model, and stops generation if an error is found

Option	Description
Save generation dependencies	When selected, PowerDesigner keeps track of the identity of each generated object. It is useful when merging two XML models which have been generated from the same PDM. Objects can be compared and recognized as the same object, even if the object has been modified in the merged XML model
Convert names into codes	When selected, object codes are generated from names using the corresponding conversion script. This is useful when generating models with very different naming conventions. With this option selected, both objects will have their codes generated from their names. If you do not select this option, generated object codes will be copied from original object codes
Generate mapping	When selected, allows to define the current PDM as the data source of the generated XML model and to create object correspondence between PDM and XML model
Enable transformations	This button is used to activate transformations during generation. When you click this button, the Pre-generation tab appears if the source model contains transformations. You can select the transformations to execute before generation. The Extended Model Definitions tab also appears for you to select extended model definition files to attach to the generated model. These files may contain post-generation transformations, in this case, the Post-Generation tab appears to let you select the transformations you want to be executed in the generated model. If the generation is an update, and the generated model contains extended model definitions with post-generation transformations, the Post-generation tab automatically appears as soon as you click the Enable Transformations button
Generate columns as elements	When selected, columns in the PDM tables are generated as child elements (instead of attributes) in the XML model. You can then set attributes to these child elements

 For more information on conversion scripts, see chapter Managing Models in the *General Features Guide*.

Check model before generation

If you select the Check model option, the procedure to generate an XML model starts by checking the validity of the PDM. An XML model results when no errors are found. You can set check options by selecting Tools→Check Model.

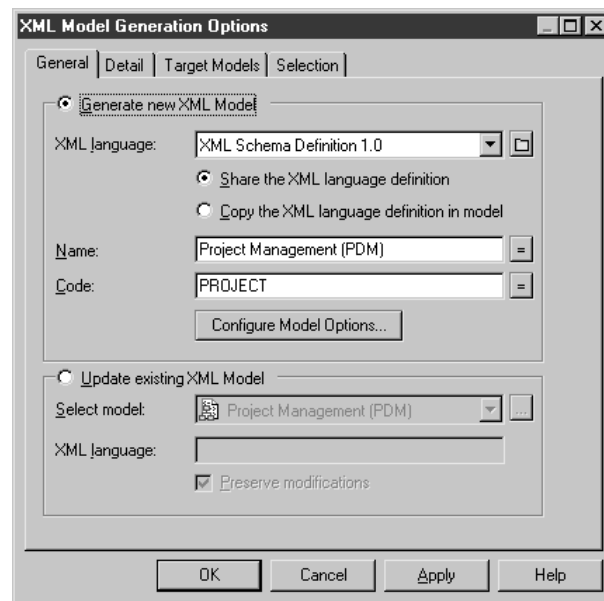
Generating a new XML model from a PDM

You can generate an XML model from a PDM or from a package in a PDM. PowerDesigner creates a new XML model containing all the objects that you selected to generate in the XML model. The newly created XML model appears in the browser and the corresponding diagram opens in the work area.

You can only generate an XML model from the active PDM diagram.

❖ To generate a new XML model from a PDM:

- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.
- 2 Select Generate new XML Model radio button.



- 3 Type a name and a code for the new XML model, otherwise it will have the same name and code as the PDM.
- 4 <optional> Click Configure Model Options to define the options of the generated XML model.

- 5 Click the Detail tab to define options and generation parameters.
- 6 Click the Target Models tab to select the target models of shortcuts in the current model.
- 7 Click the Selection tab to display the Selection page.
- 8 Select the name of a PDM from the Select Location dropdown listbox.

Generating an XML model from a package

To generate an XML model from a package, select the package name from the dropdown listbox in the upper left corner of the dialog box. To generate an XML model from a sub-package, select a sub-package from the dropdown listbox in the upper left corner of the dialog box, or select a package name and click the Include Sub-packages tool next to this dropdown listbox.

- 9 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.
- 10 Click OK.

The Output window shows the progress of the generation process. The new XML model diagram appears in the work area.

Updating an existing XML model

There are two ways to update an existing XML model, depending on whether the Preserve modifications option is selected or not:

Preserve modifications	Result
Selected	You can manually compare and merge the existing XML model (right pane) with the newly generated XML model (left pane)
Not selected	The existing XML model is automatically replaced by the newly generated XML model

If the Preserve modifications check box is selected, the Merge Models window appears after the new XML model has been successfully generated. You can use the Merge window to select objects to be updated, deleted, or added to the model to be merged. The model must be open in the workspace to be merged with a source model.

If the Preserve modifications check box is not selected, the existing model is replaced.

You can only generate an XML model from the active PDM diagram window.

❖ **To update an existing XML model by generating from a PDM:**

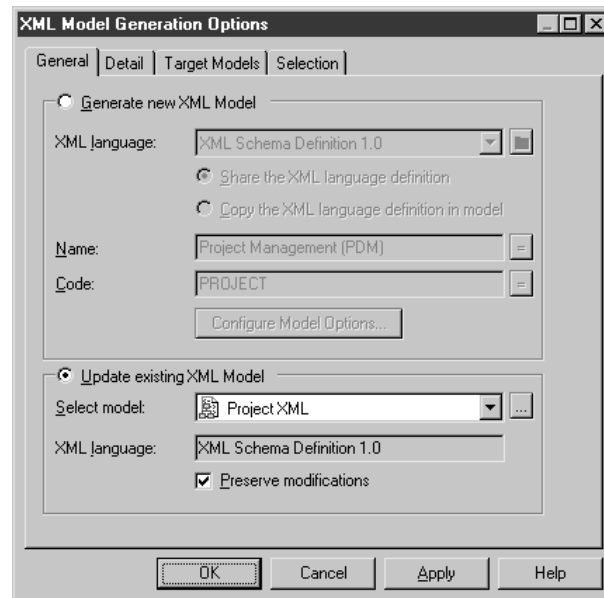
- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.

If you do not have an XML model opened in the current workspace, the Update existing XML Model option is not available.

- 2 Select the Update existing XML Model radio button.
- 3 Select a model from the Select model dropdown listbox, if the current model has already been generated.

or

Click the Ellipsis button, beside the Select model dropdown listbox, and select an existing model in the Select a model dialog box.



Preserve modifications

Select the Preserve modifications check box if you want to preserve objects in the XML model.

If you clear this check box, all existing objects in the XML model will be removed from the model, leaving only the objects generated from the PDM.

- 4 Click the Detail tab to define options and generation parameters.
- 5 Click the Target Models tab to select the target models of shortcuts in the current model.
- 6 Click the Selection tab to display the Selection page.
- 7 Select the name of a PDM from the Select Location dropdown listbox. The default XML model is generated from this PDM.

Generating an XML model from a package

To generate an XML model from a package, select the package name from the dropdown listbox in the upper left corner of the dialog box. To generate an XML model from a sub-package, select a sub-package from the dropdown listbox in the upper left corner of the dialog box, or select a package name and click the Include Sub-Objects tool next to this dropdown listbox.


- 8 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.
- 9 Click OK.

If you selected the Preserve modifications check box, the Merge Models window appears.

If you cleared the Preserve modifications check box, the updated XML model diagram appears in the work area.

Merging models

The Merge Models dialog box shows the newly generated XML model in the left pane, and the existing XML model in the right pane. You can select or clear object check boxes in the right pane for XML model objects that you want to include or delete in the model to be merged.

 For more information on merging models, see Comparing and Merging Models in the *General Features Guide*.

Generating an XML model from an Object-Oriented Model

This section explains how to generate an XML model from an Object-Oriented Model.

Generating XML Model objects

When you generate an XML model from an OOM, PowerDesigner translates OOM objects into specified XML Model objects as follows:

OOM Object	Generated object in an XML model
Class	Element. A class is generated as a child element when it has one composition link with another class
Class attribute	Element or element attribute (See generation options)
Identifier	Key
Domain	Simple type
Abstract class	Complex type
Abstract class attribute	Complex type attribute

OOM packages are generated into XML diagrams.

An OOM is generated into several XML diagrams:

- ◆ A global diagram displaying a hierarchical view of all the generated elements
- ◆ One diagram for each class diagram in order to keep the OOM design. The generated diagram displays the same object symbols as the source diagram

The Global Diagram displays the symbol of a root element called Application. This Application element (which has no equivalent in the source OOM) groups together all the key constraints (see Constraints page of the Application property sheet) generated from the OOM identifiers.

Select Symbol→Show Symbols to reveal the other generated XML Model objects (simple types and complex types).

XML model naming conventions

If the code of the generated XML model objects does not correspond to the target language naming conventions, you can define a code naming convention script to convert object names into codes.

For more information on conversion scripts, see section `.convert_name` & `.convert_code` macros in chapter Managing Models of the *General Features Guide*.

Generating and updating an XML model


The General page of the XML Model Generation Options dialog box displays the following options:

- ◆ Generate new XML model
- ◆ Update existing XML model

Generate new XML model

You must indicate the following parameters when you generate a new XML model:

Parameter	Description
XML language	Target XML language
Share	XML language for the resulting XML model. It uses a shared XML language definition file stored in the XML Languages library
Copy	XML language for the resulting XML model. It uses a copy of the XML language definition file stored in the XML Languages library
Name	File name for the resulting XML model
Code	Reference code for the resulting XML model
Configure Model Options	It lets you define the model options for the new XML model. You can select the naming conventions corresponding to the target language for example

 For more information on model options when generating an XML model from an OOM, see Defining XML model generation options.

Update existing XML model

You can generate an OOM into an existing XML model.

If you choose to update an existing XML model when the current OOM has never been generated, the Select model dropdown listbox is empty by default. You have to click the Ellipsis button to display the Select a model dialog box in which you can select a model. The Workspace Location and Physical Path columns provide more information about the model location in the workspace and the file path to help you choose the XML model. You can also use the tooltip in the dropdown listbox to verify the location and path of the model.

To update an existing XML model, you must indicate the following parameters:

Parameter	Description
Select model	Existing XML model. The OOM is generated into an XML model. This XML model is merged with the existing XML model to create an updated XML model. The dropdown listbox displays already generated models. The Ellipsis button lets you select models opened in the workspace
XML language	Current XML language definition for the XML model
Preserve modifications	Allows a comparison and merge of the newly generated XML model (default XML model) with the currently selected XML model

Clearing the Preserve modifications check box

If the Preserve modifications check box is not selected, PowerDesigner automatically replaces the existing XML model with the newly generated XML model.


If you want to choose which objects to add or delete from the model to be merged, you must select Preserve modifications to compare and merge the two XML models.

Defining XML model generation options

The Detail page of the XML Model Generation Options dialog box displays the following options:

Option	Description
Check model	When selected, it verifies the model before generating the XML model, and stops generation if an error is found

Option	Description
Save generation dependencies	When selected, PowerDesigner keeps track of the identity of each generated object. It is useful when merging two XML models which have been generated from the same OOM. Objects can be compared and recognized as the same object, even if the object has been modified in the merged XML model
Convert names into codes	When selected, object codes are generated from names using the corresponding conversion script. This is useful when generating models with very different naming conventions. With this option selected, both objects will have their codes generated from their names. If you do not select this option, generated object codes will be copied from original object codes
Generate mapping	When selected, allows to define the current OOM as the data source of the generated XML model and to create object correspondence between OOM and XML model
Enable transformations	This button is used to activate transformations during generation. When you click this button, the Pre-generation tab appears if the source model contains transformations. You can select the transformations to execute before generation. The Extended Model Definitions tab also appears for you to select extended model definition files to attach to the generated model. These files may contain post-generation transformations, in this case, the Post-Generation tab appears to let you select the transformations you want to be executed in the generated model. If the generation is an update, and the generated model contains extended model definitions with post-generation transformations, the Post-generation tab automatically appears as soon as you click the Enable Transformations button
Generate attributes as elements	When selected, attributes in the OOM classes are generated as child elements (instead of attributes) in the XML model. You can then set attributes to these child elements

 For more information on conversion scripts, see chapter Managing Models in the *General Features Guide*.

Check model before generation

If you select the Check model option, the procedure to generate an XML model starts by checking the validity of the OOM. An XML model results when no errors are found. You can set check options by selecting Tools→Check Model.

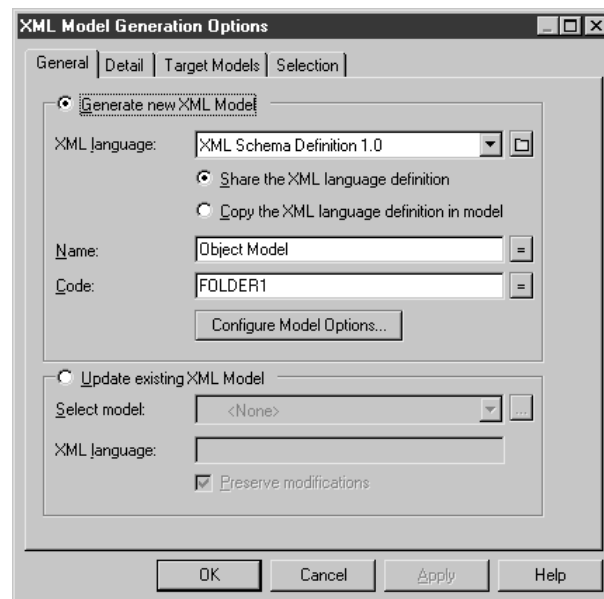
Generating a new XML model from an OOM

You can generate an XML model from an OOM or from a package in an OOM. PowerDesigner creates a new XML model containing all the objects that you selected to generate in the XML model. The newly created XML model appears in the browser and the corresponding diagram opens in the work area.

You can only generate an XML model from the active OOM diagram.

❖ To generate a new XML model from an OOM:

- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.
- 2 Select Generate new XML Model radio button.



- 3 Type a name and a code for the new XML model, otherwise it will have the same name and code as the OOM.
- 4 <optional> Click Configure Model Options to define the options of the generated XML model.

- 5 Click the Detail tab to define options and generation parameters.
- 6 Click the Target Models tab to select the target models of shortcuts in the current model.
- 7 Click the Selection tab to display the Selection page.
- 8 Select the name of an OOM from the Select Location dropdown listbox.

Generating an XML model from a package

To generate an XML model from a package, select the package name from the dropdown listbox in the upper left corner of the dialog box. To generate an XML model from a sub-package, select a sub-package from the dropdown listbox in the upper left corner of the dialog box, or select a package name and click the Include Sub-packages tool next to this dropdown listbox.

- 9 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.
- 10 Click OK.

The Output window shows the progress of the generation process. The new XML model diagram appears in the work area.

Updating an existing XML model

There are two ways to update an existing XML model, depending on whether the Preserve modifications option is selected or not:

Preserve modifications	Result
Selected	You can manually compare and merge the existing XML model (right pane) with the newly generated XML model (left pane)
Not selected	The existing XML model is automatically replaced by the newly generated XML model

If the Preserve modifications check box is selected, the Merge Models window appears after the new XML model has been successfully generated. You can use the Merge window to select objects to be updated, deleted, or added to the model to be merged. The model must be open in the workspace to be merged with a source model.

If the Preserve modifications check box is not selected, the existing model is replaced.

You can only generate an XML model from the active OOM diagram window.

❖ **To update an existing XML model by generating from an OOM:**

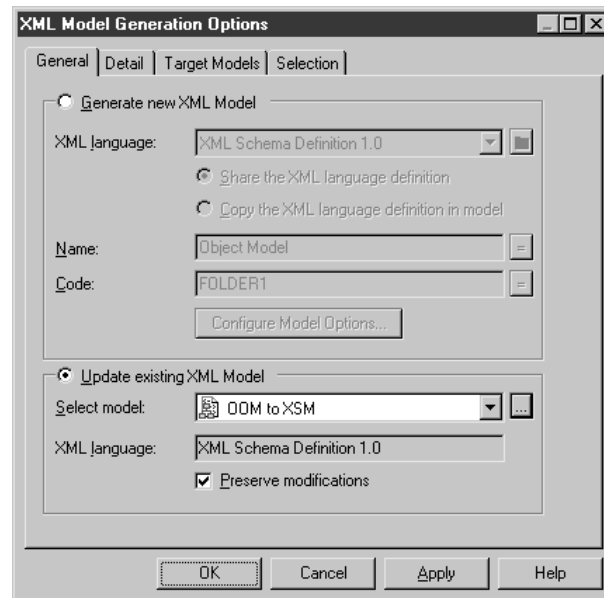
- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.

If you do not have an XML model in the current workspace, the Update existing XML Model option is not available.

- 2 Select the Update existing XML Model radio button.
- 3 Select a model from the Select model dropdown listbox, if the current model has already been generated.

or

Click the Ellipsis button, beside the Select model dropdown listbox, and select an existing model in the Select a model dialog box.



Preserve modifications

Select the Preserve modifications check box if you want to preserve objects in the XML model.

If you clear this check box, all existing objects in the XML model will be removed from the model, leaving only the objects generated from the OOM.

- 4 Click the Detail tab to define options and generation parameters.
- 5 Click the Target Models tab to select the target models of shortcuts in the current model.
- 6 Click the Selection tab to display the Selection page.
- 7 Select the name of an OOM from the Select Location dropdown listbox. The default XML model is generated from this PDM.

Generating an XML model from a package

To generate an XML model from a package, select the package name from the dropdown listbox in the upper left corner of the dialog box. To generate an XML model from a sub-package, select a sub-package from the dropdown listbox in the upper left corner of the dialog box, or select a package name and click the Include Sub-Objects tool next to this dropdown listbox.


- 8 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.
- 9 Click OK.

If you selected the Preserve modifications check box, the Merge Models window appears.

If you cleared the Preserve modifications check box, the updated XML model diagram appears in the work area.

Merging models

The Merge Models dialog box shows the newly generated XML model in the left pane, and the existing XML model in the right pane. You can select or clear object check boxes in the right pane for XML model objects that you want to include or delete in the model to be merged.

 For more information on merging models, see Comparing and Merging Models in the *General Features Guide*.

Editing an XML model report

You can edit a report of your XML model.

The Report Editor allows you to use a predefined report template or to build your own report.

↪ For more information on reports, see the *Reports User's Guide*.

What is a report?

A report is an easy to consult document that shows parts or the global content of a model. You can print a report in order to have it on a paper support, or you can generate it in RTF or HTML format for a better reusability.

Creating an XML model report

You create an XML model report using the Report Editor. You cannot open the Report Editor without at least one model opened in the workspace.

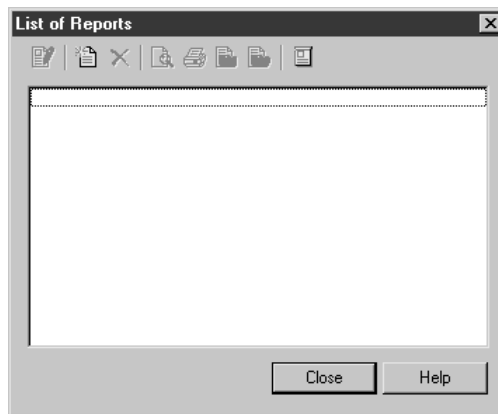
When you create a model report, you need the following information:

Option	Description
Report name	Name of the report provided by default
Language	Language in which the report can be printed. English is the default language
Report template	List of available templates related to the opened model

❖ **To create a model report:**

- 1 Select Model→Reports.

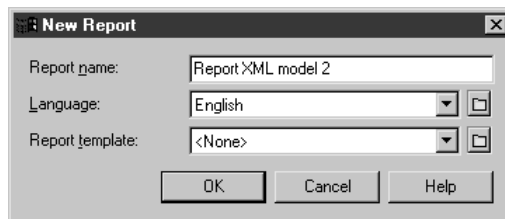
The List of Reports appears. It displays the alphabetical list of all reports saved in the model. If there are no existing reports, all the tools in the List of Reports dialog box are grayed, except for the New Report tool.



- 2 Select the New Report tool.

The New Report dialog box appears. It displays a default report name.

- 3 Type a report name in the Report name box, if you want to modify the default name.



- 4 Select a language from the Language dropdown listbox. It displays the available languages. English is the default language in which the report is printed.
- 5 Select a template from the Report template dropdown listbox. The list displays the templates related to the XML model you are working on.
or
Select None, if you do not want any particular template.

Language in a template

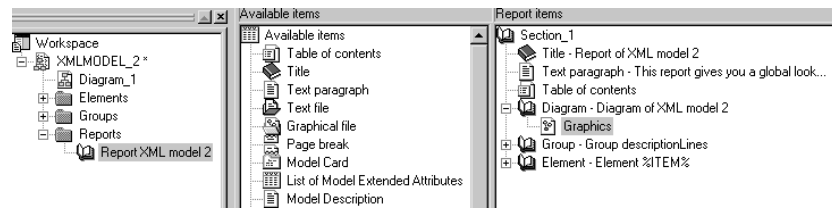
When you use a template created in a language different from the one you select to create your report, only user-defined items such as Title or Text paragraph will keep the language of the template. Other items will be displayed in the report language.

- 6 Click OK.

The Report Editor window appears. A section is created by default at the bottom of the Report items pane, which is filled with the template you have selected. If you did not select a template, the Report items pane is empty. Simultaneously a report node appears under the Reports category in the Browser.

For more information on the Report Editor, see chapter Using the Report Editors in the *Reports User's Guide*.

- 7 Build the report structure in the Report items pane.



For more information on building a report structure, see chapter Building Reports in the *Reports User's Guide*.

- 8 Select File→Save to save your report.
- 9 Click the Print, Generate RTF or Generate HTML tool in the Report toolbar.

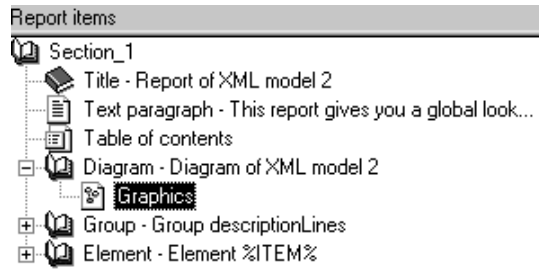
How can a report underline the hierarchical structure of an XML model?

Besides the paragraph numbering, the most efficient way to emphasize the hierarchical structure of an XML model is to insert diagram graphics in your report.

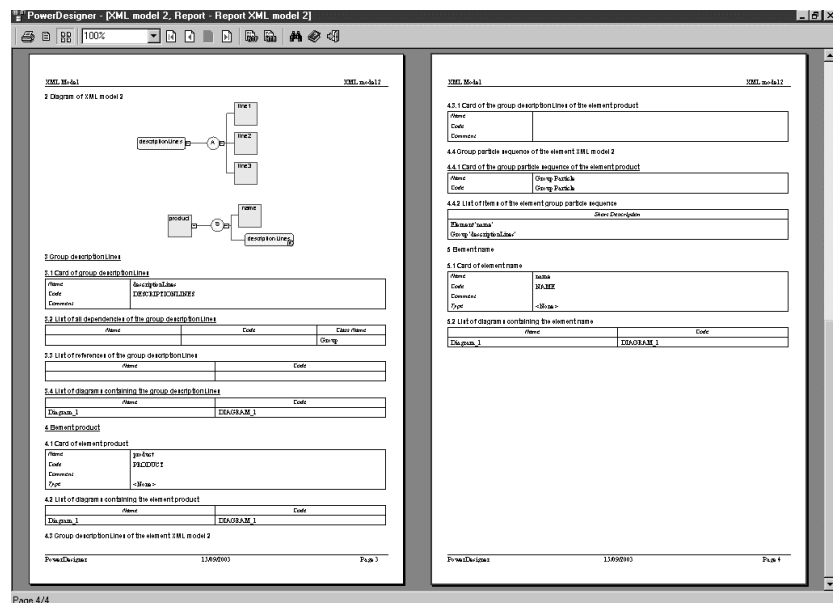
Inserting diagram graphics

You can add diagram graphics to your report by inserting the Graphics item in the Report items pane.

In the following structure, the graphical representation of the diagram will appear in the report just before the descriptive paragraphs.



To have a preview of your report, click the Print Preview tool in the Report Editor window.



CHAPTER 4

Generating and reverse engineering an XSD, a DTD or an XDR file

About this chapter This chapter describes how to generate and reverse engineer an XML Schema Definition file (.XSD), a Document Type Definition file (.DTD) or an XML-Data Reduced file (.XDR).

The same procedures apply to all XML languages.

Contents

Topic	Page
Generating an XSD, a DTD or an XDR file	202
Reverse engineering an XSD, a DTD or an XDR file	208

Generating an XSD, a DTD or an XDR file

PowerDesigner allows you to generate an XSD, a DTD or an XDR file from an XML model.

File preview

You can have a preview of an XSD, a DTD or an XDR file by clicking the Preview tab of an XML model property sheet.

Why generate an XSD file?

You can generate an XSD file for different reasons:

- ◆ It is XML-written, so it can be read by any plain text editor
- ◆ It describes the structure and data types of an XML document
- ◆ It respects the W3C Recommendation for XML Schema

Why generate a DTD file?

You can generate a DTD file for different reasons:

- ◆ It is XML-written, so it can be read by any plain text editor
- ◆ It describes the content of an XML document
- ◆ It respects the W3C XML Specification DTD

Why generate an XDR file?

You can generate an XDR file for different reasons:

- ◆ It is XML-written, so it can be read by any plain text editor
- ◆ It is a simplified XSD file
- ◆ It respects the W3C Note on XML-Data

Defining generation parameters

You can define various parameters before generating an XSD, a DTD or an XDR file.

Check model

You can check your model before generation if you select the Check model check box. This check box is available for all XML languages. The generation stops if an error is found. You must correct this error before starting the generation again.

🔗 For more information on the Check model feature, see section Checking an XML model in chapter Working with an XML model.

Defining generation targets

When you start the generation process, you can choose to:

- ◆ Generate for a selected XML language
- ◆ Use an extended model definition (.XEM) that extends the XML language

To use an extended model definition that extends the XML language, you must have an extended model definition attached to the model. You can either create a new extended model definition or import an existing extended model definition in the model.

🔗 For more information on how to create or import an extended model definition, see sections Creating an extended model definition, or Importing an extended model definition into a model, in chapter Extended Model Definitions Reference Guide in the *Advanced User Documentation*.

If you use an extended model definition to extend the XML language, you must use an extended model definition designed for this kind of generation.

To do this, you must verify that the value Complement Language Generation is selected in the extended model definition editor.

When you attach an extended model definition to your model, the Targets page in the Generation dialog box displays the target extended model definition that can be selected for generation.

🔗 For more information on the generation targets, see section Generation targets, in chapter Extended Model Definitions Reference Guide, in the *Advanced User Documentation*.

Defining generation options

If you do not see the Options page in the Generation dialog box, it means that no option has been defined in the corresponding XML language (.XSL file).

You can however customize your XML language at any time, and add any options you may need during generation. You define generation options in the Generation/Options category of the XML language available from Language→Edit Current Language.

If the Generation category does not appear, right-click the name or the icon of the XML language in the tree view, and select Add Items in the contextual menu. In the Categories list of the Selection dialog box, select Generation and click OK. Right-click the Generation category and select New→Option in the contextual menu.

Defining generation tasks

The Tasks page contains standard commands defined in the XML language. You can select any tasks to execute at generation time.

If you do not see the Tasks page in the Generation dialog box, it means that no command has been defined in the corresponding XML language (.XSL file).

You can however customize tasks for your current language. You define generation tasks and commands in the Generation/Tasks category and the Generation/Commands category of the XML language available from Language→Edit Current Language.

If the Generation category does not appear, right-click the name or the icon of the XML language in the tree view, and select Add Items in the contextual menu. In the Categories list of the Selection dialog box, select Generation and click OK. Right-click the Generation category and select New→Task in the contextual menu.

Using the Move up or down arrows

You can easily modify the order of commands execution using the Move up or Move down arrows. You can only select one value check box at a time.

How to generate an XSD, a DTD or an XDR file?

The type of generated file depends on which XML language the model has been targeted with:

Model targeted with	Generated file
XML Schema Definition 1.0	XSD
Document Type Definition 1.0	DTD
XML-Data Reduced 1.0	XDR

The XML language of an XML model is displayed in the status bar (bottom-right corner of the screen) and in the model property sheet.

For more information on changing the XML language of an XML model, see section Changing the XML language of an XML model in chapter XML Model Basics.

Parameter entities during DTD generation

Parameter entities are references to predefined values within a DTD file (See Parameter property in entity property sheet).

During DTD generation, some object properties containing inadvertently parameter values will be generated with parameter references. If you are not satisfied with this default use of parameter entities, you should clear the Parameter property before generation.

❖ To generate an XSD, a DTD or an XDR file from an XML model:

- 1 Select Language→Generate XML Schema Definition File.

or

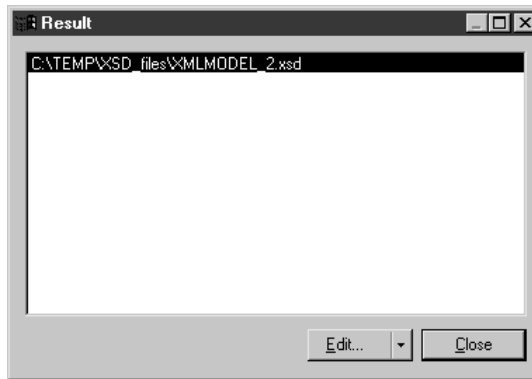
Language→Generate Document Type Definition File.

or

Language→Generate XML-Data Reduced File.

The Generation dialog box appears.

A Result box appears indicating that the XSD, DTD or XDR file has been generated in the destination directory.



The result is also displayed in the Generation page of the Output window, located in the bottom part of the main window.

- 5 Click Edit to edit the XSD, DTD or XDR file with a plain text editor.



- 6 Click Close in the Result box.

Reverse engineering an XSD, a DTD or an XDR file

PowerDesigner allows you to reverse engineer an XML Schema Definition file, a Document Type Definition file or an XML-Data Reduced file to create or update an XML model.

What is reverse engineering?

Reverse engineering is the process of examining and recovering data or source code from a file that is then used to build or update an XML model.

You can reverse engineer objects to a new model, or to an existing model. When you reverse engineer an object which already exists in a model, you use an object comparison box to choose either to replace or to keep the existing object in the model.

Parsing

PowerDesigner uses a parser software for XML reverse engineering, developed by the Apache Software Foundation (<http://www.apache.org/>).

Why reverse engineer an XSD, a DTD or an XDR file?

You can reverse engineer an XSD, a DTD or a XDR file for different reasons:

- ◆ To have a global view of the XSD, DTD or XDR file through a diagram
- ◆ To check, delete or improve the XSD, DTD or XDR file through the graphic interface

How to reverse engineer an XSD, a DTD or an XDR file?

You can define options and target models when reverse engineering an XSD, a DTD or an XDR file.

Reverse engineering options

The Options page of the Reverse Engineering dialog box displays the following options:

Option	Description
Show symbols	If selected, the generated model appears in the diagram (not only in the Browser tree view)
Expand nodes	If selected, global objects (with no parent object in the diagram) appear in the diagram with their nodes expanded (child objects)
Show elements	If selected, the elements appear in the diagram
Show groups	If selected, the groups appear in the diagram
Show complex types	If selected, the complex types appear in the diagram. Only with XSD files
Show simple types	If selected, the simple types appear in the diagram. Only with XSD files
Convert unique references to elements	If selected, target objects are expanded in place of single referencing objects. With this option, global objects with a single reference in the model are converted into child objects. Do not use this option if you want to keep the global scope of some objects

Convert Unique References

The Convert Unique References feature is also available from the Tools menu.

Reverse engineering target models

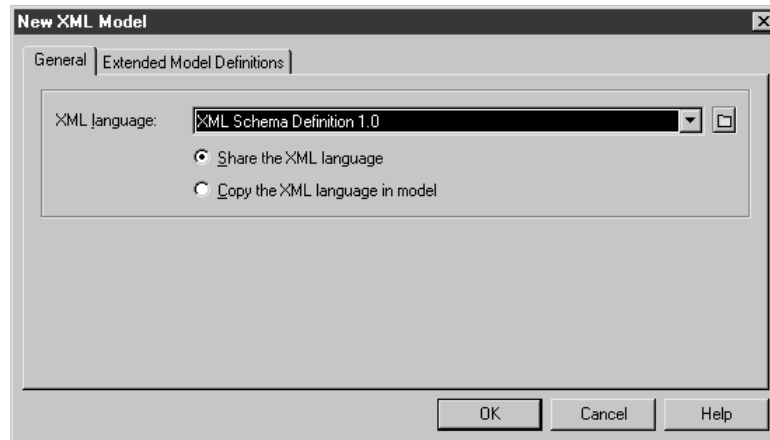
The Target Models page of the Reverse Engineering dialog box displays a list for target models. Select the Add Models tool if you want to maintain the shortcuts defined between the XSD, DTD or XDR file you are reverse engineering and other XML models.

Reverse engineering to a new XML model


❖ To reverse engineer an XSD, a DTD or an XDR file to a new XML model:

- 1 Select File→Reverse Engineer→XML Definition.

The New XML Model dialog box appears.



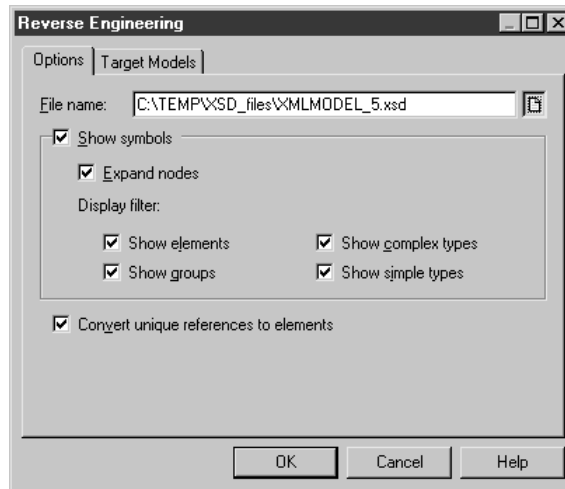
- 2 In the General page, select an XML language in the XML language dropdown listbox.
- 3 Select the Share or Copy radio button.
- 4 <optional> In the Extended Model Definitions page, you can select one or several extended model definitions to be reverse engineered with the XSD, DTD or XDR file.

 For more information on extended model definitions, see chapter Extended Model Definitions Reference Guide in the *Advanced User Documentation*.

- 5 Click OK.

The Reverse Engineering dialog box appears.

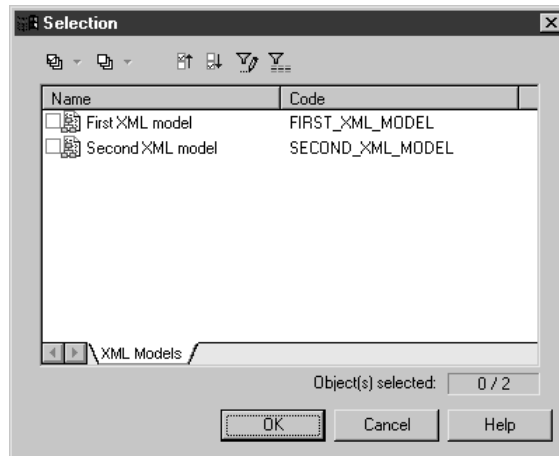
- 6 In the Options page, click the Select a File tool beside the File name box to select the XSD, DTD or XDR file you want to reverse engineer.



Select or clear options (See Reverse engineering options).

- 7 In the Target Models page, select the Add Models tool if you want to maintain the shortcuts defined between the XSD, DTD or XDR file you are reverse engineering and other XML models.

A Selection dialog box appears with a list of XML models.



For more information on shortcuts, see section Target models parameters in chapter Generating from an XML model.

- 8 Select the XML models you want to attach as target models.

- 9 Click OK.

The XML diagram corresponding to the reverse engineered XSD, DTD or XDR file appears in the diagram window. The Browser tree view reveals the new XML model with all its objects. The result is also displayed in the Reverse page of the Output window, located in the bottom part of the main window.

Reverse engineering to an existing XML model

❖ To reverse engineer an XSD, a DTD or an XDR file to an existing XML model:

- 1 Select File→Open.

The Open file dialog box appears.

- 2 Select an existing XML model (.XSM) in the directory of your choice.

- 3 Click Open.

The existing XML model appears in the Browser tree view and the diagram window.

- 4 Select Language→Reverse Engineer XML Schema Definition File.

or

Language→Reverse Engineer Document Type Definition File.

or

Language→Reverse Engineer XML-Data Reduced File.

The Reverse Engineering dialog box appears.

- 5 In the Options page, click the Select a File tool beside the File name box to select the XSD, DTD or XDR file you want to reverse engineer.

Select or clear options (See Reverse engineering options).

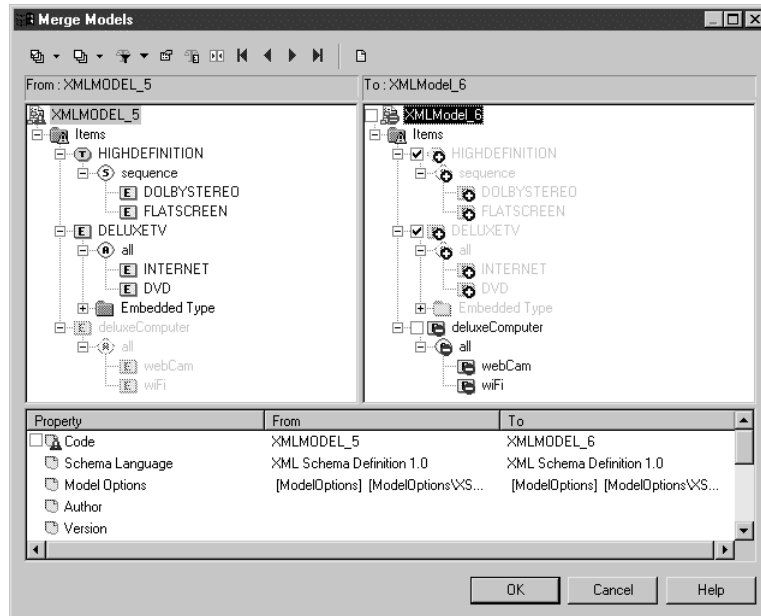
- 6 In the Target Models page, select the Add Models tool if you want to maintain the shortcuts defined between the XSD, DTD or XDR file you are reverse engineering and other XML models.

A Selection dialog box appears with a list of XML models.

- 7 Select the XML models you want to attach as target models.

- 8 Click OK.

The Merge Models dialog box appears, with the newly generated XML model in the left pane, and the existing XML model in the right pane. You can select or clear object check boxes in the right pane for objects that you want to include or delete in the model to be merged.



For more information on merging models, see section Merging models, in chapter Comparing and Merging Models of the *General Features Guide*.

- 9 Click OK.

The existing XML model (in the Browser tree view and the diagram window) is updated by the merge with the reverse engineered XSD, DTD or XDR file. The result is also displayed in the General page of the Output window, located in the bottom part of the main window.

Auto-layout

Select Symbol→Auto-Layout to help you display the new diagram.

CHAPTER 5

Exchanging data with databases supporting XML

About this chapter This chapter describes how to use an XML model to store or retrieve data in databases supporting XML.

Contents

Topic	Page
Why use XML in databases?	216
Generating an annotated schema for Microsoft SQL Server 2000	217
Generating an annotated schema for Oracle 9i2	230
Generating a DAD file for IBM DB2	237
Generating SQL/XML queries	249

Why use XML in databases?

XML is becoming a universal data exchange format. An XML file can be read with any plain text editor. Most of relational databases (RDB) now support XML so that you can store or retrieve data through XML files.

With an XML model, you can generate an **annotated schema** that will allow you to store or retrieve data in a relational database supporting XML. To generate an annotated schema, you need to attach the appropriate extended model definition (XEM file) to an XML model mapped (or not) to a PDM.

The following table lists the databases supporting XML for which you can generate annotated schemas:

Database	Mapped XML model	Targeted XML language	Required XEM file
Microsoft SQL Server 2000	Yes	XSD or XDR	Microsoft SQL Server
Oracle 9i2	No	XSD	Oracle 9i2
IBM DB2 v8.1	Yes	DTD	IBM DB2 DAD

By attaching the SQL/XML extended model definition to an XML model mapped to a PDM, you can also generate **SQL/XML queries** to retrieve data in an XML format, from relational databases supporting SQL/XML.

The best way to generate SQL/XML queries is to use the XML Builder Wizard which helps you build an XML model from a PDM. The generated XML model is mapped to the PDM and automatically linked to the SQL/XML extended model definition. (See Generating SQL/XML queries)

Generating an annotated schema for Microsoft SQL Server 2000

Microsoft SQL Server 2000 is an XML-enabled database server. It supports annotations that can be used on XSD or XDR files, to map XML data to relational data.

An **annotated schema** is an XML-coded file, targeted with an XML language and tagged with specific DBMS annotations, that allows you to store or retrieve data in an XML format, from relational databases supporting XML. An XML model allows you to generate an annotated schema (XSD or XDR) for SQL Server 2000. You need first to **map** an XML model to a PDM, then to attach the Microsoft SQL Server **extended model definition** to the mapped XML model. If need be, you can complete the mapping by defining **extended attributes** to XML objects.

Mapping XML objects to PDM objects

There are several ways to map XML objects to PDM objects:

- ◆ Using the XML Builder Wizard (See following procedure)
- ◆ Generating an XML model from a PDM
- ◆ Generating a PDM from an XML model
- ◆ Manually

The best way to map an XML model to a PDM is to use the XML Builder Wizard.

If you do not have a PDM, you can reverse engineer a database into a PDM.

☞ For more information on reverse engineering a database into a PDM, see chapter Reverse engineering in a PDM, in the *PDM User's Guide*.

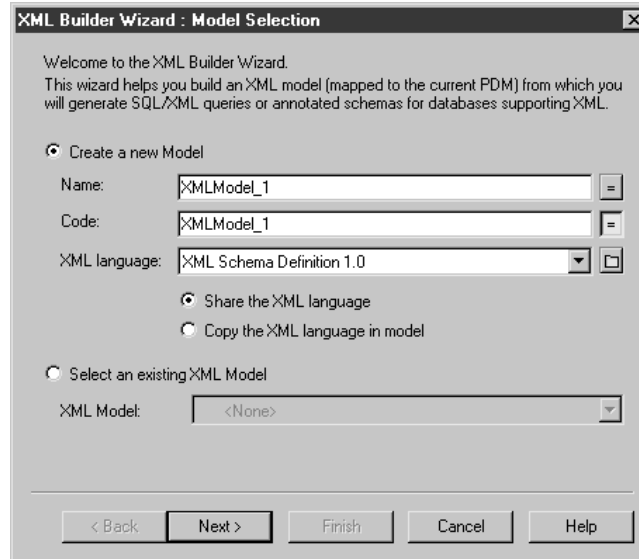
Caution

*The following procedure assumes you have a PDM open in the workspace and targeted with **Microsoft SQL Server 2000** as DBMS. If the PDM is not targeted with the proper DBMS, select Database → Change Current DBMS.*

❖ **To generate an annotated schema by mapping through the XML Builder Wizard:**

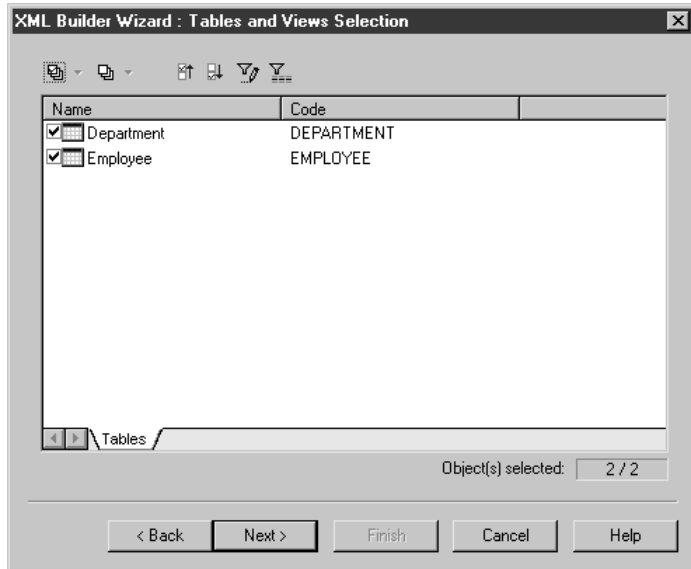
- 1 In the PDM menu bar, select Tools→XML Builder Wizard.

The Model Selection dialog box appears.



- 2 Select the **new model** option. Type a name and a code for the new model, and select XML Schema Definition 1.0 or XML-Data Reduced 1.0 in the XML language dropdown listbox.
or
Select the **existing model** option. Select a model in the XML Model dropdown listbox. This model must be targeted with XSD or XDR.
- 3 Click Next.

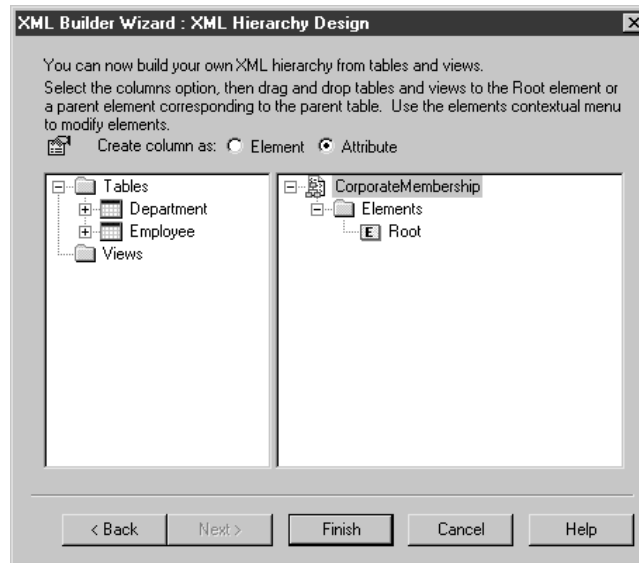
The Tables and Views Selection dialog box appears with the PDM tables list.



All tables are selected by default.

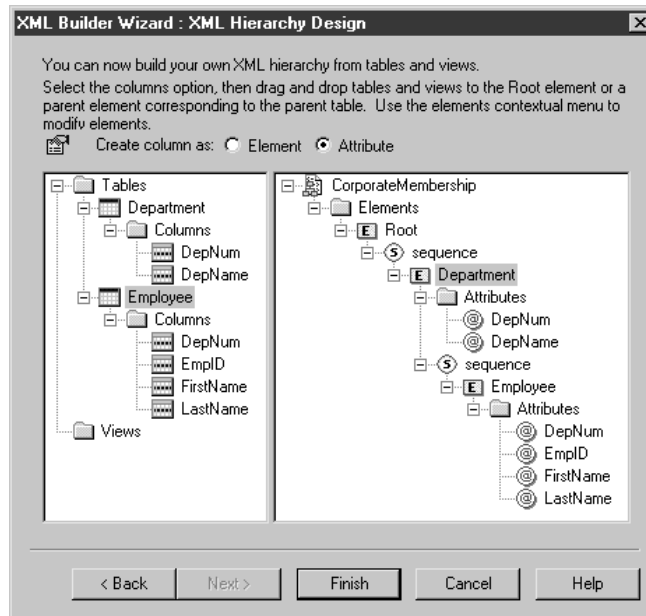
- 4 <optional> Click the **Deselect All** tool and select the tables you want to generate into XML elements.
- 5 Click Next.

The XML Hierarchy Design dialog box appears.



- 6 Click the **Element** radio button, if you want to create columns as elements.
or
Click the **Attribute** radio button, if you want to create columns as attributes.

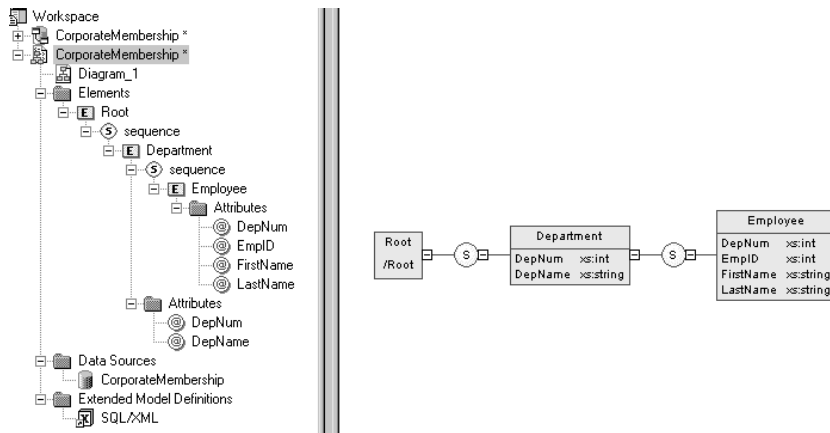
- 7 Drag and drop tables from left to right panel, and build an XML hierarchy.



For more information on building an XML hierarchy, see section [Generating an XML model via the XML Builder Wizard](#), in chapter [Generating from a PDM](#), in the *PDM User's Guide*.

- 8 Click Finish.

The new XML model appears in the workspace with the generated elements and attributes.



In the case of an existing XML model, the generated elements appear next to the existing elements.

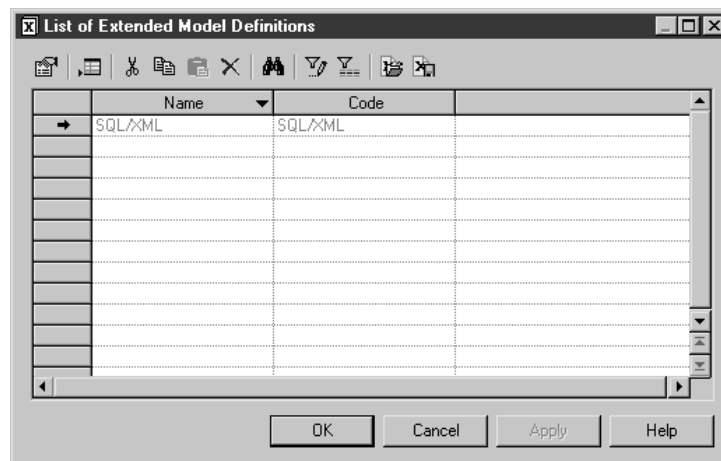
Extended model definitions

The **SQL/XML** extended model definition is automatically attached to the generated XML model.

You can attach the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema. (See Note in step 11)

- 9 In the XML model menu bar, select Model→Extended Model Definitions.

The List of Extended Model Definitions appears.

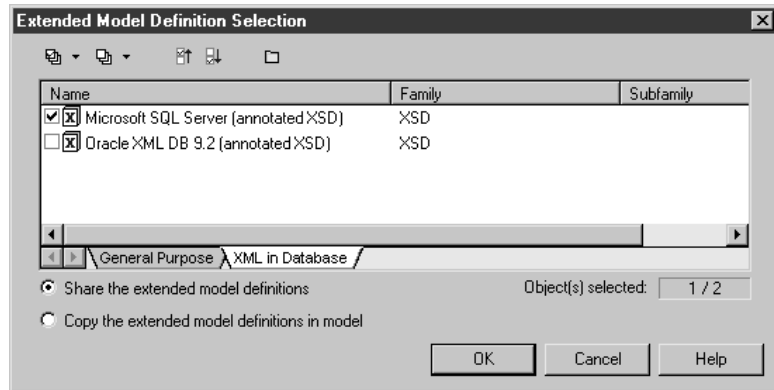


- 10 Click the **Import an Extended Model Definition** tool.



The Extended Model Definition Selection dialog box appears.

- 11 In the **XML in Database** page, select Microsoft SQL Server.



Note: In the General Purpose page, you can select the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema.

- 12 Click OK.

Microsoft SQL Server appears in the List of Extended Model Definitions.

- 13 Click OK.

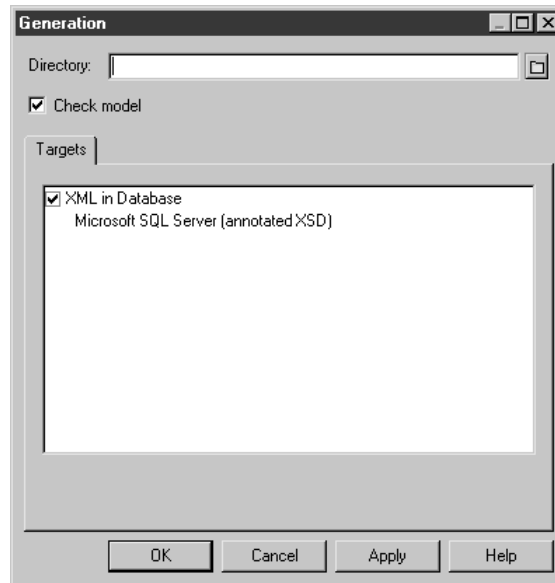
The Microsoft SQL Server extended model definition appears in the Browser tree view, attached to the generated XML model.

Annotated schema preview

By clicking the Preview page of the model property sheet, you can have a preview of the annotated schema.

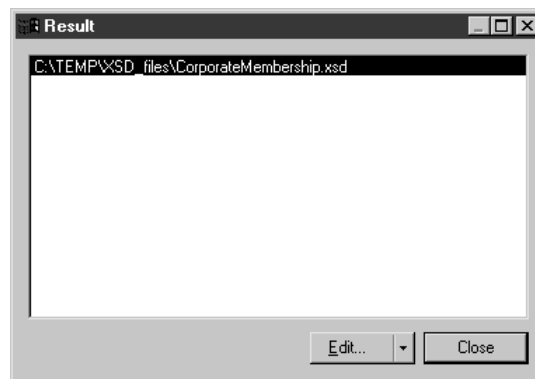
- 14 Select Language→Generate XML Schema Definition File.
or
 Select Language→Generate XML-Data Reduced File.

The Generation dialog box appears with Microsoft SQL Server selected in the Targets page.



- 15 Click the **Select a Path** button, beside the Directory box, to select a path for the annotated schema file.
- 16 Click OK.

The Result dialog box appears with the path of the annotated schema file selected.



- 17 Click Edit.

The annotated schema appears in the editor window.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xs:element name="root" sql:is-constant="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DEPARTMENT" sql:relation="DEPARTMENT">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="EMPLOYEE" sql:relation="EMPLOYEE">
                <xs:annotation>
                  <xs:appinfo>
                    <sql:relationship parent="DEPARTMENT" parent-key="DEPNUM" child-key="DEPNUM" child="EMPLOYEE"/>
                  </xs:appinfo>
                </xs:annotation>
              <xs:complexType>
                <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
                </xs:attribute>
                <xs:attribute name="EMPID" type="xs:int" sql:field="EMPID">
                </xs:attribute>
                <xs:attribute name="FIRSTNAME" type="xs:string" sql:field="FIRSTNAME">
                </xs:attribute>
                <xs:attribute name="LASTNAME" type="xs:string" sql:field="LASTNAME">
                </xs:attribute>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
          </xs:attribute>
          <xs:attribute name="DEPNAME" type="xs:string" sql:field="DEPNAME">
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Note the SQL namespace (with the sql prefix) and the SQL annotations for tables (sql:relation), columns (sql:field) and reference (sql:relationship).

18 Click Close in the Result dialog box.

Reinforcing mapping with extended attributes

If you want to modify or reinforce the mapping resulting from the the XML Builder Wizard, you can define extended attributes for XML objects.

Extended attributes are available from the Microsoft SQL Server extended model definition attached to the mapped XML model. If the element and attribute names match with the table and column names, you do not need to define extended attributes for XML objects.

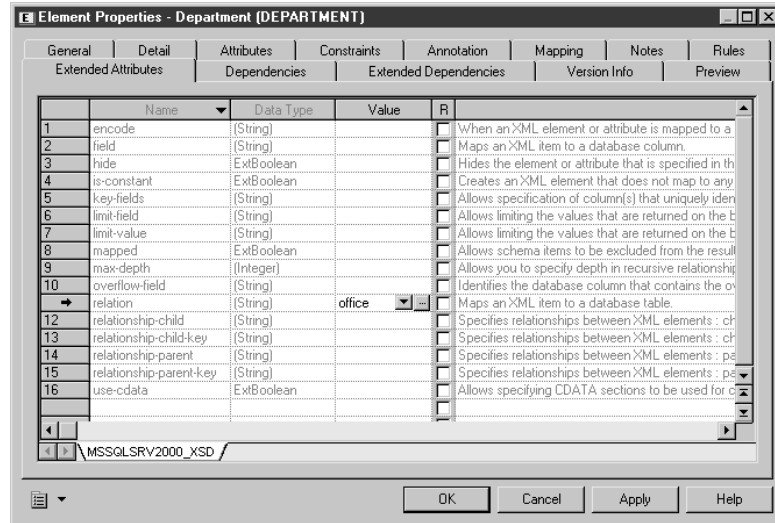
You could generate an annotated schema without a PDM and the XML Builder Wizard, only through extended attributes, but it would not be the easiest way to proceed.

Caution

The following procedure assumes you have an XML model open in the workspace, targeted with XSD or XDR, mapped with a PDM, and linked to the Microsoft SQL Server extended model definition. (See Mapping XML objects to PDM objects)

❖ **To generate an annotated schema by reinforcing mapping with extended attributes:**

- 1 Double-click an element symbol in the diagram to display its property sheet.
- 2 In the Extended Attributes page, type the name of a database **column** in the Value column of the **field** annotation, if you want to map the element with a database column.
or
Type the name of a database **table** in the Value column of the **relation** annotation, if you want to map the element with a database table.



You can define a value for the following annotations regarding an element (E) and/or an attribute (A):

Annotation	E	A	Description
encode	✓	✓	When an XML element or attribute is mapped to a SQL Server BLOB column, allows requesting a reference (URI) to be returned and used later to return BLOB data
field	✓	✓	Maps an XML item to a database column
hide	✓	✓	Hides the element or attribute specified in the schema in the resulting XML document
is-constant	✓	—	Creates an XML element that does not map to any table. The element appears in the query output

Annotation	E	A	Description
key-fields	✓	—	Allows specification of columns that uniquely identify the rows in a table
limit-field	✓	✓	Allows limiting the values that are returned on the basis of a limiting value
limit-value	✓	✓	Allows limiting the values that are returned on the basis of a limiting value
mapped	✓	✓	Allows schema items to be excluded from the result
max-depth	✓	—	Allows you to specify depth in recursive relationships that are specified in the schema
overflow-field	✓	—	Identifies the database column that contains the overflow data
relation	✓	—	Maps an XML item to a database table
relationship-child	✓	—	Specifies an element as the child table in a reference (To define only in the child element property sheet)
relationship-child-key	✓	—	Specifies an attribute as the foreign key of a child table in a reference (To define only in the child element property sheet)
relationship-parent	✓	—	Specifies an element as the parent table in a reference (To define only in the child element property sheet)
relationship-parent-key	✓	—	Specifies an attribute as the primary key of a parent table in a reference (To define only in the child element property sheet)
use-cdata	✓	—	Allows specifying CDATA sections to be used for certain elements in the XML document
prefix	—	✓	Creates valid XML ID, IDREF, and IDREFS. Prepends the values of ID, IDREF, and IDREFS with a string

- 3 Click OK.
- 4 Repeat steps 1 to 3 for each element or attribute you want to map with a database table or column, or for which you want to define extra annotations.

- 5 Select Language→Generate XML Schema Definition File.
or
Select Language→Generate XML-Data Reduced File.

The Generation dialog box appears with Microsoft SQL Server selected in the Targets page.



- 6 Click the **Select a Path** button, beside the Directory box, to select a path for the annotated schema file.
- 7 Click OK.

The Result dialog box appears with the path of the annotated schema file selected.



8 Click Edit.

The annotated schema appears in the editor window.

```
?xml version="1.0" encoding="UTF-8" ?>
xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xs:element name="Root" sql:is-constant="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DEPARTMENT" sql:relation="office">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="EMPLOYEE" sql:relation="clerk">
                <xs:annotation>
                  <xs:appinfo>
                    <sql:relationship parent="office" parent-key="officeNum" child="clerk" child-key="officeNum" />
                  </xs:appinfo>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="DEPNUM" type="xs:int" sql:field="officeNum">
              </xs:attribute>
            <xs:attribute name="EMPID" type="xs:int" sql:field="clerkID">
              </xs:attribute>
            <xs:attribute name="FIRSTNAME" type="xs:string" sql:field="clerkFName">
              </xs:attribute>
            <xs:attribute name="LASTNAME" type="xs:string" sql:field="clerkLName">
              </xs:attribute>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="DEPNUM" type="xs:int" sql:field="officeNum">
          </xs:attribute>
        <xs:attribute name="DEPNAME" type="xs:string" sql:field="officeName">
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Note the SQL namespace (with the sql prefix) and the SQL annotations for tables (sql:relation), columns (sql:field) and reference (sql:relationship).

9 Click Close in the Result dialog box.

Generating an annotated schema for Oracle 9i2

Oracle 9i2 is a database server with a native XML storage and retrieval technology called Oracle XML DB. There is no mapping between XML data and relational data. Tables, columns and abstract data types (ADT) are created from **annotated schemas** (XSDs). Annotated schemas are XML-coded files, targeted with an XML language and tagged with specific DBMS annotations, that allow you to store or retrieve data in an XML format, from relational databases supporting XML.

An XML model allows you to generate an annotated schema (XSD) for Oracle 9i2. You just need to attach the Oracle 9i2 **extended model definition** to the XML model. Oracle 9i2 uses by default the name of the XML elements present in the annotated schema to generate SQL objects. You can override the creation of SQL objects by defining **extended attributes** for elements, complex types and the XML model.

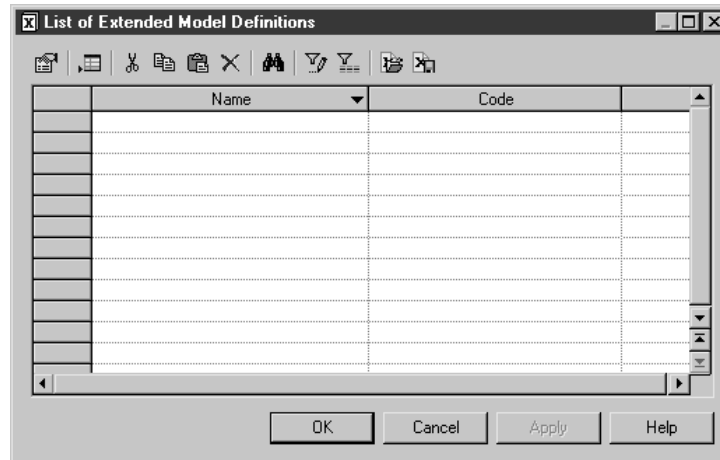
Caution

The following procedure assumes you have an XML model open in the workspace and targeted with XSD.

❖ **To generate an annotated schema for Oracle 9i2:**

- 1 Select Model→Extended Model Definitions.

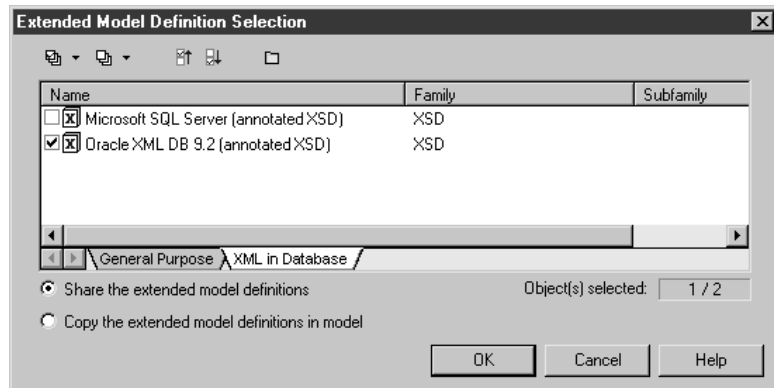
The List of Extended Model Definitions appears.



- 2 Click the Import an Extended Model Definition tool.

The Extended Model Definition Selection dialog box appears.

- 3 In the XML in Database page, select Oracle XML DB 9.2.

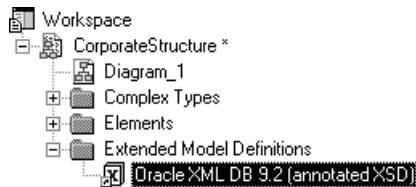


- 4 Click OK.

Oracle XML DB 9.2 appears in the List of Extended Model Definitions.

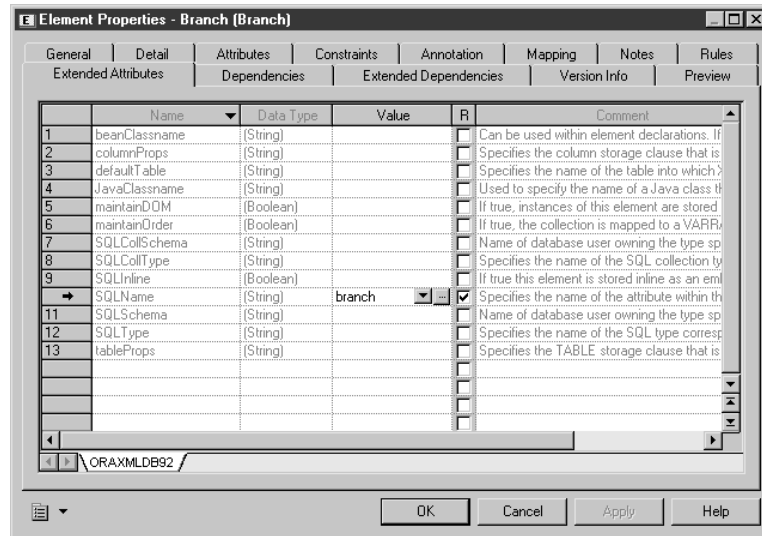
- 5 Click OK.

The extended model definition appears in the Browser tree view, in a folder attached to the XML model.



- 6 Double-click an **element** symbol in the diagram to display its property sheet.

- In the Extended Attributes page, type the name of a table or a column in the Value column of the **SQLName** annotation, if you want to create a **table** or a **column** from the selected element.



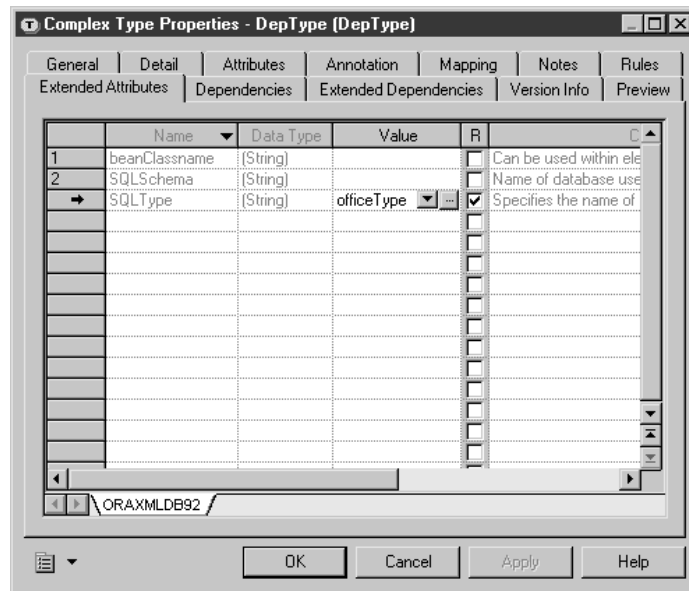
You can define a value for the following annotations:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
columnProps	Specifies the column storage clause that is inserted into the default CREATE TABLE statement. It is useful mainly for elements that are mapped to tables, namely top-level element declarations and out-of-line element declarations
defaultTable	Specifies the name of the table into which XML instances of this schema should be stored. This is most useful in cases when the XML is being inserted from APIs where table name is not specified (for example, FTP and HTTP)

Annotation	Description
javaClassname	Used to specify the name of a Java class that is derived from the corresponding bean class, to ensure that an object of this class is instantiated during bean access. If a JavaClassname is not specified, Oracle XML DB will instantiate an object of the bean class directly
maintainDOM	If true, instances of this element are stored so that they retain DOM fidelity on output. This implies that all comments, processing instructions, namespace declarations, and so on, are retained in addition to the ordering of elements. If false, the output need not be guaranteed to have the same DOM behavior as the input
maintainOrder	If true, the collection is mapped to a VARRAY. If false, the collection is mapped to a NESTED TABLE
SQLCollSchema	Name of the database user owning the type specified by SQLCollType
SQLCollType	Specifies the name of the SQL collection type corresponding to this XML element that has maxOccurs > 1
SQLInline	If true this element is stored inline as an embedded attribute (or a collection if maxOccurs > 1). If false, a REF (or collection of REFs if maxOccurs > 1) is stored. This attribute will be forced to false in certain situations (like cyclic references) where SQL will not support inlining
SQLName	Specifies the name of the attribute within the SQL object that maps to this XML element
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration
tableProps	Specifies the TABLE storage clause that is appended to the default CREATE TABLE statement. This is meaningful mainly for global and out-of-line elements

- 8 Click OK.
- 9 Repeat steps 6 to 8 for each element you want to generate into a table or a column, or for which you want to define extra annotations.
- 10 Double-click a **complex type** symbol in the diagram to display its property sheet.

- 11 In the Extended Attributes page, type the name of an abstract data type (ADT) in the Value column of the **SQLType** annotation, if you want to create an **ADT** from the selected complex type.



You can define a value for the following annotations:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration

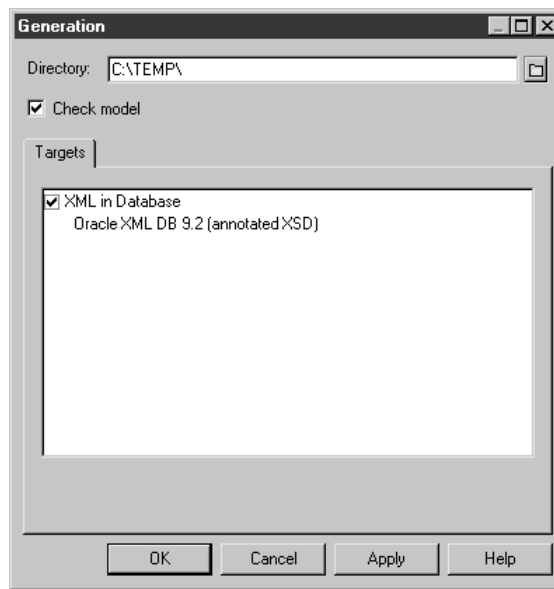
- 12 Click OK.
- 13 Repeat steps 10 to 12 for each complex type you want to generate into an ADT, or for which you want to define extra annotations.
- 14 <optional> In the Browser tree view, double-click the **model** item to display its property sheet.

- 15 In the Extended Attributes page, you can select a value (false or true) for the following annotations:

Annotation	Description
mapUnboundedStringToLob	If true, unbounded strings are mapped to CLOB by default. Similarly, unbounded binary data get mapped to BLOB, by default. If false, unbounded strings are mapped to VARCHAR2(4000), and unbounded binary components are mapped to RAW(2000)
storeVarrayAsTable	If true, the VARRAY is stored as a table (OCT). If false, the VARRAY is stored in a LOB

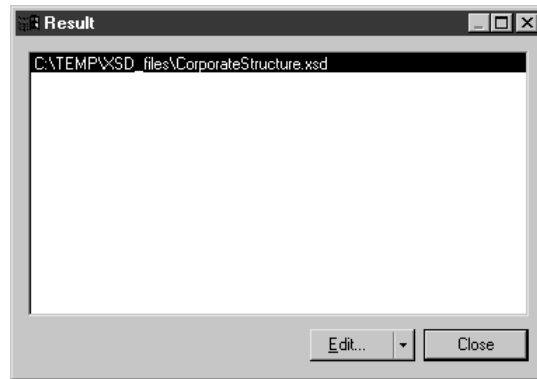
- 16 Click OK.
 17 Select Language→Generate XML Schema Definition File.

The Generation dialog box appears with the extended model definition selected in the Targets page.



- 18 Select a path for the annotated schema with the **Select a Path** tool beside the Directory box.
 19 Click OK.

The Result dialog box appears with the path of the generated schema file.



20 Click Edit.

The annotated schema appears in the editor window.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="http://xmlns.oracle.com/xdm">
  <xs:element name="Branch" sql:SQLName="branch">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Department" type="DepType" sql:SQLName="office"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="DepType" sql:SQLType="officeType">
    </xs:complexType>
  </xs:schema>
```

Note the Oracle namespace (with the sql prefix) and annotations for tables (sql:SQLName) and ADTs (sql:SQLType)

21 Click Close in the Result dialog box.

Generating a DAD file for IBM DB2

IBM DB2 v8.1 (or higher) is a database server with an add-in for XML storage and retrieval called IBM DB2 Extender. XML data (elements, attributes) are mapped to relational data (tables, columns) through Document Access Definition files (.DAD).

There are three types of DAD files:

Type of storage	Type of mapping	Description
Xcolumn	column	The Root element is mapped to a table, and its attributes or child elements are mapped to columns identified by an XPath
Xcollection	SQL	The DAD file starts with a SQL statement for the table mapped to the Root element, and each child element or attribute is mapped to a column or a table name
Xcollection	RDB	A Relational Database node, with a table and a column name, is associated with each attribute or child element of the Root element

An XML model targeted with DTD allows you to generate DAD files for IBM DB2. You need first to **map** an XML model to a PDM, then to attach the IBM DB2 DAD **extended model definition** to the mapped XML model, and finally (optional) to define **extended attributes** for global elements. Extended attributes specify the type of DAD file generated for each global element. The DAD file is generated with the DTD file and a SQL file for stored procedures.

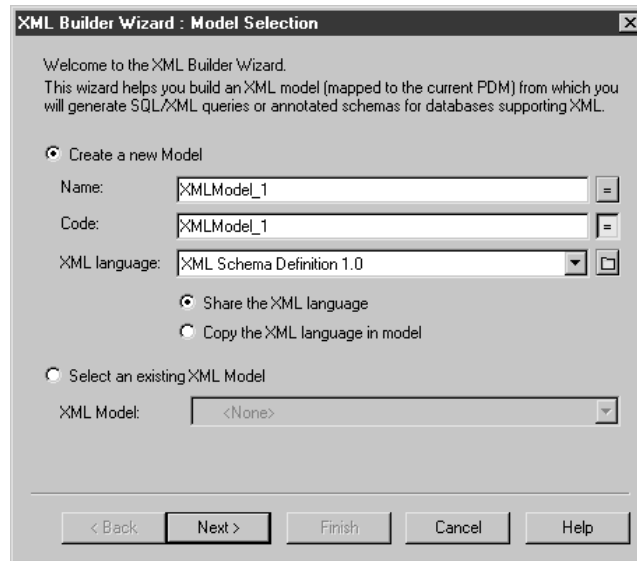
Caution

*The following procedure assumes you have a PDM open in the workspace and targeted with **IBM DB2 UDB 8.x Common Server** as DBMS. If the PDM is not targeted with the proper DBMS, select Database → Change Current DBMS.*

❖ To generate a DAD file for IBM DB2:

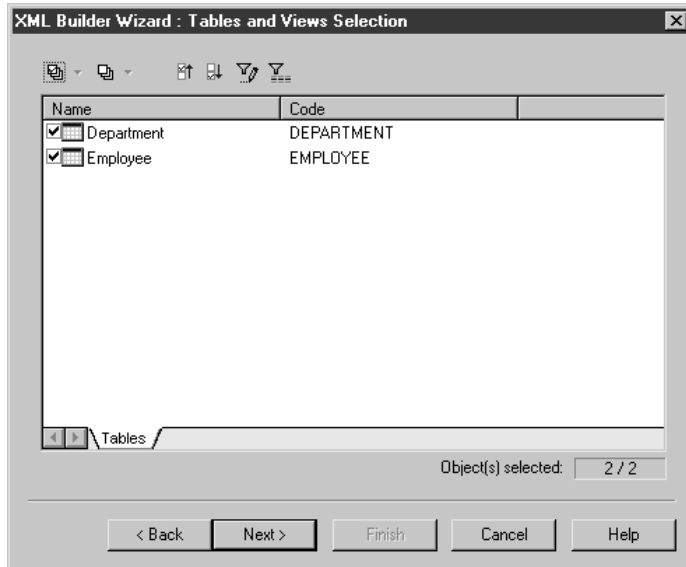
- 1 In the PDM menu bar, select Tools → XML Builder Wizard.

The Model Selection dialog box appears.



- 2 Select the new model option. Type a name and a code for the new model, and select **Document Type Definition 1.0** in the XML language dropdown listbox.
or
Select the existing model option. Select a model in the XML Model dropdown listbox. This model must be targeted with **DTD**.
- 3 Click Next.

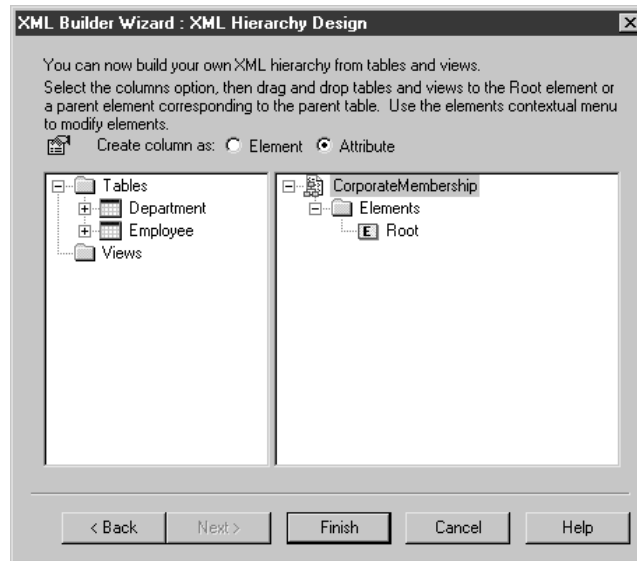
The Tables and Views Selection dialog box appears with the PDM tables list.



All tables are selected by default.

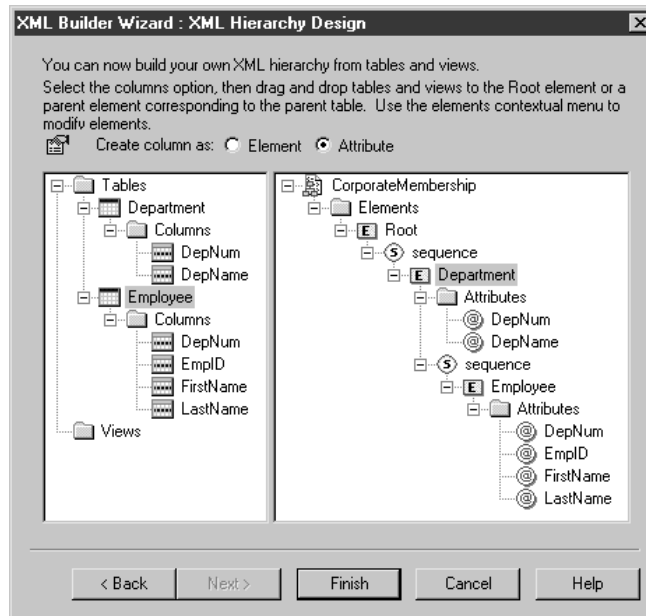
- 4 <Optional> Click the **Deselect All** tool and select the tables you want to generate into XML elements.
- 5 Click Next.

The XML Hierarchy Design dialog box appears.



- 6 Click the **Element** radio button, if you want to create columns as elements.
or
Click the **Attribute** radio button, if you want to create columns as attributes.

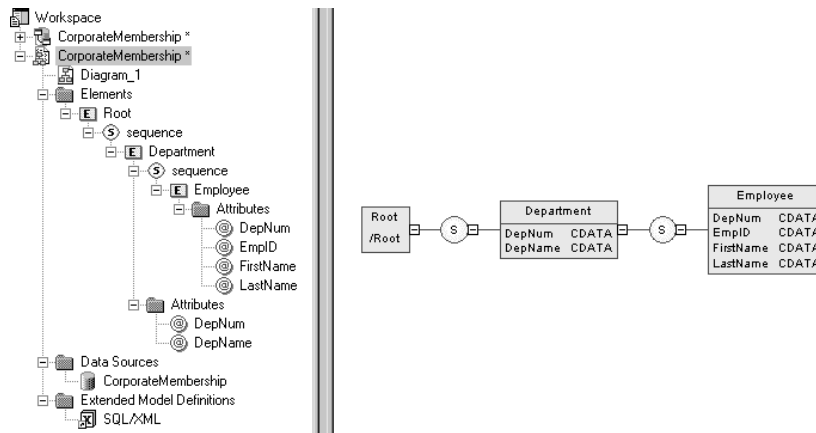
- 7 Drag and drop tables from left to right panel and build an XML hierarchy.



For more information on building an XML hierarchy, see section [Generating an XML model via the XML Builder Wizard](#), in chapter [Generating from a PDM](#), in the *PDM User's Guide*.

- 8 Click Finish.

The new XML model appears in the workspace with the generated elements and attributes.



In the case of an existing XML model, the generated elements appear next to the existing elements.

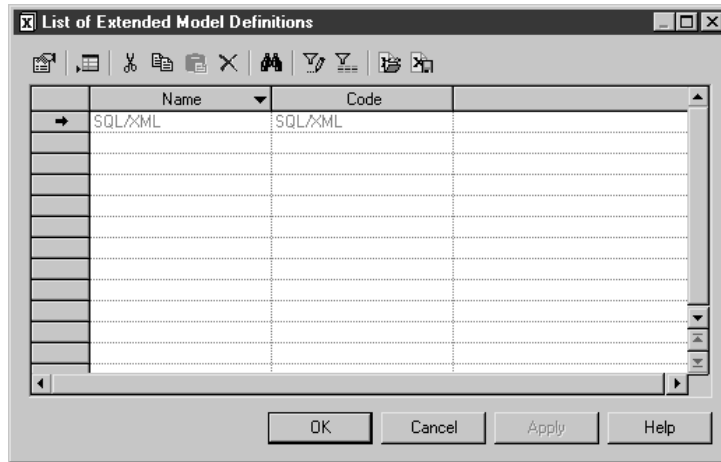
Extended model definitions

The **SQL/XML** extended model definition is automatically attached to the generated XML model.

You can attach the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema. (See Note in step 11)

- 9 Select Model→Extended Model Definitions.

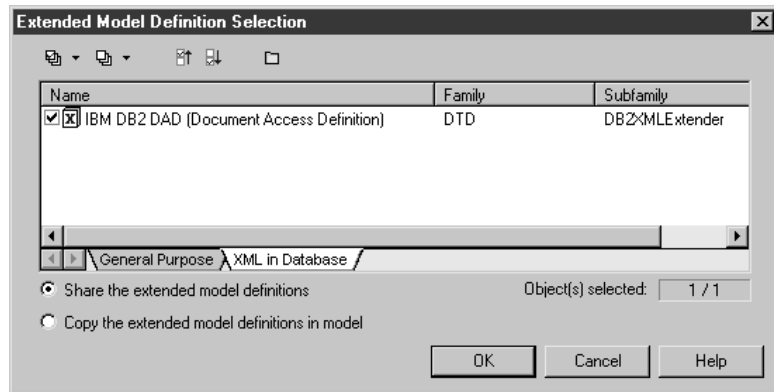
The List of Extended Model Definitions appears.



- 10 Click the Import an Extended Model Definition tool.

The Extended Model Definition Selection dialog box appears.

- 11 In the XML in Database page, select IBM DB2 DAD.



Note: In the General Purpose page, you can select the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema.

- 12 Click OK in the Extended Model Definition Selection dialog box.
IBM DB2 DAD appears in the List of Extended Model Definitions.
- 13 Click OK in the List of Extended Model Definitions.

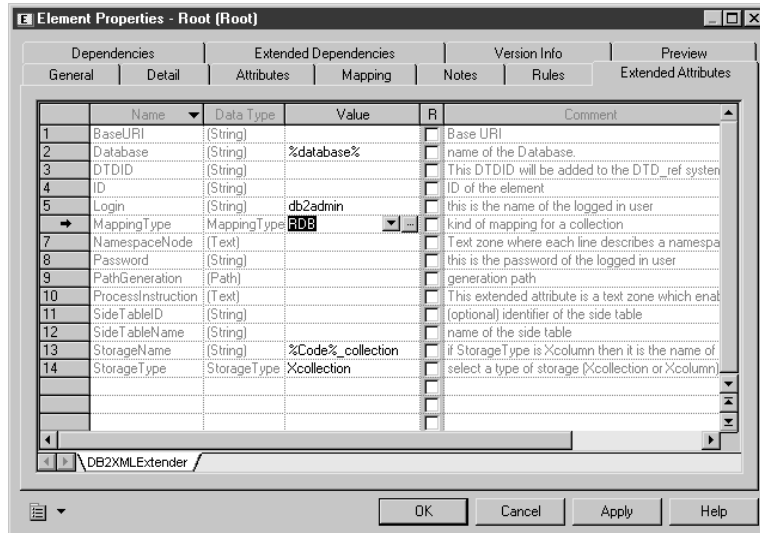
The IBM DB2 DAD extended model definition appears in the Browser tree view, attached to the generated XML model.

DAD file preview

In the Preview page of the Root element property sheet, click the **DB2XMLExtender.DAD File** tab to preview the DAD file.
If the DAD File tab is not available, click the Select Generation Targets tool to select IBM DB2 DAD in the Targets list and click OK.

- 14 <Optional> <Defining the type of DAD file> Double-click a global element in the diagram to display its property sheet.

- In the Extended Attributes page, click the **DB2XMLExtender** tab and select a value for the **StorageType** (Xcollection or Xcolumn). In the case of an Xcollection, select a value for the **MappingType** (RDB or SQL).



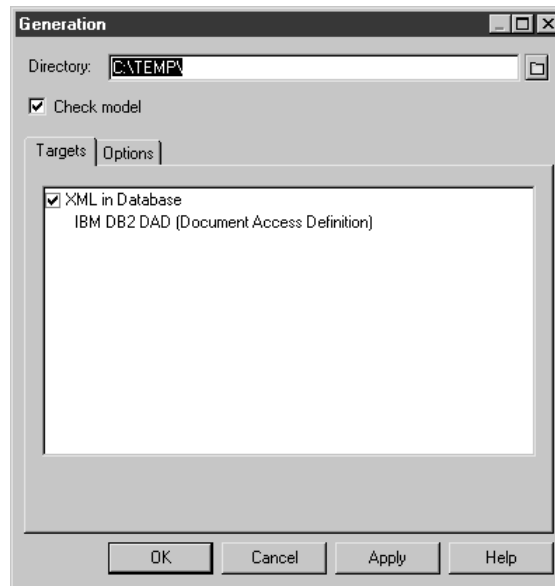
You can define a value for the following extended attributes:

Extended attribute	Description
Database	Name of the database
DTDID	ID added to the DTD_ref system table in DB2 XML Extender
Login	Name of the logged-in user
MappingType	Type of mapping for a collection
NamespaceNode	Text zone where each line describes a namespace couple (name = value). The separator character is '='
Password	Password of the logged-in user
PathGeneration	Generation path
ProcessInstruction	A text zone that enables the user to enter some instruction
SideTableID	Identifier of the side table (optional)
SideTableName	Name of the side table

Extended attribute	Description
StorageName	If StorageType is Xcolumn, then it is the name of the sidetable column
StorageType	Type of storage (Xcollection or Xcolumn)

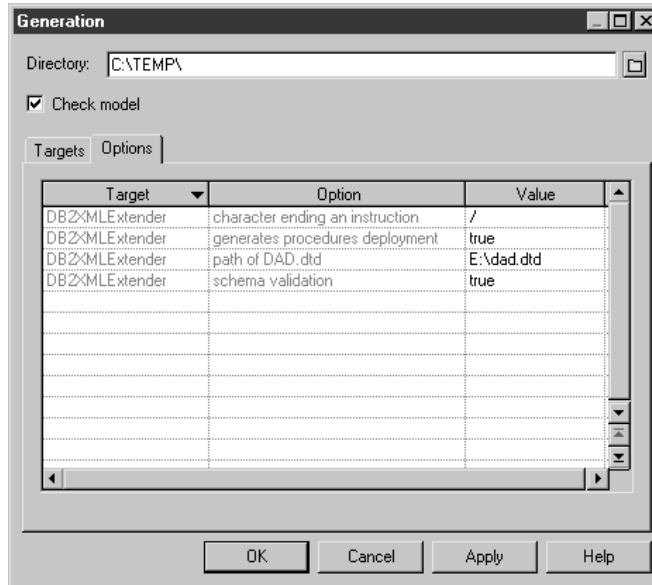
- 16 Click OK.
- 17 Repeat steps 14 to 16 for each global element you want to determine the type of DAD file or other extended attributes.
- 18 Select Language→Generate Document Type Definition File.

The Generation dialog box appears with IBM DB2 DAD selected in the Targets page.



Note: The DAD files will be generated simultaneously with the DTD file and a SQL file for stored procedures.

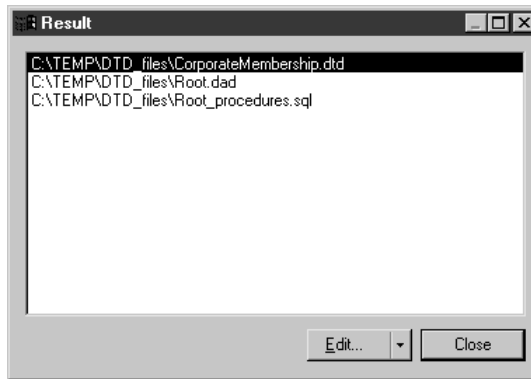
- 19 <optional> In the Options page, generation options are set by default. You can change their value.



Option	Description
Character ending an instruction	Character ending instructions in the SQL file for stored procedures
Generates procedures deployment	Generation of a SQL script for stored procedures enabling XML data storage and facilitating XML data retrieval
Path of DAD.dtd	Path of the DTD file installed with IBM DB2 Extender and describing the specific syntax of DAD files
Schema validation	Validation tag in the DAD files to check the conformity of DAD files with the DAD syntax

- 20 Click the **Select a Path** button, beside the Directory box, to select a path for the DAD, DTD and SQL files.
- 21 Click OK.

The Result dialog box appears with the paths of the generated DAD, DTD and SQL files.



22 Select the path of a DAD file and click Edit.

The selected DAD file appears in the Editor window.

◆ Extract of a DAD file defined with **Xcollection** as StorageType, and **RDB** as MappingType:

```
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Root SYSTEM "C:\TEMP\DTD_files\CorporateMembership.dtd"</doctype>
<root_node>
<element_node name="Root">
  <element_node name="DEPARTMENT">
    <attribute_node name="DEPNUM">
      <RDB_node>
        <table name="DEPARTMENT"/>
        <column name="DEPNUM" type="INTEGER"/>
      </RDB_node>
    </attribute_node>
    <attribute_node name="DEPNAME">
      <RDB_node>
        <table name="DEPARTMENT"/>
        <column name="DEPNAME" type="VARCHAR(254)"/>
      </RDB_node>
    </attribute_node>
  <element_node name="EMPLOYEE">
    <attribute_node name="DEPNUM">
      <RDB_node>
        <table name="EMPLOYEE"/>
        <column name="DEPNUM" type="INTEGER"/>
      </RDB_node>
    </attribute_node>
  </element_node>
</element_node>
</root_node>
</xcollection>
</DAD>
```

◆ DAD file defined with **Xcolumn** as StorageType:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<xcolumn>
<table name="DEPARTMENT" key="DEPNUM" orderBy="DEPNUM, , DEPNAME">
  <column name="DEPNUM"
    type="INTEGER"
    path="/DEPARTMENT/@DEPNUM"
    multi_occurrence="no"/>
  <column name="DEPNAME"
    type="VARCHAR(254)"
    path="/DEPARTMENT/@DEPNAME"
    multi_occurrence="no"/>
</table>
<table name="EMPLOYEE" key="EMPID" orderBy="EMPID, DEPNUM, FIRSTNAME, LASTNAME">
  <column name="DEPNUM"
    type="INTEGER"
    path="/DEPARTMENT/EMPLOYEE/@DEPNUM"
    multi_occurrence="no"/>
  <column name="EMPID"
    type="INTEGER"
    path="/DEPARTMENT/EMPLOYEE/@EMPID"
    multi_occurrence="no"/>
  <column name="FIRSTNAME"
    type="CHAR(1)"
    path="/DEPARTMENT/EMPLOYEE/@FIRSTNAME"
    multi_occurrence="no"/>
  <column name="LASTNAME"
    type="CHAR(1)"
    path="/DEPARTMENT/EMPLOYEE/@LASTNAME"
    multi_occurrence="no"/>
</table>
</xcolumn>
</DAD>
```

23 Repeat step 22 to edit another DAD file.

or

Click Close in the Result dialog box.

Generating SQL/XML queries

SQL/XML is an XML extension of the Structured Query Language. With SQL/XML, you retrieve relational data using extended SQL syntax, and produce a result using XML. SQL/XML is made of five main functions:

- ◆ **XMLELEMENT**: to edit an element with a name, a list of attributes (optional) and a list of values (optional)
- ◆ **XMLATTRIBUTES**: to edit a list of attributes with names and values
- ◆ **XMLAGG**: to edit in multiple rows a concatenation of elements, from a single XML value corresponding to a single column
- ◆ **XMLCONCAT**: to edit in the same row a concatenation of elements, from several XML values corresponding to several columns
- ◆ **XMLFOREST**: to edit in the same row a concatenation of elements, from several SQL values corresponding to several columns. The name and value of a column become the name and value of an element

An XML model allows you to generate SQL/XML queries for **global elements**, whatever the targeted XML language (XSD, DTD or XDR). You need first to map an XML model to a PDM, then to attach the SQL/XML extended model definition to the mapped XML model.

The best way to map an XML model to a PDM is to use the **XML Builder Wizard** from a PDM. The generated XML model is automatically mapped to the PDM and linked with the SQL/XML extended model definition. If need be, you can still modify the mapping through the Mapping page of elements and complex types property sheets.

↪ For more information on the XML Builder Wizard, see Generating an XML model via the XML Builder Wizard, in chapter Generating from a PDM, in the *PDM User's Guide*.

Generated SQL/XML queries cannot be parameterized.

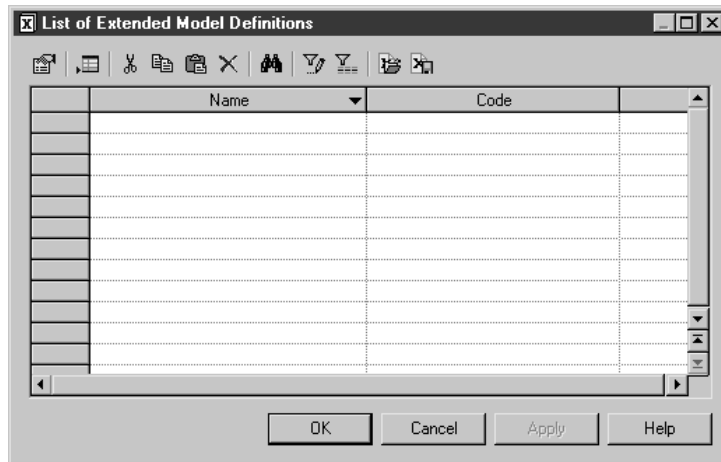
Caution

The following procedure assumes you have an XML model open in the workspace and mapped to a PDM.

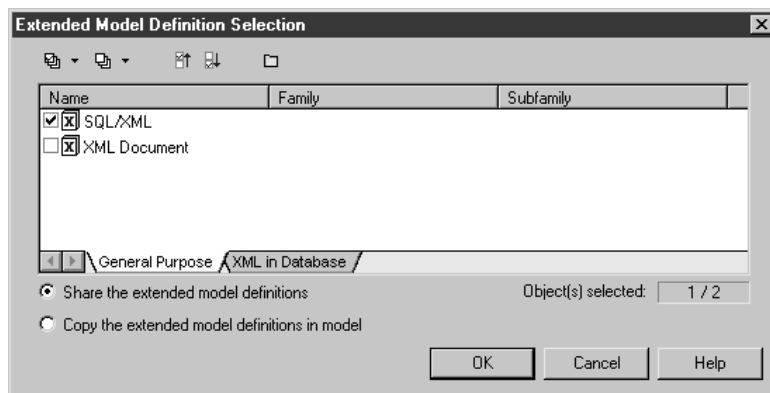
❖ **To generate SQL/XML queries:**

- 1 < If the SQL/XML extended model definition is already attached to the XML model, go to step 6 > Select Model→Extended Model Definitions.

The List of Extended Model Definitions appears.

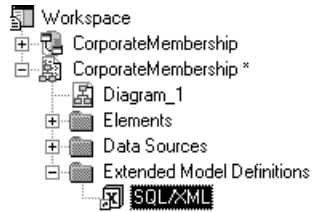


- 2 Click the Import an Extended Model Definition button.
The Extended Model Definition Selection dialog box appears.
- 3 In the General Purpose page, select SQL/XML.

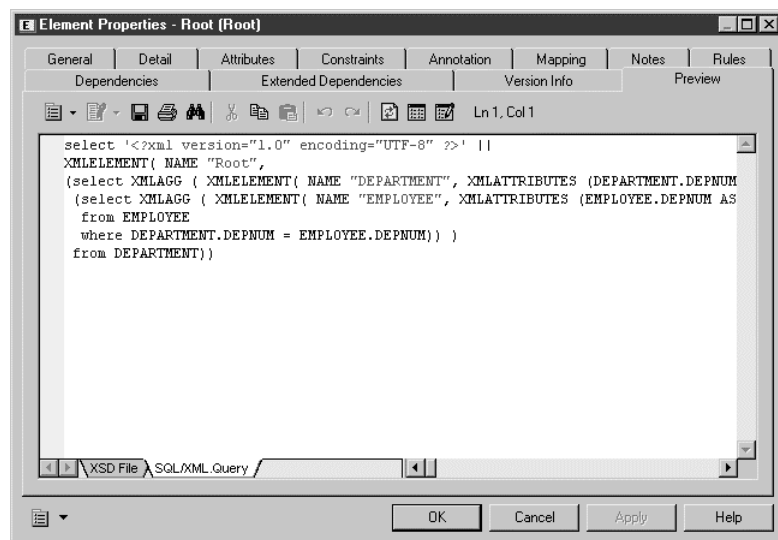


- 4 Click OK.
SQL/XML appears in the List of Extended Model Definitions.
- 5 Click OK.

The SQL/XML extended model definition appears in the Browser tree view, attached to the model.



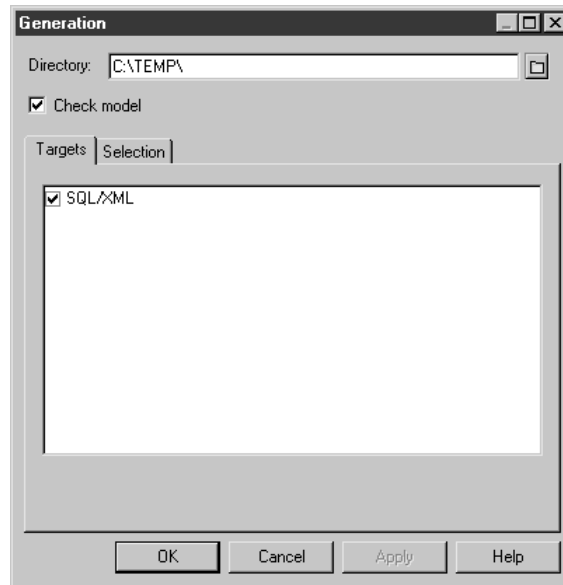
- 6 < optional > Double-click a **global element** in the diagram to display its property sheet.
- 7 In the Preview page, click the SQL/XML.Query tab to display the SQL/XML query.



The Preview page is in read-only mode, you cannot modify the script.

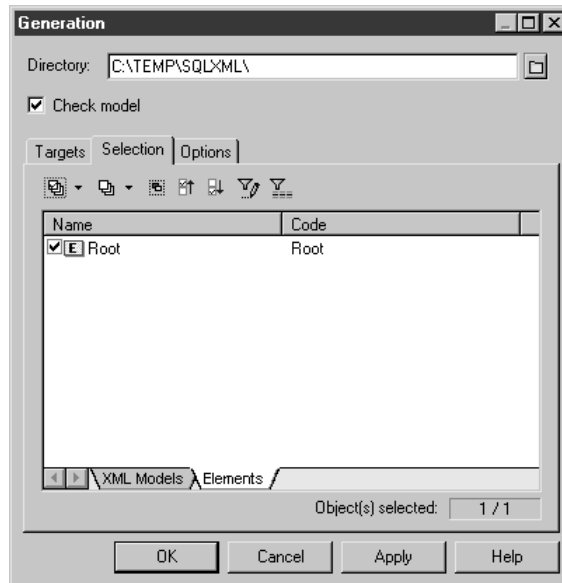
- 8 Click OK.
- 9 Select Tools→Extended Generation.

The Generation dialog box appears with SQL/XML selected in the Targets page.



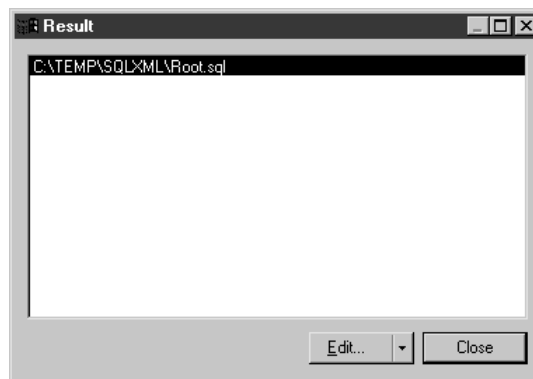
- 10 Click the **Select a Path** button beside the Directory box to select a path for the SQL/XML queries.

- 11 In the Selection page, select the global elements for which you want to generate a SQL/XML query. (Only one global element in the example)



- 12 Click OK.

The Result dialog box appears with a path for each SQL/XML query. (Only one in the example)



- 13 Select the path of a SQL/XML query and click Edit.

The SQL/XML query appears in the Editor window. (Extract)

```
select '<?xml version="1.0" encoding="UTF-8" ?>' ||
XMLLEMENT( NAME "Root",
(select XMLAGG ( XMLLEMENT( NAME "DEPARTMENT", XMLATTRIBUTES (DEPARTMENT.DEPNUM
(select XMLAGG ( XMLLEMENT( NAME "EMPLOYEE", XMLATTRIBUTES (EMPLOYEE.DEPNUM AS
  from EMPLOYEE
  where DEPARTMENT.DEPNUM = EMPLOYEE.DEPNUM)) )
from DEPARTMENT))
```

14 Repeat step 13 to edit another SQL/XML query.

or

Click Close in the Result dialog box.

Once generated, SQL/XML queries are processed by SQL interpreters in databases supporting XML.

CHAPTER 6

Generating from an XML model

About this chapter This chapter describes how to generate an XML model from an XML model.

Contents

Topic	Page
Generation basics	256
Generating an XML model from an XML model	259

Generation basics

You can generate an XML model from an XML model.

There are two options to generate an XML model from an XML model:

Generation option	Description
Generate new XML model	It creates a copy of the source XML model, converting the source language (DTD, XSD or XDR) into the target language
Update existing XML model	It creates a default model with the objects translated from the XML model that is merged with an existing XML model. You can update, delete or add objects in the existing XML model (right pane) based on modifications made in the default model (left pane)

Caution

The option Update existing XML model only works with an XML model as source model.

You can only merge models of the same type.

🔗 For more information on merging two models, see chapter Comparing and Merging Models in the *General Features Guide*.

Target models parameters

An external shortcut depends on a target object in a different model. External shortcuts allow you to share objects between different models.

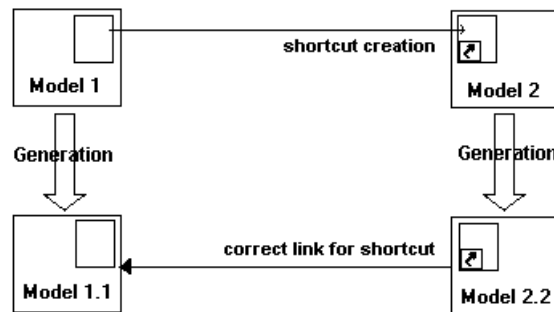
When you generate models into other models, you can preserve the link between an external shortcut and its target model through generation.

You can use the Target Models tab to select generated models to retrieve the corresponding target objects of external shortcuts. External shortcuts can then be correctly linked to target objects in the correct target model.

The Target Models page displays the following columns:

Column	Description
Target Models	Original target model of the shortcut (not editable)
Generated Models	It lets you select the model that will be used as the target for the generated shortcut

The model generation process allows you to define the target object of a shortcut in a generated model.



For example, here is the proper sequence of events for external shortcuts generation:

- ◆ Model 1 is the target model of a shortcut in Model 2
- ◆ Model 1 is generated to Model 1.1
- ◆ Model 2 is prepared for generation to Model 2.2 by associating appropriate parameters in the Target Models page:

If Model 1 is still opened in the workspace The Target Models column displays the original target model (Model 1), and its path. The Generated Models column displays the last generated model the first time you generate Model 1; the next time you generate Model 1, the Generated Models column displays the last model selected. You can click the arrow in the Generated Models column to modify the model selection in order to allow the creation of a correctly linked shortcut.

If Model 1 is closed in the workspace The Target Models column displays the original target model (Model 1), and its path. The Generated Models column displays <none>. When you click into the Generated Models column, the original target model Model 1 is automatically opened in the workspace in order to find the models generated from Model 1. You can use the arrow to select Model 1.1, the new target that will allow the creation of a correctly linked shortcut in Model 2.2.

- ◆ The external shortcut in Model 2.2 is correctly generated with a link to its target object in Model 1.1.

🔗 For more information on shortcuts, see chapter Managing Shortcuts in the *General Features Guide*.

You can also preserve the link between an external replication and its target model through generation.

↪ For more information on the generation of object replications, see section *Generating replications*, in chapter *Managing Object Replications*, in the *General Features Guide*.

Generating an XML model from an XML model

This section explains how to generate an XML model from an XML model.

Why generate an XML model from an XML model?

You can generate an XML model from an XML model when you need to keep two models synchronized during the design process.

This kind of generation allows you to create a copy of a given model and define generation links between objects in the source XML model and their equivalent in the generated XML model. When changes are made to the source model, they can then be easily propagated to the generated models using the Update Existing Model generation mode.

The generated model is the one that usually contains more information.

Generating and updating an XML model

The General page of the XML Model Generation Options dialog box displays the following options:

- ◆ Generate new XML model
- ◆ Update existing XML model

Generate new XML model

You must indicate the following parameters when you generate a new XML model:

Parameter	Description
XML language	Target XML language
Share	XML language for the resulting XML model. It uses a shared XML language stored in the XML Languages library
Copy	XML language for the resulting XML model. It uses a copy of the XML language stored in the XML Languages library
Name	File name for the resulting XML model
Code	Reference code for the resulting XML model
Configure Model Options	It lets you define the model options for the new XML model

Update existing XML model

☞ For more information on model options when generating an XML model from an XML model, see section Defining model options in chapter XML Model Basics.

You can generate an XML model into an existing XML model.

In the Update Existing Model groupbox, the Select Model dropdown listbox contains all the models already generated from the current model. You can use the tooltip in the dropdown listbox to verify the workspace location and physical path of the generated models.

When the current model has never been generated or when the generation link is broken, the Select Model dropdown listbox is empty. You can click the Ellipsis button beside the Select Model dropdown listbox to display a Select Model dialog box. This dialog box lets you select a model of the correct type to update among models available in the workspace (either open or closed). You can use the Workspace Location and Physical Path columns in the Select Model dialog to select the model to update.

To update an existing XML model, you must indicate the following parameters:

Parameter	Description
Select model	Existing model already generated from the current model. Current model and selected generated model are merged to create an updated model. The dropdown listbox displays all the models already generated from the current model. The Ellipsis button lets you select models available in the workspace (either open or closed)
Preserve modifications	Allows a comparison and merge of the newly generated XML model (default XML model) with the currently selected XML model

Clearing the Preserve modifications check box

If the Preserve modifications check box is not selected, PowerDesigner automatically replaces the existing XML model with the newly generated XML model. If you want to choose which objects to add or delete from the existing XML model, you must select Preserve modifications to compare and merge the two XML models.

Defining generation options

The Detail page of the XML Model Generation Options dialog box displays the following options:

Option	Description
Check model	When selected, verifies the model before generating the XML model, and stops generation if an error is found
Save generation dependencies	When selected, PowerDesigner keeps track of the identity of the origin of each generated object. It is useful when merging two XML models which have been generated from the same XML model. Objects can be compared and recognized as the same object, even if the object has been modified in the merged XML model. If not selected, origin objects have no link with generated objects
Enable transformations	This button is used to activate transformations during generation. When you click this button, the Pre-generation tab appears if the source model contains transformations. You can select the transformations to execute before generation. The Extended Model Definitions tab also appears for you to select extended model definition files to attach to the generated model. These files may contain post-generation transformations, in this case, the Post-Generation tab appears to let you select the transformations you want to be executed in the generated model. If the generation is an update, and the generated model contains extended model definitions with post-generation transformations, the Post-generation tab automatically appears as soon as you click the Enable Transformations button

Check model before generation

If you select the Check model option, the procedure to generate an XML model starts by checking the validity of the XML model. An XML model results when no errors are found. You can set check options by selecting Tools→Check Model.

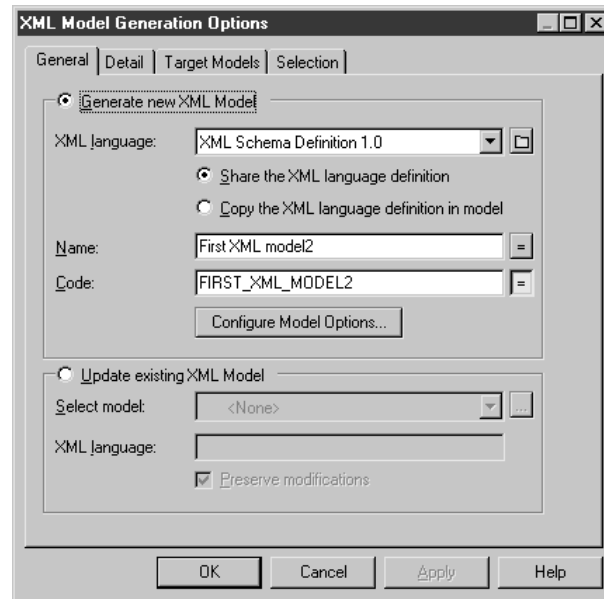
Generating a new XML model from an XML model

You can generate an XML model from an XML model. PowerDesigner creates a new XML model containing all the objects that you selected to generate in the XML model. The newly created XML model appears in the browser and the corresponding diagram opens in the work area.

You can only generate an XML model from the active XML model diagram.

❖ **To generate a new XML model from an XML model:**

- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.
- 2 Select the Generate new XML Model radio button.



- 3 Select an XML language in the XML language dropdown listbox.
- 4 Select Share or Copy radio button.
- 5 Type a name and a code for the new XML model, otherwise it will have the same name and code as the current model.
- 6 <optional> Click Configure Model Options to define the options of the generated model.
- 7 Click the Detail tab to define options and generation parameters.
- 8 Click the Target Models tab to select the target models of shortcuts in the current model.
- 9 Click the Selection tab to display the Selection page.
- 10 Select the name of an XML model from the Select Location dropdown listbox.

- 11 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.
- 12 Click OK.

The Output window shows the progress of the generation process. The diagram of the new XML model appears in the work area.

Updating an existing XML model

There are two ways to update an existing XML model, depending on whether the Preserve modifications option is selected or not:

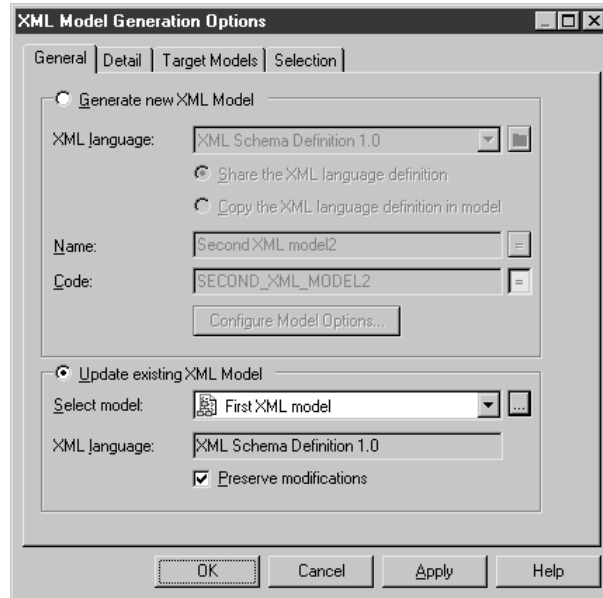
Preserve modifications	Result
Selected	You can manually compare and merge an existing XML model (right pane) with the newly generated XML model (left pane)
Not selected	The existing XML model is automatically replaced by the newly generated XML model

You can only generate an XML model from the active XML model diagram window.

❖ To update an existing XML model by generating from an XML model:

- 1 Select Tools→Generate XML Model to display the XML Model Generation Options dialog box.
- 2 Select the Update existing XML Model radio button.

- 3 Select a model from the Select model dropdown listbox, if the current model has already been generated.
or
Click the Ellipsis button, beside the Select model dropdown listbox, and select a model model available in the workspace in the Select a model dialog box.



Preserve modifications

Select the Preserve modifications check box if you want to preserve objects in the XML model.

If you clear this check box, all existing objects in the XML model will be removed from the model, leaving only the objects generated from the XML model.

- 4 Click the Detail tab to define options and generation parameters.
- 5 Click the Target Models tab to select the target models of shortcuts in the current model.
- 6 Click the Selection tab to display the Selection page.
- 7 Select the name of an XML model from the Select Location dropdown listbox. The default XML model is generated from this XML model.
- 8 Select the check boxes for the objects you want to generate, and clear the check boxes for the objects you do not want to generate.

- 9 Click OK.

If you selected the Preserve modifications check box, the Merge Models window appears.

If you cleared the Preserve modifications check box, the updated XML model diagram appears in the work area.

Merging models


The Merge Models dialog box shows the newly generated XML model in the left pane, and the existing XML model in the right pane. You can select or clear object check boxes in the right pane for objects that you want to include or delete in the model to be merged.


🔗 For more information on merging models, see chapter Comparing and Merging Models in the *General Features Guide*.

XML Model Glossary

all	A group particle indicating that child elements can appear in any order, each of them once or not
annotation	Additional information about a model and its objects proceeding from documentation and/or applications
any	Allows you to attach any type of object to a choice or a sequence group particle
any attribute	Allows you to insert any attribute of specified namespaces into an element, a complex type or an attribute group
attribute	Additional information about an element or a complex type
attribute group	A set of attributes that can be incorporated into an element, a complex type or another attribute group, with only the reference of its name
business rule	A written statement specifying what an XML model must contain or how it must be structured to support business needs
character data	In a schema or a DTD, all text that is not markup and that will not be parsed
choice	A group particle indicating that only one child element can be linked to its parent element
complex type	A data type definition to determine the attributes and child elements of a parent element
complex content	Allows you to extend or restrict the values of a complex type with mixed content or elements only
derivation	Used to extend or restrict the values of simple types and complex types
DTD	Document Type Definition. An XML language defining the content of an XML file with a list of legal elements
element	The basic building block of an XML model
embedded type	Locally defined data type. For an object encapsulated into another object
entity	Enables you to include predefined values, external XML or non-XML files (URL values) in an XML model defined with a DTD

extension	Derivation of a complex type to extend the values of its base type
facet	A constraint on the set of values of a simple type
field	An XPath expression that specifies the value (or one of the values) used to define an identity constraint (unique, key or keyRef)
global object	A global object has a global scope. It has no parent object and can be reused locally in the model through referenced objects. In a schema, it is directly linked to the <schema> root element
group	A set of elements arranged by a group particle (all, choice or sequence)
group particle	An indication (all, choice, sequence) on how child elements are related with their parent element
ID	Identifier (key word) for a model object. It must be unique within the model. To be XML-valid, it must start by an underscore or a letter, followed by alphanumeric characters
identity constraint	An indication to specify that element or attribute values must be unique within their specified scopes
import	Identifies a namespace whose schema components are referenced by the current schema
include	Allows you to include a specified schema file in the target namespace of the current schema
key	An identity constraint to specify that an element or attribute value (or set of values) must be a key within a specified scope. A key means that data should be unique, not null and always present within a specified scope
keyRef	An identity constraint to specify that an element or attribute value (or set of values) corresponds to those of a specified key or unique constraint
list	Derivation of a simple type to restrict its values to a list of values of a specified data type
local object	A local object has a local scope, within a parent object. It cannot be reused elsewhere in the model
namespace	Original location for the definition of an object. It should be a URI
no-colon name (NCName)	A name without a colon (:), beginning with a letter or an underscore character, and followed by any combination of characters
notation	Allows you to describe the format of non-XML data within an XML model

qualified name (QName)	A name beginning with a prefix and a colon (:). The prefix is associated with a namespace URI. For example: xs:schema, xs being associated with http://www.w3.org/2001/XMLSchema
RDB	Relational database. A database organized in terms of relations between data
redefine	Allows you to redefine simple and complex types, groups and attribute groups from an external schema file in the current schema
reference	A property to define a local object by reference to a global object. The referenced (local) object has the same properties as the reference (global) object
restriction	Derivation of a simple type or a complex type to restrict the values of their base type
reverse-engineering	Process of examining and recovering data or source code from a file that is then used to build or update an XML model
schema	Diminutive for XML Schema or XML Schema Definition (XSD). It defines the structure and data types of an XML document
scope	Sphere of action of an object. It can be local (within a parent object) or global (within a model)
selector	An XPath expression that selects a set of elements for an identity constraint (unique, key or keyRef)  For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model
sequence	A group particle indicating that child elements must appear at least once in the order of their declaration
simple content	Allows you to extend or restrict the values of a complex type supporting character data or a simple type
simple type	A data type definition for the value of elements or attributes with text-only content
SQLX	XML extension of the Structured Query Language
stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
target namespace	Namespace of all the schema elements in the model. Its name is a URI which does not refer to any file but only to an assigned name. A prefix can be assigned to the namespace. All the schema elements with this prefix in their start-tag will be associated with the namespace
type	Data type. It can be predefined or user-defined, simple or complex.

union	Derivation of a simple type to restrict its values to a collection of built-in and simple data types
unique	An identity constraint used to specify that an element or attribute value (or set of values) must be unique or null within a specified scope
URI	Uniform Resource Identifier. A string of characters which identifies an Internet resource
XDR	XML-Data Reduced. An XML language defining the structure of an XML file. It is a simplified XSD (or schema)
XML	Extensible Markup Language. Used for structuring documents with self-describing tags in plain text format
XML language	Language used to define the structure of an XML model. It can be Document Type Definition 1.0 or XML Schema Definition 1.0
XPath	XML Path Language. A language to address parts of an XML document  For more information on XPath, see XPath abbreviated syntax in section Defining an identity constraint selector, in chapter Building an XML model
XSD	XML Schema Definition (or schema). An XML language defining the structure of an XML file. It supports namespaces and datatypes

Index

A

- all 53
- annotated schema
 - for Microsoft SQL Server 2000 217
 - for Oracle 9i2 230
- annotation 115
 - application information 115
 - check 165
 - create annotation 118
 - documentation 115
 - global 115
 - local 115
- any 58
 - namespace 58
 - process contents 58
- any attribute 61
 - namespace 61
 - process contents 61
- application information 115
- arrange symbols 34
- attribute 46
 - add attribute to element 51
 - attribute type (XDR) 48
 - AttributeType (XDR) 48
 - check 156
 - data type 48
 - default 50
 - default (XDR) 49
 - detail properties 50
 - embedded type 48
 - fixed 50
 - form 50
 - general properties 48
 - global 48
 - local 48
 - reference 48
 - stereotype 48
 - type (XDR) 49
 - use 50
 - values (DTD, XDR) 50

- attribute group 82
 - check 157
 - create attribute group 85
 - properties 82
 - reference 82
 - stereotype 82
- attribute mapping 175
 - mapped to 175
 - sources 175
- AttributeType (XDR)
 - default 49
 - dt type 49
 - dt values 49
 - name 49
- automatic correction 140, 145

B

- base type 102
 - complex type restriction 111
 - simple type restriction 104
- bibliography ix
- block 19, 40, 94
- business rule (XSM) 132
 - apply business rule 135
 - check 149
 - client 133
 - constraint 132
 - create business rule 133
 - definition 132
 - fact 132
 - formula 132
 - list 133
 - requirement 132
 - server 133
 - stereotype 132
 - type 132
 - validation 132

C

- check model 140
 - annotation 165
 - attribute 156
 - attribute group 157
 - automatic correction 140, 145
 - business rule (XSM) 149
 - complex type 152
 - correct error 145
 - data source 149
 - display details 145
 - element 153
 - entity 151
 - error list 145
 - extended link 162
 - extended object 162
 - extension 163
 - file 150
 - group 155
 - group particle 147
 - import 158
 - include 152
 - key 159
 - keyRef 160
 - levels of problem severity 140
 - manual correction 145
 - model 148
 - namespaces 148
 - notation 156
 - object selection 141
 - objects 147
 - re-check 145
 - redefine 158
 - replication 163
 - restriction 163
 - shortcut 148
 - simple type 152
 - simple type list 164
 - simple type union 165
 - target namespace 148
 - unique 161
- child element 52
- choice 53
- collapse 34
- collapse tab 55
- compare XML models 179
- complex content 101
- complex type 92
 - attributes properties 94

- complex type (*continued*)
 - block 94
 - check 152
 - complex content 101
 - content 93
 - create complex type 96
 - detail properties 94
 - final 94
 - general properties 93
 - global 92
 - local 92
 - mapping properties 95
 - mixed 94
 - simple content 100
 - stereotype 93
- constraint 64
- content 37, 93
- context node 70
- convert local to global object 177
- convert unique references 208

D

- DAD file 237
- data source 166
 - check 149
 - create data source 166
 - model type 166
 - models 166
- data type
 - attribute 48
 - attribute check 156
 - complex type 92
 - complex type display preference 96
 - complex type restriction 111
 - element 37
 - element check 153
 - extension 102
 - external shortcut 87
 - simple type 89
 - simple type list 112
 - simple type list check 164
 - simple type restriction 104
 - simple type union 113
 - simple type union check 165
- database
 - DAD file 237
 - IBM DB2 237
 - Microsoft SQL Server 2000 217

- database (*continued*)
 - Oracle 9i2 230
 - SQL/XML queries 216, 249
 - XML in database 216
- default 40, 50
- derivation 102
 - extension 102
 - restriction 104
 - simple type list 112
 - simple type union 113
- diagram 32
- display preference
 - any 61
 - complex type 96
 - element 52
 - group 82
 - simple type 90
- documentation 115
- DTD 3

E

- element 35
 - attributes properties 41
 - block 40
 - check 153
 - child element 52
 - constraints properties 42
 - content 37
 - create element 43
 - data type 37
 - default 40
 - detail properties 40
 - element type (XDR) 37
 - ElementType (XDR) 37
 - embedded type 37
 - final 40
 - fixed 40
 - form 40
 - general properties 37
 - global 37
 - group 76
 - local 37
 - mapping properties 42
 - maxOccurs (XDR) 39
 - minOccurs (XDR) 39
 - parent element 52
 - reference 37
 - stereotype 37

- element (*continued*)
 - substitution group 40
 - type (XDR) 39
 - values (XDR) 50
- ElementType (XDR)
 - content 39
 - dt type 39
 - dt values 39
 - model 39
 - order 39
- embedded type 37, 48, 104, 112
- enable transformations 182, 190, 261
- entity 122
 - check 151
 - create entity 123
 - properties 122
 - stereotype 122
 - value 122
- enumeration 104
- error 140
- error list 145
- expand 34
- expand all 34
- expand tab 55
- extended
 - dependencies 15
 - link check 162
 - model definition 13
 - object check 162
- extension 102
- extension check 163
- external schema 21
- external shortcut 256

F

- facet 105
 - enumeration 105
 - fixed 105
 - fraction digits 105
 - length 105
 - maximum exclusive 105
 - maximum inclusive 105
 - maximum length 105
 - minimum exclusive 105
 - minimum inclusive 105
 - minimum length 105
 - pattern 105
 - total digits 105

- facet (*continued*)
 - whitespace 105
- field 64, 66, 67, 73
 - stereotype 73
 - XPath 73
- file check 150
- final 19, 40, 90, 94
- fixed 40, 50, 105
- form 40, 50
- fraction digits 104
- functional overview 2

G

- generate 256
 - basics 256
 - check model 182
 - check model 261
 - check model parameter 203
 - DTD file from XML model 202
 - enable transformations 182, 190, 261
 - external shortcut 256
 - model options 261
 - model options from OOM 190
 - model options from PDM 182
 - new XML model from OOM 192
 - new XML model from PDM 184
 - new XML model from XML model 261
 - OOM to XML model 188
 - options parameter 203
 - PDM to XML model 180
 - preserve modifications 181, 185, 189, 193, 259, 263
 - target model 256
 - target object 256
 - targets parameter 203
 - tasks parameter 203
 - updated XML model from OOM 193
 - updated XML model from PDM 185
 - updated XML model from XML model 263
 - XDR file from XML model 202
 - XML language 181, 189, 259
 - XML model to XML model 259
 - XSD file from XML model 202
- global objects 176
- group 53, 76
 - check 155
 - create group 79
 - properties 76

- group (*continued*)
 - reference 76
 - stereotype 76
- group particle 53
 - all 53
 - check 147
 - choice 53
 - create group particle 55
 - properties 54
 - sequence 53
- group symbols 34

I

- IBM DB2 237
- identity constraint 64
 - create identity constraint 69
 - field 73
 - key 66
 - keyRef 67
 - selector 70
 - unique 64
- import 125
 - check 158
 - create import 126
 - namespace 125
 - schema location 125
- include 127
 - check 152
 - create include 128
 - schema location 127

K

- key 42, 66
 - check 159
 - field 66
 - properties 66
 - selector 66
 - stereotype 66
- keyRef 42, 67
 - check 160
 - field 67
 - properties 67
 - reference 67
 - selector 67
 - stereotype 67

L

- length 104
- level of problem severity
 - error 140
 - warning 140
- link 8, 45, 57, 81, 98
 - child object to complex type 98
 - child object to element 45
 - child object to group 81
 - child object to group particle 57
 - child object to parent object 8
- local objects 176

M

- manipulate XML objects graphically 176
- map object 166
 - attributes mapping 169, 170, 171, 172
 - complex type sources 170, 172
 - create mapping for XML object 172
 - data source 166
 - element sources 169, 171
 - XML Model to OOM 171
 - XML Model to PDM 169
- mapping
 - attribute 175
 - attribute sources 175
 - mapped to 175
- maximum exclusive 104
- maximum inclusive 104
- maximum length 104
- member types 113
- merge XML models 179
- Microsoft SQL Server 2000
 - annotated schema through mapping 217
 - reinforcing mapping with extended attributes 225
- minimum exclusive 104
- minimum inclusive 104
- minimum length 104
- mixed 94, 101
- model 3
 - attribute form 19
 - block 19
 - check 148
 - check XML model 142
 - compare and merge XML models 179
 - create XML model 24
 - detach model from workspace 28

model (continued)

- detail properties 19
- element form 19
- environment 10
- extended definitions 13
- extended dependencies 15
- external schemas properties 21
- final 19
- general properties 17
- items properties 20
- namespaces properties 22
- naming conventions 14
- open existing model 28
- options 14
- preview properties 24
- save and close 29

N

- namespace 22, 58, 61, 125
- node
 - context node 70
 - root node 70
- notation 120
 - check 156
 - create notation 121
 - properties 120
 - public 120
 - stereotype 120
 - system 120

O

- Oracle 9i2 230

P

- parent element 52
- pattern 104
- preserve modifications 181, 185, 189, 193
- process contents 58, 61
- public 120

R

- RDB 216
- re-check model 145

- redefine 128
 - check 158
 - create redefine 130
 - schema location 128
- reference 37, 48, 67, 76, 82
- replication check 163
- report 196
 - create model report 196
 - underline hierarchical structure 198
- restriction 104
 - check 163
 - create restriction on complex type 111
 - create restriction on simple type 108
 - enumeration 104
 - fraction digits 104
 - length 104
 - maximum exclusive 104
 - maximum inclusive 104
 - maximum length 104
 - minimum exclusive 104
 - minimum inclusive 104
 - minimum length 104
 - pattern 104
 - total digits 104
 - whitespace 104
- reverse engineer
 - convert unique references to elements 208
 - expand nodes 208
 - options 208
 - show symbols 208
 - target models 208, 209
 - to existing XML model 212
 - to new XML model 209
 - XSD, DTD or XDR file to existing XML model 208
 - XSD, DTD or XDR file to new XML model 208
- root node 70

S

- schema 3
- schema location 125, 127, 128
- selector 64, 66, 67, 70
 - stereotype 70
 - XPath 70
- sequence 53
- shortcut
 - check 148

- shortcut (*continued*)
 - managing external shortcuts through references and data types 87
- simple content 100
- simple type 89
 - check 152
 - create simple type 90
 - derive by list 112
 - derive by union 113
 - derived by list 89
 - derived by restriction 89
 - derived by union 89
 - final 90
 - general properties 90
 - list check 164
 - stereotype 90
 - union check 165
- SQL/XML query (in XML model) 216, 249
- stereotype 37, 48, 64, 66, 67, 70, 73, 76, 82, 90, 93, 120, 122, 132
- substitution group 40
- system 120

T

- tab
 - collapse tab 55
 - expand tab 55
- target
 - model 256
 - namespace 19
 - object 256
- total digits 104
- typographic conventions ix

U

- union
 - member types 113
 - properties 113
- unique 42, 64
 - check 161
 - field 64
 - properties 64
 - selector 64
 - stereotype 64
- URI 22
- use 50

V

- value
 - attribute (DTD, XDR) 50
 - element (XDR) 50

W

- warning 140
- whitespace 104
- workspace 28

X

- XDR 3
 - any 58
 - attribute type 48
 - AttributeType 48
 - element type 37
 - ElementType 37
- XML 3
 - diagram 32
 - model 3
 - objects 5
- XML language 10, 259
 - change XML language 11
 - choose XML language 28
- XPath 73
 - abbreviated syntax 70
 - expressions 70
- XSD 3

