



Business Process Modeling
PowerDesigner® 16.0

Windows

DOCUMENT ID: DC38088-01-1600-01

LAST REVISED: July 2011

Copyright © 2011 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. 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.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. A ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of the respective companies with which they are associated.

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, Inc., One Sybase Drive, Dublin, CA 94568

Contents

- PART I: Building BPMs1**
- CHAPTER 1: Getting Started with Business Process Modeling3**
 - Creating a BPM5**
 - BPM Properties7
 - Previewing Process Code8**
 - Customizing your Modeling Environment10**
 - Setting Model Options10
 - Setting BPM Display Preferences11
 - Viewing and Editing the Process Language Definition
 - File11
 - Changing the Process Language12
 - Extending your Modeling Environment12
 - Linking Objects with Traceability Links13
- CHAPTER 2: Building Process Hierarchy Diagrams15**
 - Process Hierarchy Diagram Objects16**
 - Building Process Hierarchies16**
 - Building Hierarchies with the Process Tool17
 - Expanding and Collapsing Process Hierarchies18
 - Arranging, Moving, and Reusing Processes in a Hierarchy18
 - Reusing Processes19
 - Creating Default Flows Between Processes in a Business Process Diagram19

CHAPTER 3: Building Business Process Diagrams	21
.....	
Top-Level Diagram Basics	22
Top-level Diagram Objects	23
Choreography Diagram Basics	23
Allocating Responsibilities	24
Tracing the Process Choreography	24
Analyzing Data	25
Modeling the Implementation of Processes	29
Choreography Diagram Objects	30
Data Flow Diagram Basics	32
Processes (BPM)	32
Creating a Process	34
Process Properties	34
Specifying Implementation Types	37
Example: Using the Execute Operation	
Implementation Type	40
Decomposing Processes	44
Decomposing an Atomic Process	45
Creating a Decomposed Process from a	
Selection of Symbols	46
Converting a Business Process Diagram to a	
Decomposed Process	47
Removing a Level of Decomposition	48
Working with Sub-Process Diagrams	49
Working with Composite Views	49
Working with Data and Resource CRUD Matrices	50
Organization Units (BPM)	52
Creating an Organization Unit	53
Creating Organization Units with the Swimlane	
Tool	54
Organization Unit Properties	54
Attaching Processes to Organization Units	55

Displaying a Committee Process	55
Managing Swimlanes and Pools	56
Moving, Copying and Pasting Swimlanes	57
Grouping and Ungrouping Swimlanes	58
Creating Links Between Pools of Swimlanes	60
Changing the Orientation of Swimlanes	60
Resizing Swimlanes	61
Changing the Format of a Swimlane	61
Starts (BPM)	62
Creating a Start	62
Start Properties	63
Ends (BPM)	63
Creating an End	64
End Properties	64
Decisions (BPM)	65
Creating a Decision	67
Decision Properties	67
Synchronizations (BPM)	68
Creating a Synchronization	69
Synchronization Properties	69
Flows (BPM)	70
Creating a Flow	71
Flow Properties	71
Role Associations (BPM)	73
Creating a Role Association	74
Role Association Properties	74
Events (BPM)	75
Creating an Event	76
Event Properties	76
Event Handlers	77
Message Formats (BPM)	78
Creating a Message Format	79
Message Format Properties	79
Message Parts (BPM)	81
Creating a Message Part	82

- Message Part Properties82
- Data (BPM)83**
 - Creating Data84
 - Data Properties84
 - Linking Data with Other PowerDesigner Model
 - Objects85
 - Linking Data Objects to External Model Objects
.....86
 - Exporting Data to Other PowerDesigner Models
.....88
 - Importing Data from Other PowerDesigner
Models89
 - Specifying Data for a Flow, a Resource Flow or a
Message Format91
 - Migrating the Data of a Flow to a Process92
- Resources (BPM)93**
 - Creating a Resource94
 - Resource Properties94
- Resource Flows (BPM)95**
 - Creating a Resource Flow96
 - Resource Flow Properties97
- Service Providers (BPM)98**
 - Creating a Service Provider100
 - Service Provider Properties100
 - Importing a Service Provider from a WSDL File103
 - Browsing for a WSDL File on a UDDI Server104
 - Importing and Exporting Service Providers From/to
Another Model106
 - Importing a Service Provider from an OOM or a
PDM107
 - Exporting a Service Provider from a BPM108
- Service Interfaces (BPM)110**
 - Creating a Service Interface110
 - Service Interface Properties111
- Operations (BPM)111**

Creating an Operation	112
Using the Create New Operation Wizard	112
Operation Properties	115
XSD Documents	117
Creating an XSD Document	117
XSD Document Properties	118
Attaching an XML Model to an XSD Document	119
Variables (BPM)	119
Creating a Variable	120
Variable Properties	120
Correlation Keys (BPM)	121
Creating a Correlation Key	122
Correlation Key Properties	122
Data Transformations	123
Creating a Data Transformation	123
Data Transformation Properties	123
Example: Defining a Data Transformation	125
CHAPTER 4: Building Process Service Diagrams ...	127
Process Service Diagram Objects	127
CHAPTER 5: Simulating a Business Process Model with SIMUL8	129
Modeling for Simulation	131
Reviewing SIMUL8 Default Properties	134
Simulating a BPM	134
Exporting a BPM to SIMUL8	134
Analyzing Results and Fine-Tuning the Simulation	135
Synchronizing SIMUL8 Changes Back to PowerDesigner	137
Recovering a BPM from a SIMUL8 file	137
SIMUL8 Object Properties	138
SIMUL8 Work Center Properties	138
SIMUL8 Required Resource Properties	140

SIMUL8 Resource Properties140
SIMUL8 Work Entry Point Properties141
SIMUL8 Work Exit Point Properties142
SIMUL8 Route Properties143
SIMUL8 Diagram Properties144

PART II: Working with BPMs147

CHAPTER 6: Checking a BPM149

Package Checks149
Process Checks150
Decision Checks151
Synchronization Checks152
Flow Checks153
Resource Checks154
Resource Flow Checks154
Organization Unit Checks155
Start Checks156
End Checks157
Message Format Checks157
Data Checks158
Service Provider Checks159
Service Interface Checks160
Operation Checks160
Variable Checks161
Data Transformation Checks162
Correlation Key Checks163
Event Checks164
Choreography Task Checks165
Conversation Node Checks166
Communication Link Checks167

CHAPTER 7: Generating and Reverse Engineering Process Languages 169

- Generating Process Language Files from a BPM 169**
- Reverse Engineering Source Files into a BPM 171**
 - Reverse Engineering into a New BPM 171
 - Reverse Engineering into an Existing BPM 172

CHAPTER 8: Generating Other Models from a BPM 173

- Generating an Orchestration BPM 173**
- Generating an Orchestration BPM from an Analysis BPM 174**
- Generating an Orchestration BPM from a Collaborative BPM 175**
- Generating an Orchestration BPM from an Orchestration BPM 175**
- Changing Target from Analysis to Data Flow Diagram .. 176**

CHAPTER 9: Importing Visio Diagrams into PowerDesigner 179

PART III: Process Language Definition Reference ... 181

CHAPTER 10: Business Process Modeling Notation (BPMN) 183

- Conversation Diagrams (BPMN) 183**
- Choreography Diagrams (BPMN) 184**
 - Associating a Conversation Node with a Choreography Diagram or Task 186
- Collaboration and Process Diagrams (BPMN) 186**
- Participants (BPMN) 189**

Conversation Nodes (BPMN)	189
Choreography Tasks (BPMN)	190
Events (BPMN)	191
Gateways (BPMN)	194
Activities (BPMN)	195
Data (BPMN)	196
Correlation Keys and Correlation Properties (BPMN) ...	197
Messages (BPMN)	197
Flows and Links (BPMN)	197
CHAPTER 11: Data Flow Diagram (DFD)	199
Creating a Data Flow Diagram	201
Designing for Data Flow Diagram	201
Process	201
Flow	201
Data Store	202
External Entity	202
Split/merge	202
Process and Data Store Numbering	203
Data Flow Diagram Balancing	205
CHAPTER 12: Service Oriented Architecture (SOA)	
.....	207
CHAPTER 13: BPEL4WS and WS-BPEL	209
Modeling for BPEL Languages	210
Building BPEL Top-Level Diagrams	210
Building BPEL Choreography Diagrams	211
Building BPEL Messages	216
BPEL Top-Level Process Properties	217
WS-BPEL 2.0 Object Properties	218
BPEL4WS 1.1 Object Properties	221
Generating for BPEL Languages	223

Reverse Engineering BPEL Languages	224
CHAPTER 14: Sybase WorkSpace	227
Top-Down Design	228
Importing Existing Services	229
Importing EJB or Java Web Services	230
Importing BPEL Files	230
Services	231
Business Processes	231
Variables	232
Designing XSD Data Types	233
Partner Links	233
Service Invocation	234
Designing a One-Way Service Invocation	235
Designing a Request/Reply Service Invocation	235
Interface Activities	236
Designing a Receive Activity	236
Designing a Send Activity	237
Assign Activities	238
Split-Join Activities	240
Complex Activities	240
Loop Activities	241
Event Handling Activities	241
Delay and Terminate Activities	242
Sequence Flow from Activities	243
Correlations	243
Switching to Sybase WorkSpace Business Process	
Language	247
Importing WorkSpace Services	247
Invoking WorkSpace Services	248
Generating for Sybase WorkSpace Business Process ..	248
Defining Sybase WorkSpace Business Process	
Generation Parameters	249

Generating Sybase WorkSpace Business Process
Files249

CHAPTER 15: Electronic Business XML (EbXML)...251

EbXML Business Process Specification Schema

(BPSS)252
 ebXML Top-Level Diagrams and Processes255
 Designing a Business Transaction256
 Designing a Binary Collaboration261
 Designing a MultiParty Collaboration266
 Generating EbXML BPSS Files267
 Selecting EbXML Generation Options268
 Reverse Engineering EbXML BPSS269

EbXML Collaboration Protocol Agreement (CPA)269

 Designing a Partner's Identification271
 Designing CanSend/CanReceive Actions271
 Designing a Delivery Channel272
 Designing a Transport Element274
 Unsupported Concepts275
 Generating for EbXML CPA275

Index277

PART I

Building BPMs

The chapters in this part explain how to model your business processes in PowerDesigner®.

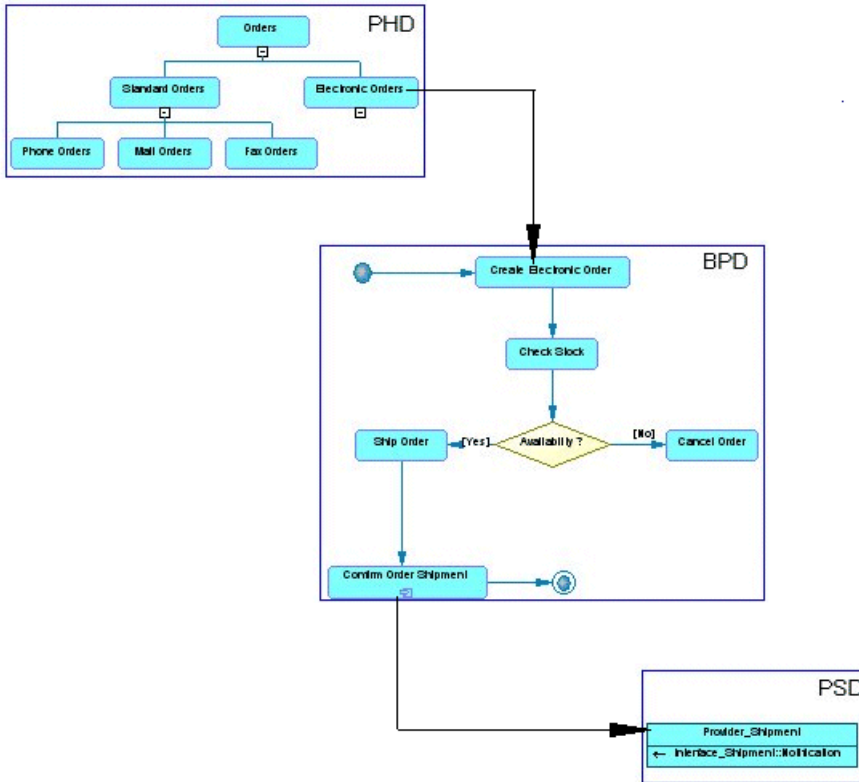
Getting Started with Business Process Modeling

A *business process model (BPM)* helps you identify, describe, and decompose business processes. You can analyze your system at various levels of detail, and focus alternatively on control flow (the sequence of execution) or data flow (the exchange of data). You can use BPEL, BPMN, and many other process languages.

The PowerDesigner® BPM allows you to analyze and design the implementation and execution of business processes using the following diagrams:

- Process hierarchy diagram (PHD) - A high-level diagram, which analyzes your business functions as a hierarchy of processes (see *Chapter 2, Building Process Hierarchy Diagrams* on page 15).
- Business process diagram - Analyzes the control flow of a process at any level of the process hierarchy. You can analyze how sub-processes will be allocated to people, organizations, or groups, the control flow of the process and how data flows through it, along with the implementation of your sub-processes. If you are using an orchestration engine, you can implement processes using one or more service providers (see *Chapter 3, Building Business Process Diagrams* on page 21).
- Process service diagram - [service orchestration languages only] Displays your service providers and any dependencies between them (see *Chapter 4, Building Process Service Diagrams* on page 127).

The diagram below shows how these diagrams can interact within your model. The process hierarchy diagram displays the processes of your system in a hierarchy. Each of these processes is analyzed in its own business process diagram, and service providers used to implement the sub-processes are displayed in a process service diagram:



The PowerDesigner BPM supports many of the most popular process languages:

- Analysis languages — [no code generation] used by business analysts to describe the system without any implementation details:
 - Analysis - An implementation-neutral notation.
 - BPMN - A standard graphical notation to represent the control flow of a business process. Suitable to refine the analysis of a system with respect to standards (see *Chapter 10, Business Process Modeling Notation (BPMN)* on page 183).
 - Data Flow Diagram - For identifying data exchanges between processes (see *Chapter 11, Data Flow Diagram (DFD)* on page 199).
- Service Orchestration languages — (or execution languages) used by technical analysts to describe the implementation of business processes as Web services or applications and define how they can be connected to accomplish specific tasks:
 - Service Oriented Architecture (SOA) - [no code generation] Suitable to define the invocation of services by processes (see *Chapter 12, Service Oriented Architecture (SOA)* on page 207).
 - BPEL4WS 1.1 or WS-BPEL 2.0 - Suitable to define the invocation of services by processes. Focus on the implementation of one partner engaged in the collaboration of

a BPM associated with ebXML (see *Chapter 13, BPEL4WS and WS-BPEL* on page 209).

- Sybase® WorkSpace Business Process 1.5: Used to implement processes using Business Process Service in Sybase WorkSpace (see *Chapter 14, Sybase WorkSpace* on page 227).
- Collaborative languages — used by business analyst to document Business-to-Business exchanges (B2B):
 - ebXML 1.01 and 1.04: Choreography language, which describes the collaboration agreements between partners, such as two banks, that are all considered at the same level (see *Chapter 15, Electronic Business XML (EbXML)* on page 251).

Objects that are available to you in your model depends on the process language you have selected. For example, if you select the Analysis process language, the data transformation object is not available.

Note: If you created a model with PowerDesigner 9 and attached a XEM (such as ebXML for example), the model will be automatically linked to the most appropriate process language, otherwise it will be linked by default to the Analysis process language.

Suggested Bibliography

- The Workflow reference Model - <http://www.wfmc.org>.
- Business Process Model Language Specification - <http://www.bpmi.org/>.
- Document: Business Process Specification Schema - <http://www.ebxml.org/>.
- Alan Kotok, David R. Webber, David RR Webber - ebXML: The New Global Standard for Doing Business on the Internet - New Riders Publishing, 2001.
- Business Process Execution Language for Web Services Specification – <http://www.ibm.com/developerworks/library/specification/ws-bpel/>.

Creating a BPM

You create a new business process model by selecting **File > New Model**.

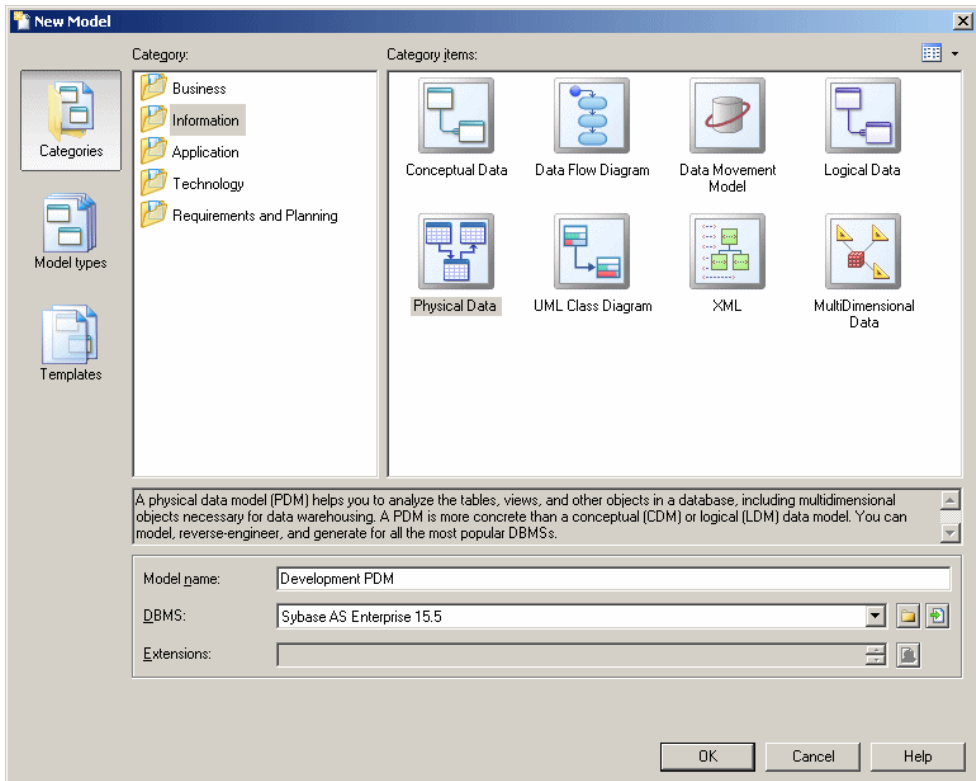
Note: In addition to creating a BPM from scratch with the following procedure, you can also:

- Reverse-engineer existing process language code (see *Chapter 7, Generating and Reverse Engineering Process Languages* on page 169).
 - Import a SIMUL 8 file (see *Chapter 5, Simulating a Business Process Model with SIMUL8* on page 129).
 - Open a legacy PowerDesigner Process Analyst Model (see *Chapter 11, Data Flow Diagram (DFD)* on page 199).
-

The New Model dialog is highly configurable, and your administrator may hide options that are not relevant for your work or provide templates or predefined models to guide you through

model creation. When you open the dialog, one or more of the following buttons will be available on the left hand side:

- **Categories** - which provides a set of predefined models and diagrams sorted in a configurable category structure.
- **Model types** - which provides the classic list of PowerDesigner model types and diagrams.
- **Template files** - which provides a set of model templates sorted by model type.



1. Select **File > New Model** to open the New Model dialog.
2. Click a button, and then select a category or model type (**Business Process Model**) in the left-hand pane.
3. Select an item in the right-hand pane. Depending on how your New Model dialog is configured, these items may be first diagrams or templates on which to base the creation of your model.

Use the **Views** tool on the upper right hand side of the dialog to control the display of the items.

4. Enter a model name.

The code of the model, which is used for script or code generation, is derived from this name using the model naming conventions.

5. Select a target process language , which customizes PowerDesigner's default modifying environment with target-specific properties, objects, and generation templates.

By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the **Embed Resource in Model** button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.

6. [optional] Click the **Select Extensions** button and attach one or more extensions to your model.
7. Click **OK** to create and open the business process model .

Note: Sample BPMs are available in the Example Directory.

BPM Properties

You open the model property sheet by right-clicking the model in the Browser and selecting **Properties**.

Each business process model has the following model properties:

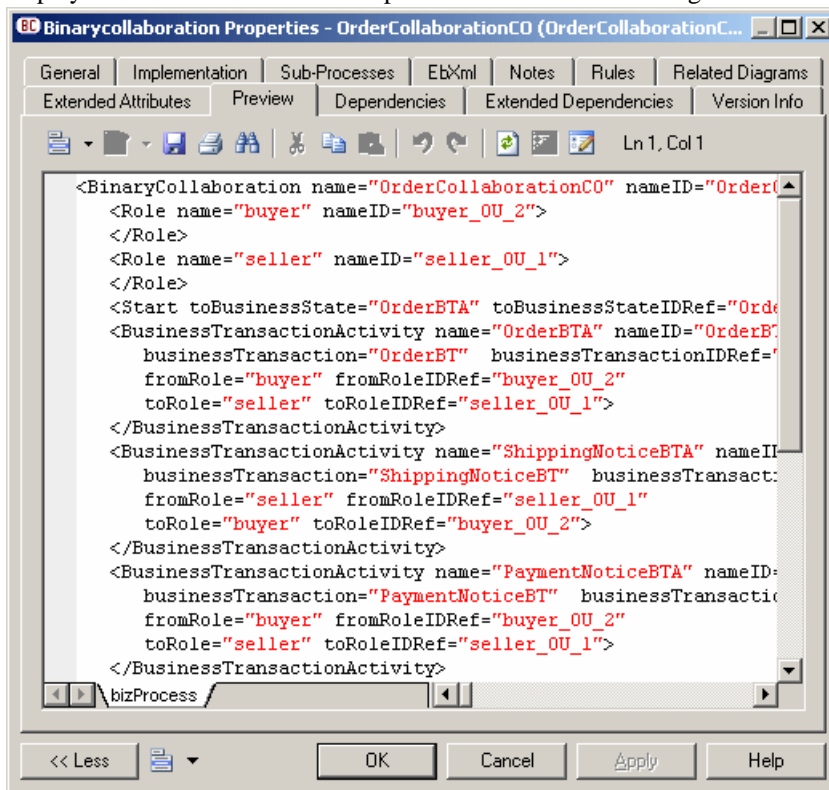
Property	Description
Name/Code/Comment	Identify the model. The name should clearly convey the model's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the model. By default the code is auto-generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Filename	Specifies the location of the model file. This box is empty if the model has never been saved.
Author	Specifies the author of the model. If you enter nothing, the Author field in diagram title boxes displays the user name from the model property sheet Version Info tab. If you enter a space, the Author field displays nothing.
Version	Specifies the 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.
Process language	Specifies the model target.
Default diagram	Specifies the diagram displayed by default when you open the model.

Property	Description
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Previewing Process Code









Click the **Preview** tab in the property sheet of the model, packages, processes, and various other model objects in order to view the code that will be generated for it.



For example, if you have selected a Service Orchestration family language, the Preview page displays the Schema file that corresponds to the XML file to be generated.



The following tools are available on the **Preview** tab toolbar:

CHAPTER 1: Getting Started with Business Process Modeling

Tools	Description
	<p>Editor Menu [Shift+F11] - Contains the following commands:</p> <ul style="list-style-type: none"> • New [Ctrl+N] - Reinitializes the field by removing all the existing content. • Open... [Ctrl+O] - Replaces the content of the field with the content of the selected file. • Insert... [Ctrl+I] - Inserts the content of the selected file at the cursor. • Save [Ctrl+S] - Saves the content of the field to the specified file. • Save As... - Saves the content of the field to a new file. • Select All [Ctrl+A] - Selects all the content of the field. • Find... [Ctrl+F] - Opens a dialog to search for text in the field. • Find Next... [F3] - Finds the next occurrence of the searched for text. • Find Previous... [Shift+F3] - Finds the previous occurrence of the searched for text. • Replace... [Ctrl+H] - Opens a dialog to replace text in the field. • Go To Line... [Ctrl+G] - Opens a dialog to go to the specified line. • Toggle Bookmark [Ctrl+F2] Inserts or removes a bookmark (a blue box) at the cursor position. Note that bookmarks are not printable and are lost if you refresh the tab, or use the Show Generation Options tool • Next Bookmark [F2] - Jumps to the next bookmark. • Previous Bookmark [Shift+F2] - Jumps to the previous bookmark.
	<p>Edit With [Ctrl+E] - Opens the previewed code in an external editor. Click the down arrow to select a particular editor or Choose Program to specify a new editor. Editors specified here are added to the list of editors available at Tools > General Options > Editors.</p>
	<p>Save [Ctrl+S] - Saves the content of the field to the specified file.</p>
	<p>Print [Ctrl+P] - Prints the content of the field.</p>
	<p>Find [Ctrl+F] - Opens a dialog to search for text.</p>
	<p>Cut [Ctrl+X], Copy [Ctrl+C], and Paste [Ctrl+V] - Perform the standard clipboard actions.</p>
	<p>Undo [Ctrl+Z] and Redo [Ctrl+Y] - Move backward or forward through edits.</p>
	<p>Refresh [F5] - Refreshes the Preview tab.</p> <p>You can debug the GTL templates that generate the code shown in the Preview tab. To do so, open the target or extension resource file, select the Enable Trace Mode option, and click OK to return to your model. You may need to click the Refresh tool to display the templates.</p>

Tools	Description
	Select Generation Targets [Ctrl+F6] - Lets you select additional generation targets (defined in extensions), and adds a sub-tab for each selected target. For information about generation targets, see <i>Customizing and Extending PowerDesigner > Extension Files > Extending Generation and Creating Separate Generation Targets</i> .
	Show Generation Options [Ctrl+W] - Opens the Generation Options dialog, allowing you to modify the generation options and to see the impact on the code.

Customizing your Modeling Environment

The PowerDesigner business process model provides various means for customizing and controlling your modeling environment.

Setting Model Options

You can set BPM model options by selecting **Tools > Model Options** or right-clicking the diagram background and selecting **Model Options**.

You can set the following options on the Model Settings page:

Option	Description
Name/Code case sensitive	Specifies that the names and codes for all objects are case sensitive, allowing you to have two objects with identical names or codes but different cases in the same model. If you change case sensitivity during the design process, we recommend that you check your model to verify that your model does not contain any duplicate objects.
Enable links to requirements	Displays a Requirements tab in the property sheet of every object in the model, which allows you to attach requirements to objects (see <i>Requirements Modeling</i>).
External Short-cut Properties	Specifies the properties that are stored for external shortcuts to objects in other models for display in property sheets and on symbols. By default, All properties appear, but you can select to display only Name/Code to reduce the size of your model. Note: This option only controls properties of external shortcuts to models of the same type (PDM to PDM, EAM to EAM, etc). External shortcuts to objects in other types of model can show only the basic shortcut properties.

Option	Description
Default Message Format	<p>Specifies the default setting for the Message Format property for flows and resource flows. You can choose:</p> <ul style="list-style-type: none"> • None - Flows are created without any default message format, as the event is of minor importance. You may choose this option if you do not want to specify data flows in your BPM. • Undefined - Flows are created with an undefined message format, which you will specify subsequently.
Data Flow Diagram Notation	[Data Flow Diagram only] Specifies whether to use the Gane & Sarson or Yourdon notation for your DFD symbols.

For information about controlling the naming conventions of your models, see *Core Features Guide > The PowerDesigner Interface > Objects > Object Properties > Naming Conventions*.

Setting BPM Display Preferences

PowerDesigner display preferences allow you to customize the format of object symbols, and the information that is displayed on them. To set business process model display preferences, select **Tools > Display Preferences** or right-click the diagram background and select **Display Preferences** from the contextual menu.

For detailed information about customizing and controlling the attributes and collections displayed on object symbols, see *Core Features Guide > The PowerDesigner Interface > Diagrams, Matrices, and Symbols > Display Preferences*.

Viewing and Editing the Process Language Definition File

Each BPM is linked to a definition file that extends the standard PowerDesigner metamodel to provide objects, properties, data types, and generation parameters and templates specific to the language being modeled. Definition files and other resource files are XML files located in the `Resource Files` directory inside your installation directory, and can be opened and edited in the PowerDesigner Resource Editor.

Warning! We strongly recommend that you make a back up of the resource files delivered with PowerDesigner before editing them.

To open your model's definition file and review its extensions, select **Language > Edit Current Process Language**.

For detailed information about the format of these files, see *Customizing and Extending PowerDesigner > Object, Process, and XML Language Definition Files*.

Note: 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

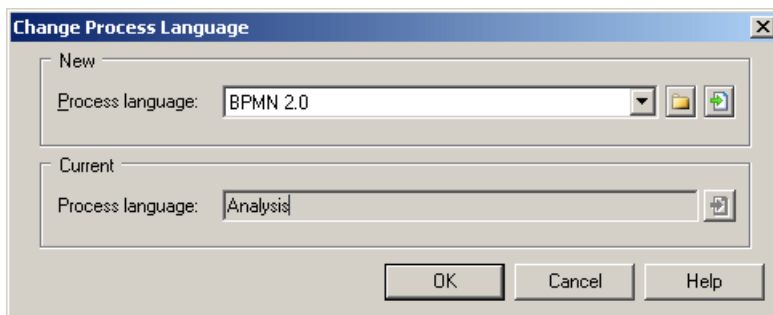
accepting bug reports and will provide fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. Users are invited to assist Sybase by testing fixes of the definition provided by Sybase and report any continuing inconsistencies.

Changing the Process Language

You can change the **process language** being modeled in your BPM at any time.

Note: You may be required to change the process language if you open a model and the associated definition file is unavailable. Language definition files are frequently updated in each version of PowerDesigner and it is highly recommended to accept this change, or otherwise you may be unable to generate for the selected language.

1. Select **Language > Change Current Process Language:**



2. Select a **process language from the list.**

By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the **Embed Resource in Model** button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.

3. Click **OK.**

A message box opens to tell you that the process language has been changed.

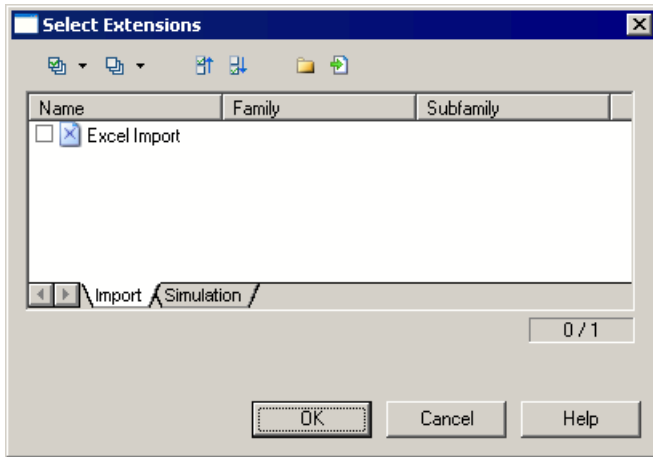
4. Click **OK to return to the model.**

Extending your Modeling Environment

You can customize and extend PowerDesigner metaclasses, parameters, and file generation with extensions, which can be stored as part of your model or in separate extension files (*.xem) for reuse with other models.

To access extension defined in a *.xem file, simply attach the file to your model. You can do this when creating a new model by clicking the **Select Extensions** button at the bottom of the New Model dialog, or at any time by selecting **Model > Extensions** to open the List of Extensions and clicking the **Import an Extension** tool.

In each case, you arrive at the Select Extensions dialog, which lists the extensions available, sorted on sub-tabs appropriate to the type of model you are working with:



To get started extending objects, see *Core Features Guide > The PowerDesigner Interface > Objects > Extending Objects*. For detailed information about working with extensions, see *Customizing and Extending PowerDesigner > Extension Files*.

Linking Objects with Traceability Links

You can create traceability links to show any kind of relationship between two model objects (including between objects in different models) via the **Traceability Links** tab of the object's property sheet. These links are used for documentation purposes only, and are not interpreted or checked by PowerDesigner.

For more information about traceability links, see *Core Features Guide > Linking and Synchronizing Models > Getting Started with Linking and Syncing > Creating Traceability Links*.

Building Process Hierarchy Diagrams

A *process hierarchy diagram* (or functional decomposition diagram) provides a graphical view of the functions of a system and helps you decompose them into a tree of sub-processes.

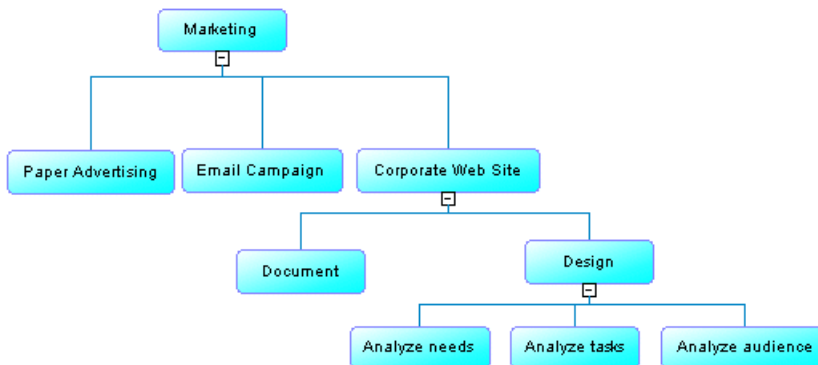
Note: To create a process hierarchy diagram in an existing BPM, right-click the model in the Browser and select **New > Process Hierarchy Diagram**. To create a new model, select **File > New Model**, choose Business Process Model as the model type and **Process Hierarchy Diagram** as the first diagram, and then click **OK**.

The PHD is mostly used during the analysis phase of a project. Business analysts and managers use it to:

- Define all the processes performed in the scope of a business function
- Focus on process recognition and enumeration; only process names are defined at this step
- Decompose already identified processes into sub-processes until an appropriate atomic level is reached
- Reorganize sub-processes, if necessary, by changing their parent
- Display in a single view the whole hierarchy of an already described process or of any decomposed sub-process

During the creation of your diagram, you can move or reuse processes as necessary.


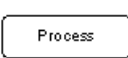


In the following example, the root process, Marketing, is decomposed into three sub-processes, Paper Advertising, Email Campaign, and Corporate Web Site. The latter is in turn decomposed into two sub-processes, and so on:



Each of these processes can be analyzed in its own business process diagram (see *Chapter 3, Building Business Process Diagrams* on page 21).

Process Hierarchy Diagram Objects

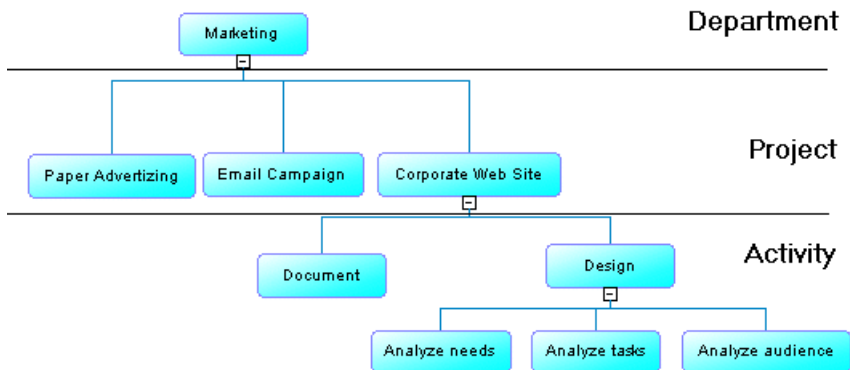
PowerDesigner supports all the objects necessary to build process hierarchy diagrams.

Object	Tool	Symbol	Description
Process			Business function within an organization that can be decomposed into smaller and smaller related parts until it reaches a sufficient level of decomposition. See <i>Building Process Hierarchies</i> on page 16. For detailed information about process objects and the properties you can specify for them, see <i>Processes (BPM)</i> on page 32.
Decomposition Link			Hierarchical link between two processes.

Building Process Hierarchies

A process hierarchy comprises a set of processes and the decomposition links that connect them.

At each level of decomposition, processes can represent different types of functions. In the following example, the Marketing process is a department, the Corporate Web Site is a project, and the Analyze needs process is an activity:


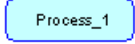

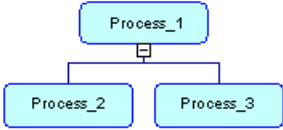


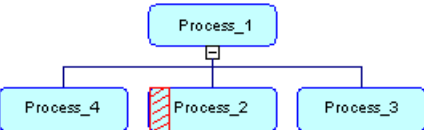


If you are working with the Data Flow Diagram process language, you can analyze each process at the first level of your hierarchy in its own context diagram to show the primary

relationships between the organization and the external entities with which it deals (see *Chapter 11, Data Flow Diagram (DFD)* on page 199).

Building Hierarchies with the Process Tool

You build a process hierarchy using the Process tool. The following table summarizes the different types of processes that you can create with this tool:

To create a ...	Cursor	Click...
Root process		Any empty space in the diagram window. Example: 
Sub-process		A root process symbol or the bottom part of any other process. Example: 
Sibling process	 	The left or right part of any process symbol, except the root process. Example: 

By default, a process hierarchy displays from top to bottom. To change the orientation, select **Tools > Display Preferences > General** in the menu bar, and select Horizontal in the Orientation groupbox. Symbols are automatically re-arranged to respect the new display preference.

You cannot use the composite view feature (see *Decomposing Processes* on page 44) for a process in a PHD.

Note: You can also create root processes by right-clicking the model node in the Browser, and selecting **New > Process**, or from the list of processes available from the Model menu. You can decompose a process and create sub-processes in the Browser by right-clicking a process and selecting Decompose Object in the contextual menu. Except for root processes, processes created in the Browser are not automatically displayed in the diagram. To display one or all of the missing sub-process symbols, right-click the parent process symbol, and select Complete or Complete All in the contextual menu.

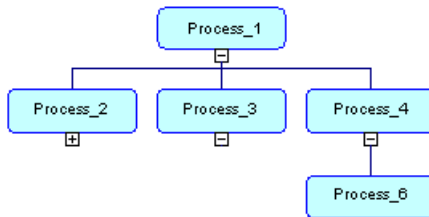
Expanding and Collapsing Process Hierarchies

You can expand and collapse all or part of your hierarchy by right-clicking the + and — signs at the bottom of your process symbols.

Atomic processes have no sign, while a decomposed process without any sub-processes has a — sign.

Example

In the following example, Process_1 and 4 are expanded, Process_2 is collapsed, Process_3 is decomposed, and process_6 is atomic.



Note: You can expand one or all levels of a part of a hierarchy by right-clicking the parent process symbol and select Expand (All) from the contextual menu. Only the first level or all levels of a part of the hierarchy is expanded.

Completing the Hierarchy

You can delete a sub-process symbol from the hierarchy in the diagram window without deleting the object from the model. You can use the Complete and Complete All commands from the contextual menu of a parent process symbol to redisplay one or all missing sub-processes in the hierarchy.

Arranging, Moving, and Reusing Processes in a Hierarchy

When building your PHD, you may find that you need to arrange or move your symbols, or that you want to reuse a process in more than one place in the hierarchy.



Arranging Processes

To re-arrange your symbols, you can:

- Right-click a root process or parent process symbol, and select Arrange Symbols from the contextual menu. All the sub-processes of the selected process are evenly spaced.
- Select **Symbol > Auto-Layout**. All hierarchies are arranged in horizontal or vertical rows depending on the display preferences. The relative position of the processes within a hierarchy is unchanged. If you have previously selected one or more symbols in the diagram, you will be prompted to choose whether to layout the selected symbols or all symbols.

Moving Processes

You can move a process or a decomposition link in the diagram window:

When you move a...symbol	Cursor	Result
Process		The process and any sub-processes if any are moved. If you drop the process onto another process, it becomes a sub-process of the second process.
Decomposition link		All the processes, beneath it in the hierarchy, are moved. Note that only the horizontal or vertical part of a link (depending on the display preferences) can be selected for the move, and you can only move a link up or down.

Reusing Processes

You can reuse a process that already exists in your hierarchy in order to avoid duplicating its functions in your model.

1. Right-click the process within which you want to reuse the process, and select Reuse Process from the contextual menu. An object selection dialog box opens, which lists all the other processes available in the model.
2. Select the process that you want to reuse and click OK.

A shortcut to the selected process is added as a sub-process to the first process.

By default, the symbol of the shortcut displays its target.

A process shortcut is always displayed as an atomic process. You cannot decompose the shortcut or expand its hierarchy, even if its target object has sub-processes.

Creating Default Flows Between Processes in a Business Process Diagram

For each process that has sub-processes, PowerDesigner can create a default control flow linking all of its sub-processes in a business process diagram.

Right-click a parent process in the PHD and select Build Default Flows between Processes in the contextual menu.

The control flow is automatically displayed in a business process diagram. The flow contains a start and an end with all the sub-processes linked by flows between them. Note that only the first level of the hierarchy is displayed. You can further refine the control flow by creating other objects in the diagram.

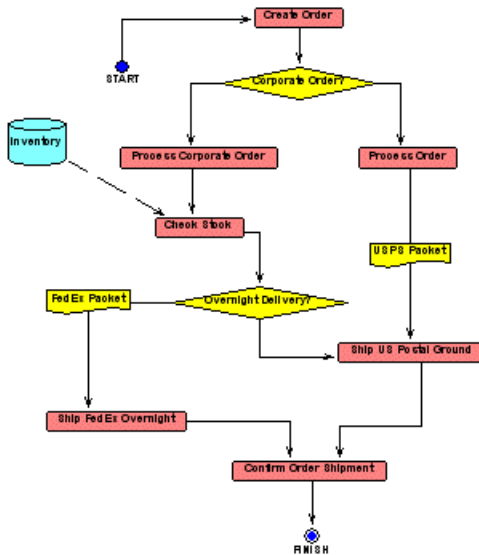
For more information, see *Chapter 3, Building Business Process Diagrams* on page 21.

Building Business Process Diagrams

A *business process diagram* (or process flow diagram) provides a graphical view of the control flow (the sequence of execution) or data flow (the exchange of data) between processes at any level in your system.

A business process diagram can be created in a model, a package or within a decomposed process.

In the following example the path of an order depends on whether it is a corporate order. The control flow passes through the Process Corporate Order process, then through the Check Book process, which checks the book availability in the Inventory resource. The check is done through the Inventory resource. Then the control flow path depends on whether it is an overnight delivery. If yes, the control flow passes through the Ship FedEx Overnight process with a message format specifying the format of the information exchanged (an administrative form for example). Then the shipment is confirmed. In any cases the control flow will go to Finish whether or not it is a corporate order.



There are three types of BPDs for modeling different aspects of a system:

- *Top-level diagram* – focuses on the roles played by business partners in relation to a system (see *Top-Level Diagram Basics* on page 22)
- *Choreography diagram* – focuses on allocating responsibility for activities, choreographing objects, analyzing flows of data, and modeling the implementation of activities (see *Choreography Diagram Basics* on page 23)
- *Data flow diagram* – focuses on data exchange between processes (see *Data Flow Diagram Basics* on page 32)

Top-Level Diagram Basics

A top-level diagram is a special form of business process diagram required by ebXML and BPEL languages, which provides a high-level representation of a system and its interactions with business partners.

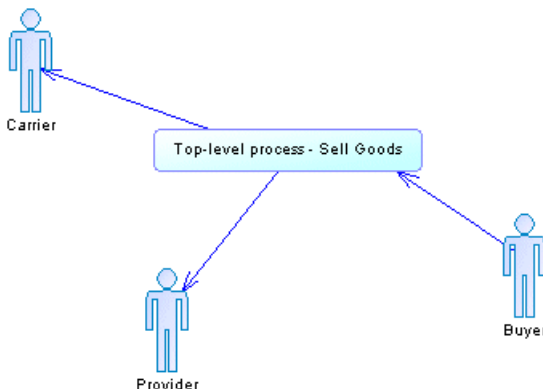
Note: To create a business process diagram in an existing BPM, right-click the model in the Browser and select **New > Business Process Diagram**. To create a new model, select **File > New Model**, choose Business Process Model as the model type and **Business Process Diagram** as the first diagram, and then click **OK**.

For other languages, the top-level diagram is simply the highest level of choreography diagram (see *Choreography Diagram Basics* on page 23).

When working with ebXML and BPEL languages, business or technical analysts should identify the business partners of their system in order to specify its scope and the interactions with those partners.

Partners are connected to a top-level process by role associations and can perform either initiating or responding roles in relation to the system.


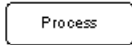




In the following example, Carrier, Provider, and Buyer are business partners, which interact with the Sell Goods top-level process. The Buyer performs an initiating role in relation to the system, while the Provider and the Carrier perform a responding role:



Having created a top-level diagram, you can then decompose your top-level process to create a choreography diagram (see *Choreography Diagram Basics* on page 23).

Top-level Diagram Objects

PowerDesigner supports all the objects necessary to build top-level diagrams.

Object	Tool	Symbol	Description
Process			Top-level process that interacts with business partners (see <i>Processes (BPM)</i> on page 32).
Organization unit			Business partner (a company, a system, a service, an organization, a user or a role) that interacts with the top-level process (see <i>Organization Units (BPM)</i> on page 52).
Role association			Interaction between a top-level process and a business partner (see <i>Role Associations (BPM)</i> on page 73).

Choreography Diagram Basics

You create a choreography diagram for any decomposed process for ebXML and BPEL languages, and at any level of the process hierarchy for other languages. Each choreography diagram contains a control flow, which organizes the sub-processes directly below the process from which the diagram has been created.

Note: To create a choreography diagram in an existing BPM, right-click the model in the Browser and select **New > Choreography Diagram**. To create a new model, select **File > New Model**, choose Business Process Model as the model type and **Choreography Diagram** as the first diagram, and then click **OK**.

The choreography diagram is the core BPM diagram, which lets you perform the following tasks:

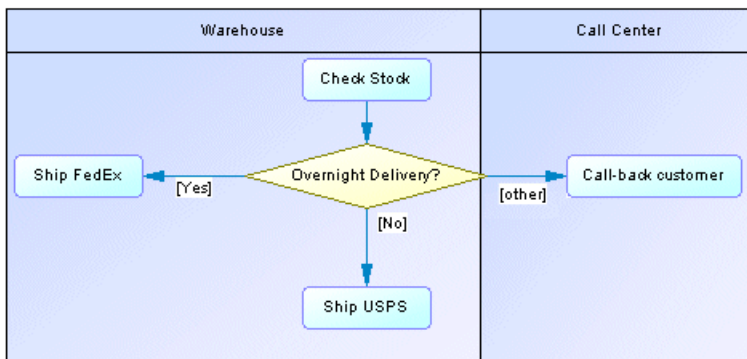
- Allocate responsibilities to organization units in a system (see *Allocating responsibilities* on page 24)
- Trace the choreography of the processes in a system (see *Tracing the process choreography* on page 24)
- Analyze how data flows through a system (see *Analyzing data* on page 25)
- Model the implementation of processes in a system (see *Modeling the implementation of processes* on page 29)

Allocating Responsibilities

You can use a choreography diagram to analyze how processes modeled in the system will be allocated to people, groups or organizations. These resources are modeled as organization units and are represented in the choreography diagram as swimlanes.

You allocate a process to an organization unit by placing it in the appropriate swimlane (see *Attaching and detaching a process to/from an organization unit* on page 55). Allocating responsibilities in this way helps you to avoid unassigned tasks and duplicated assignments.

In the following example the Warehouse organization unit is responsible for checking the stock and managing the shipping of goods, and the Call Center organization unit is responsible for calling back customers:

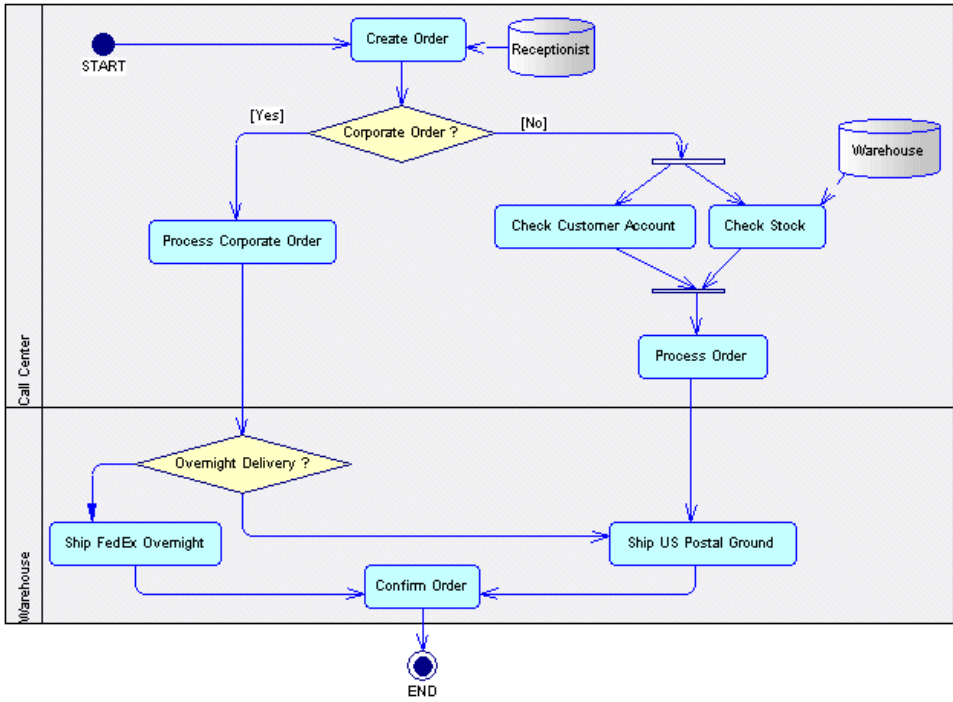


Tracing the Process Choreography

A choreography diagram describes the control flow of a process by showing a path from one or more starts through a sequence of sub-processes, decisions, synchronizations, and resources to one or more ends. The placement of a process in a swimlane or another shows which organization unit is responsible for it. The parent process being analyzed in the diagram must wait for the end of all its sub-processes before it terminates.

In BPMN and orchestration languages, you can model a break in the normal flow of a process using events (see *Events (BPM)* on page 75). You can catch an event using an event handler (see *Event handlers* on page 77) or generate an event from the process (see *Process Properties* on page 34).

In the following example, the processing of an order proceeds differently depending on whether or not it is a corporate order. Both possible paths are reunited in the Confirm Order process:

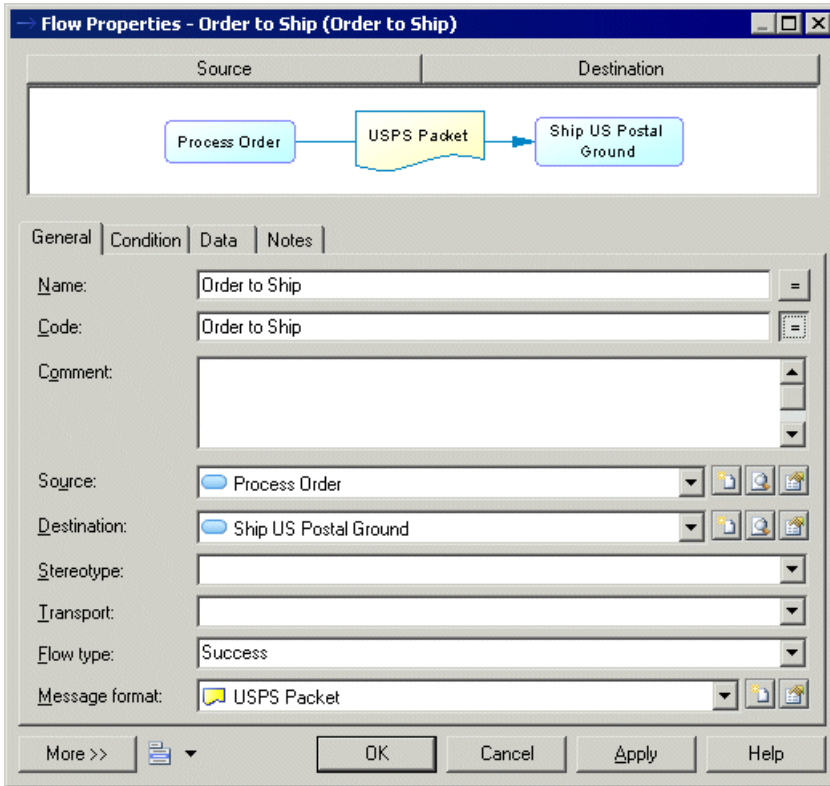


Analyzing Data

The choreography diagram provides various ways to model the flow of data in the system.

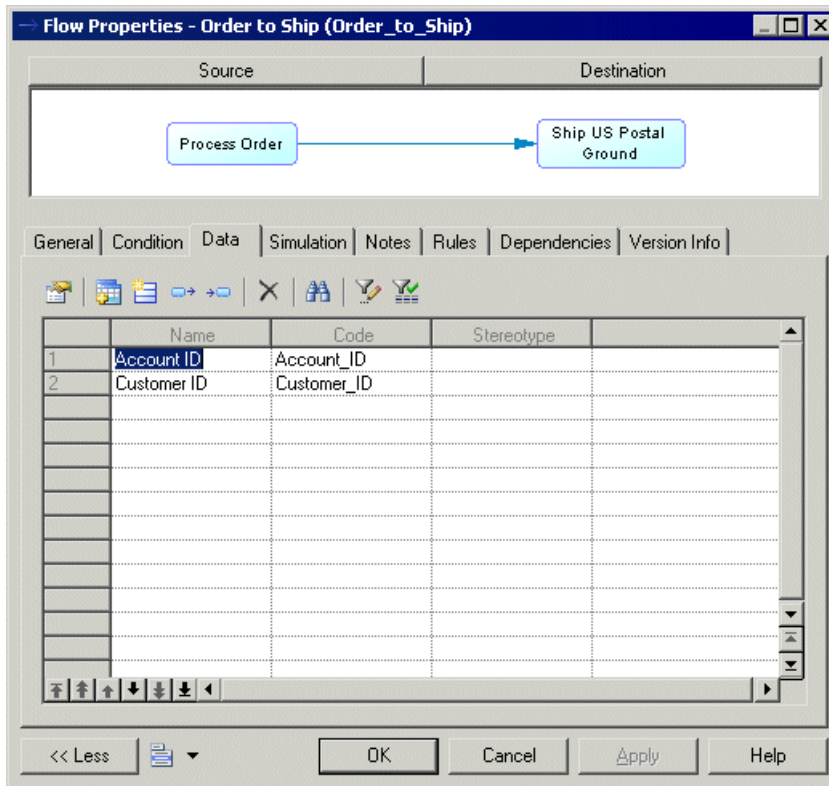
- Message formats on flows – [Analysis language and ebXML language only] to define an exchange format for large amounts of data that transit between processes. Message formats can be useful for web services, and are usually defined by a DTD or an XSD (see *Message Formats (BPM)* on page 78).

In the following example, the Order to Ship flow is associated with the USPS Packet message format to specify the format of the data exchanged between the Process Order process and the Ship US Postal Ground process:



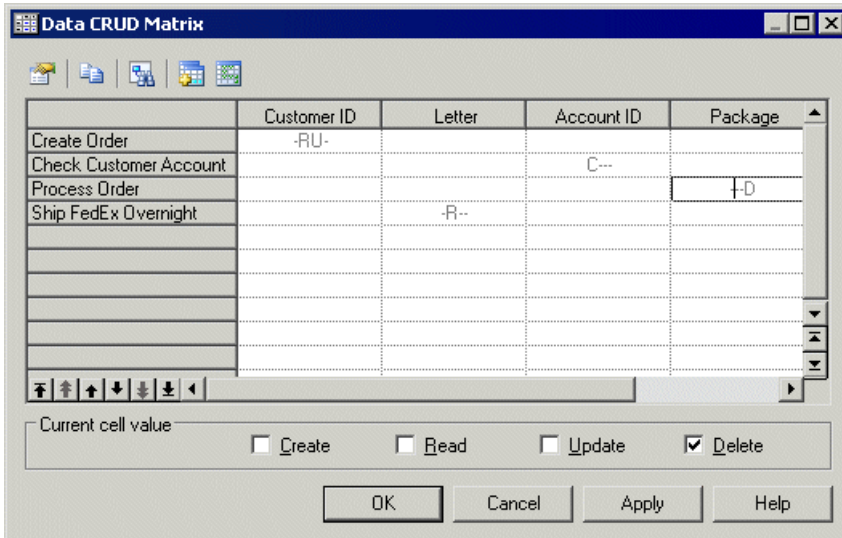
- Data on flows – [Analysis language and DFD language only] to model data (see *Data (BPM)* on page 83) without specifying its format. Data can be associated with objects defined in a PDM, OOM, or CDM (see *Exchanging data with other PowerDesigner models* on page 85).

In the following example, the Order to Ship flow conveys the Account ID and Customer ID data from the Process Order process to the Ship US Postal Ground process:



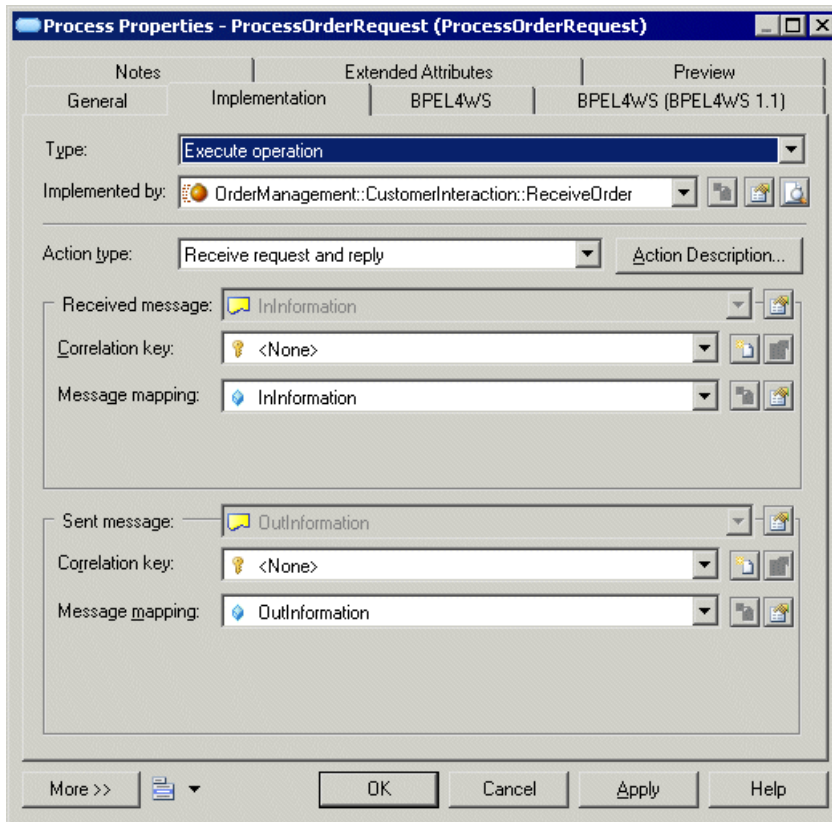
- Data CRUD – [Analysis, DFD, and ebXML languages only] to specify the actions (create, read, update and delete) a process can perform on data (see *Working with Data and Resource CRUD Matrices* on page 50).

In the following example, the Data CRUD Matrix shows that the Create Order process Reads and updates the Customer ID data, the Check Customer Account process creates the Account ID data, and so on:



- Input/output messages on processes – [orchestration languages only] to specify data exchange between partners (see *Example: Using the Execute operation implementation type* on page 40). No data is specified on flows between processes when modeling with orchestration languages.

In the following example, the ProcessOrderRequest process is implemented by a ReceiveOrder operation, which receives an "Ininformation" message from a partner and replies with an "Outinformation" message:



Note: See also the data flow diagram, which offers another way of analyzing data, by focusing on data exchange between processes (see *Data Flow Diagram Basics* on page 32).

Modeling the Implementation of Processes

The choreography diagram provides various ways to model the implementation of processes in a system.


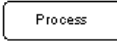
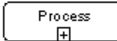

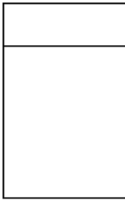





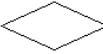






You may start by analyzing your system then import a WSDL file (see *Importing a Service Provider from a WSDL File* on page 103) in order to implement your processes. You can:






- Describe your processes in a textual way, as a series of actions, which can be done by an employee for example
- Automate your processes using an execution engine [orchestration languages]
 - Implement your process by a service provider operation (see *Example: Using the Execute Operation Implementation Type* on page 40)
 - Perform data transformations (see *Specifying Implementation Types* on page 37)
 - Generate events (see *Specifying Implementation Types* on page 37)

- Describe the internal behavior of each partner involved in a collaborative process [collaborative languages] (see *Chapter 15, Electronic Business XML (EbXML)* on page 251)
- Describe a loop process (see *Specifying Implementation Types* on page 37)
- Use an existing process to specify the implementation of your current process [analysis languages (except DFD) and collaborative languages] (see *Specifying Implementation Types* on page 37)

Choreography Diagram Objects

PowerDesigner supports all the objects necessary to build choreography diagrams.

Object	Tool	Symbol	Description
Process			Task to perform (see <i>Processes (BPM)</i> on page 32).
Composite process	None		Complex process decomposed to be further detailed (see <i>Processes (BPM)</i> on page 32).
Organization unit			Organization, service or person that is responsible for a process (see <i>Organization Units (BPM)</i> on page 52).
Role association			Unidirectional relationship that models a link between objects (see <i>Role Associations (BPM)</i> on page 73).
Flow			Path of the control flow between processes (see <i>Flows (BPM)</i> on page 70).
Decision			Decision to take when several flow paths are possible. Only one path will be triggered at execution time (see <i>Decisions (BPM)</i> on page 65).
Synchronization			Enables synchronization of flows between two or more concurrent actions or allows the design of a split (see <i>Synchronizations (BPM)</i> on page 68).
Start			Starting point of the processes described in the choreography diagram (see <i>Starts (BPM)</i> on page 62).
End			Termination point of the processes described in the choreography diagram (see <i>Ends (BPM)</i> on page 63).

Object	Tool	Symbol	Description
Events	None	None	Instantaneous and observable occurrence during the course of a business process (see <i>Events (BPM)</i> on page 75).
Message format	None		Format definition of data exchanged between processes (see <i>Message Formats (BPM)</i> on page 78).
Message part	None	None	Portion of the WSDL (Web Services Description Language) message (see <i>Message Parts (BPM)</i> on page 81).
Data	None	None	Piece of information exchanged between processes (see <i>Data (BPM)</i> on page 83).
CRUD matrix	None	—	Table that shows the actions a process can perform on data or resources (see <i>Working with Data and Resource CRUD Matrices</i> on page 50).
Resource			Storage unit of abstract data circulating within the model, which is accessed by a process to perform actions (see <i>Resources (BPM)</i> on page 93).
Resource flow			Access of a process to a resource (see <i>Resource Flows (BPM)</i> on page 95).
Service provider	None	None	Object that contains a set of interfaces and operations (see <i>Service Providers (BPM)</i> on page 98).
Service interface	None	None	Object that contains a set of operations (see <i>Service Interfaces (BPM)</i> on page 110).
Operation	None	None	Input and output elements defined in terms of messages or message parts (see <i>Operations (BPM)</i> on page 111).
XSD document	None	None	Object that contains the data schema handled by a service provider (see <i>XSD Documents</i> on page 117).
Variable	None	None	Data container (see <i>Variables (BPM)</i> on page 119).
Correlation key	None	None	Set of variables, which is used to identify a process instance (see <i>Correlation Keys (BPM)</i> on page 121).

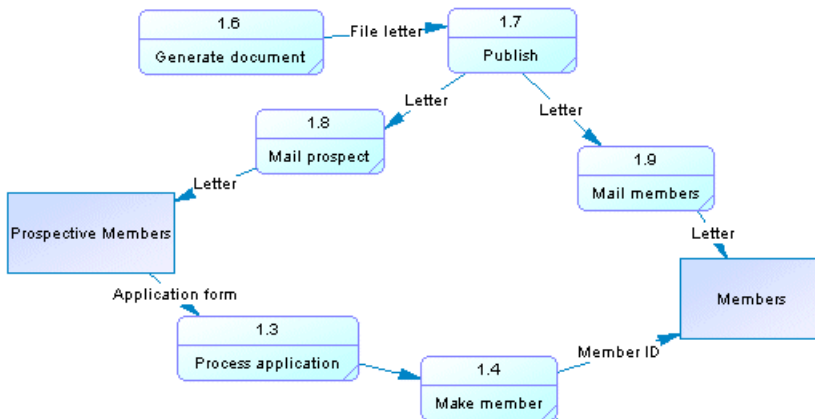
Object	Tool	Symbol	Description
Data transformation	None	None	Object that allows the copy of data from a source to a target with potential transformations (see <i>Data Transformations</i> on page 123).

Data Flow Diagram Basics

A data flow diagram (DFD) lets you graphically represent the flow of data through a system without any indication of time.

Analysts can use DFD to model the functions a system has to carry out and the interactions between those functions in terms of data exchanges between processes.

In the following example, the Publish process sends letters to the Mail prospect and the Mail members processes; these processes send letters to the Prospective Members and Members external entities, and so on:



For more information about the data flow diagram, see *Chapter 11, Data Flow Diagram (DFD)* on page 199.

Processes (BPM)

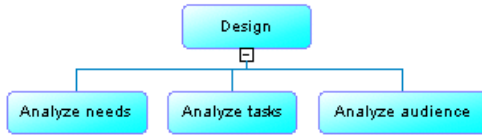
A *process* is a manual or automated action, such as "Process order", or "Send a mail".

You can decompose processes in order to analyze their actions more closely, and further decompose your sub-processes until you reach an appropriate level of detail. A process that is not decomposed is called an atomic process (or activity) (see *Decomposing Processes* on page 44).

A process can be created in the following diagrams with any target languages:

- Process Hierarchy Diagram – each process forms part of a hierarchy, which begins with a single top-level process, and is decomposed into sub-processes. Each decomposed process can be analyzed in its own business process diagram.

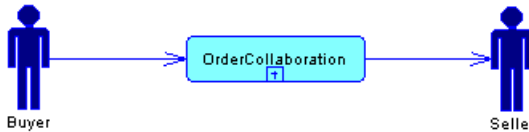
In the following example, the top-level Design process is decomposed into three sub-processes named Analyze needs, Analyze tasks, and Analyze audience:



For more information, see *Chapter 2, Building Process Hierarchy Diagrams* on page 15.

- Business Process Diagram – processes can be created in each of the three types of business process diagrams:
 - Top-level diagram – the process is a top-level process, or global service, that interacts with partners.

In the following example, the top-level OrderCollaboration process interacts with the Buyer and Seller partners:

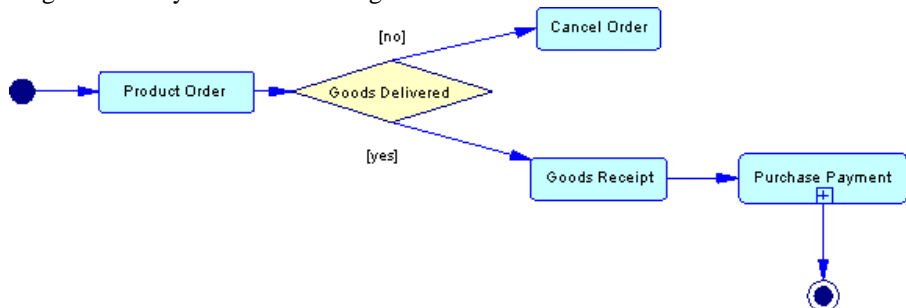


For more information, see *Top-Level Diagram Basics* on page 22.

- Choreography diagram – processes are linked together in a control flow, which passes from one or more starts to one or more ends. When the process gains control, it performs its actions and then, depending on the result of the action, the flow is passed to another process.

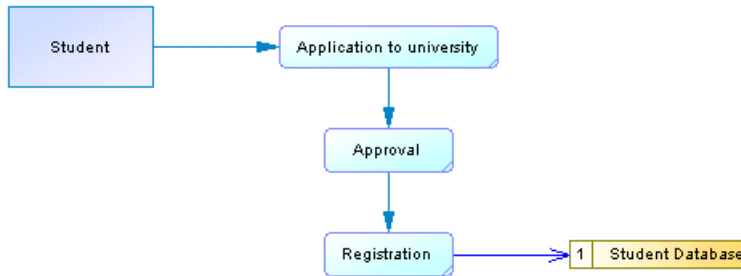
PowerDesigner allows you a great deal of flexibility in your analysis of your processes. You can simply link processes together to show the high-level control flow, or refine your model by specifying their implementation (see *Specifying Implementation Types* on page 37).

In the following example, the path of an order depends on whether the product can be delivered or not. If yes, the control flow passes through the Goods Receipt process, then through the Purchase Payment process, whose details are displayed in a sub-diagram. Finally the control flow goes to Finish:



For more information, see *Choreography Diagram Basics* on page 23.

- Data Flow Diagram – processes are locations where data are transformed.
In the following example, data are sent by the external Student entity through three sequencing processes, and finally are stocked into the Student Database data store:



For more information, see *Chapter 11, Data Flow Diagram (DFD)* on page 199.

Creating a Process

You can create a process from the Toolbox, Browser, or **Model** menu.

- Use the **Process** tool in the Toolbox.
- Select **Model > Processes** to access the List of Processes, and click the **Add a Row** tool.
- Right-click the model, package or decomposed process in the Browser, and select **New > Process**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Process Properties

To view or edit a process's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.

Property	Description
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Organization unit	<p>Specifies the organization unit (see <i>Organization Units (BPM)</i> on page 52) linked to the process. You can select <Committee Process> to specify that it is realized by more than one organization unit (see <i>Displaying a Committee Process</i> on page 55).</p> <p>Click the Properties tool beside this box to open the property sheet of the selected organization unit or the Ellipsis tool to open the list of organization units and create new ones.</p>
Timeout	Specifies the timeout limit. The default value is zero. When the value is not set to zero, it means that a timeout exception occurs if the execution of the activation takes more than the specified timeout limit. You can type any alphanumeric value in the Timeout box (Example: 20 seconds).
Duration	Specifies estimated or statistic duration to execute the action. This property is used for documentation purposes.
Composite status	<p>Specifies whether the process is decomposed into sub-processes. You can choose between:</p> <ul style="list-style-type: none"> • Atomic Process (default) – the process does not contain any sub-processes. • Decomposed Process – the process can contain sub-processes. A Sub-Processes tab is displayed in the property sheet to list these sub-processes, and a sub-diagram is created below the process in the Browser to display them (see <i>Decomposing Processes</i> on page 44). <p>If you revert the process from Decomposed to Atomic status, then any sub-processes that you have created will be deleted.</p>
Number ID	<p>Specifies an incrementing number to help you identify processes. You can modify this value at any time by entering an integer greater than 0. Any change you make will not, by default, affect other numbers in the series.</p> <p>When working in a data flow diagram, you can at any time right-click the diagram background and select Renumber Process IDs to renumber the processes following their position in the data flow (see <i>Process and Data Store Numbering</i> on page 203). Sub-processes inherit the number ID of their parent.</p>
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Implementation Tab

The **Implementation** tab lets you define how the process (activity) is implemented. Note that only decomposed processes can be implemented for the ebXML and BPEL languages:

Depending on the implementation type you specify, the properties available on this tab will change. The following properties are always available.

Property	Description
Type	Specifies the type of process implementation (see <i>Specifying Implementation Types</i> on page 37).
[implementation object]	Depending on the implementation type you choose, an additional field or tab may be displayed, allowing you to specify a process, event, expression, operation, or data transformation upon which the implementation acts. You can use the tools to the right of the list to create an object, browse the available objects, or view the properties of the currently selected object.
Action type	<p>[None and Reuse process implementations only] Specifies the way the process should be executed. You can choose between:</p> <ul style="list-style-type: none">• Manual• Automated• User-defined <p>[Execute operation implementation only] Specifies the type of message exchange the activity performs. You can choose between:</p> <ul style="list-style-type: none">• Receive request – receives a message from a partner.• Receive request and reply – receives a message from a partner and sends a message in response.• Invoke operation – initiates a message sent to a partner, the partner can respond or not.• Reply – sends a message to a partner in response to a received message.• Reply fault – sends a fault message to a partner in response to a received message.
Implementation (text box)	Specifies additional information about the process execution. You can enter any appropriate information in this box, as well as open, insert and save text files. You can open the Implementation tab by right-clicking the process symbol in the diagram, and selecting Implementation in the contextual menu.

Assignments Tab

This tab is only available for a process with the Assign implementation type, and lists the data transformations (see *Data Transformations* on page 123) required for the atomic assign tasks that compose the activity. An assign activity is an Xpath or XSLT expression that allows you to

copy a variable value to another variable value, or to calculate the value of an expression and store it in a variable. The following properties are available:

Properties	Description
Assigned variable	Specifies the variable or organization unit [BPEL languages only] that receives the result of the transformation. Select an object from the list or click the Ellipses button to browse the available objects.
Assigned part	Specifies the message part (when the assigned variable is typed by a message format) that receives the result of the transformation. Select an object from the list.
Input variable	Specifies a source variable or organization unit (to identify the partner to whom the message is sent). Select an object from the list, or click the Ellipses button to browse the available objects.
Input part	Specifies a source message part when the input variable is typed by a message format. Select an object from the list.

The following tabs are also available:

- **Sub-Processes** - [decomposed processes] Lists the sub-processes contained in the process (see *Decomposing processes* on page 44).
- **Local Variables** - [orchestration language decomposed processes] Lists the variables (see *Variables (BPM)* on page 119) local to the current process. Variables are mainly used to build the messages the process sends to its partners.
- **Data** - [Analysis and Data Flow Diagram languages] Lists the data associated with the process. Use the Add Objects and Create an Object tools to add items to the list and select the appropriate CRUD (Create, Read, Update, Delete) columns to specify the types of action the process can perform on the data (see *Data (BPM)* on page 83).

Note: You can migrate the data of a flow to its source or destination process, using the Migrate to Destination Process and Migrate to Source Process tools in the flow property sheet. See *Migrating the data of a flow to a process* on page 92.

Specifying Implementation Types

You can add additional detail to your processes by specifying the type of implementation required for their execution. Note that only decomposed processes can have their implementation specified with ebXML and BPEL languages.

1. Open the property sheet of a process and click the Implementation tab.
2. Select an implementation type. The following list details the available implementation types, and specifies where appropriate, the required implementation object:
 - *<None>* [no object] - default. No implementation defined, or the implementation consists of a textual description in the implementation box.
 - *Loop* [expression text] – (available for all languages) Lets you specify a type of composite activity, which iterates over the set of activities inside of it and creates a

sub-diagram, which details the actions to perform inside the loop. The following loop-specific properties are displayed:

Property	Description
Loop expression	Specifies the loop condition.
Loop type	Specifies the loop type. Some target languages provide predefined types.

- *Reuse process* [process] – (available for Analysis, ebXML, and BPMN) uses another process for its implementation. The following Reuse process-specific property is displayed:

Property	Description
Implemented by	Specifies the implementation process (see <i>Processes (BPM)</i> on page 32).

- *Execute operation* [operation] – (available for orchestration languages) implements a process by a service operation to design the reception and emission of messages. The following execute operation-specific properties are displayed:

Property	Description
Implemented by	Specifies the implementation operation (see <i>Operations (BPM)</i> on page 111). When working with orchestration languages, you can drag and drop an operation from the Browser to the diagram to automatically create an activity (that sends/receives messages) implemented by this operation.
Action description	Specifies the way the action is executed. Click the Action Description button to open a text editor, in which you can enter any appropriate information, as well as open, insert and save text files.
Received message	[when required by the action] Specifies the received message format associated with the selected operation. You can specify a: <ul style="list-style-type: none"> • Correlation key - (see <i>Correlation Keys (BPM)</i> on page 121) which allows the process engine to direct a received message to the correct activity instance. Received correlation key are mostly used for receive request activities. • Message mapping - (see <i>Variables (BPM)</i> on page 119) which retrieves the content of the received message. The variable corresponds to the first message of the operation for receive activities, and to the second message of the operation for activities that send messages.

Property	Description
Sent message	[when required by the action] Specifies the sent message format associated with the selected operation. You can additionally specify a: <ul style="list-style-type: none"> • Correlation key - (see <i>Correlation Keys (BPM)</i> on page 121) which contains the information that is useful to the partner in the next exchange with the activity. • Message mapping - (see <i>Variables (BPM)</i> on page 119) which sends information to a partner. The variable corresponds to the second message of the operation for receive activities, and to the first message of the operation for activities that send messages.

- *Generate Event* [event] – (available for orchestration languages and BPMN) specifies the generation of events, and can be used to raise an exception. The following generate event-specific properties are displayed:

Property	Description
Event	Specifies the implementation event (see <i>Events (BPM)</i> on page 75). You can specify events to model the following specific activities: <ul style="list-style-type: none"> • Wait activity – (timer event) pauses the process for a specified duration, or until a specified time. • Throw activity – (fault event) causes a specific fault to occur to abort a transaction, activity or process and triggers the fault handler (see <i>Event handlers</i> on page 77) for the given process. • Compensate activity – (compensation event) triggers the cancellation of actions performed by an already terminated process using a compensation handler.
Event mapping	[Only available for fault events] Lets you associate a data with the fault by selecting a local variable from the list. This variable stores the fault data.

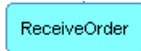
- *Assign* [data transformation] – (available for orchestration languages) uses data transformations to allow the copy of data from one variable to another. Enables the display of the Assignments tab (see *Process Properties* on page 34)
3. Complete any additional fields as necessary to specify a process, event, expression, operation, or data transformation upon which the implementation acts. Use the tools to the right of these fields to create a new object, or view the properties of the currently selected object.
 4. Click OK to save your changes and return to the diagram.

When a process is implemented, its symbol or the graphical symbol in within it changes to correspond to the implementation type you selected.

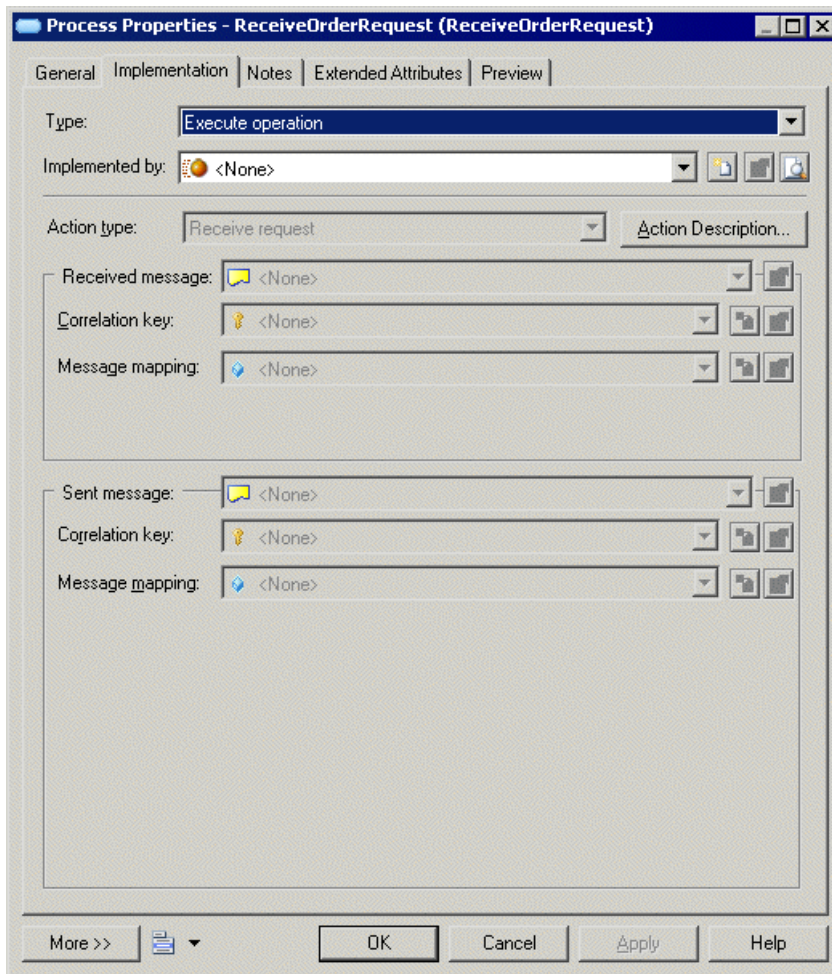
Example: Using the Execute Operation Implementation Type

One of the most common implementation types is Execute operation, which implements the process by a service provider operation, and may specify messages exchanged.

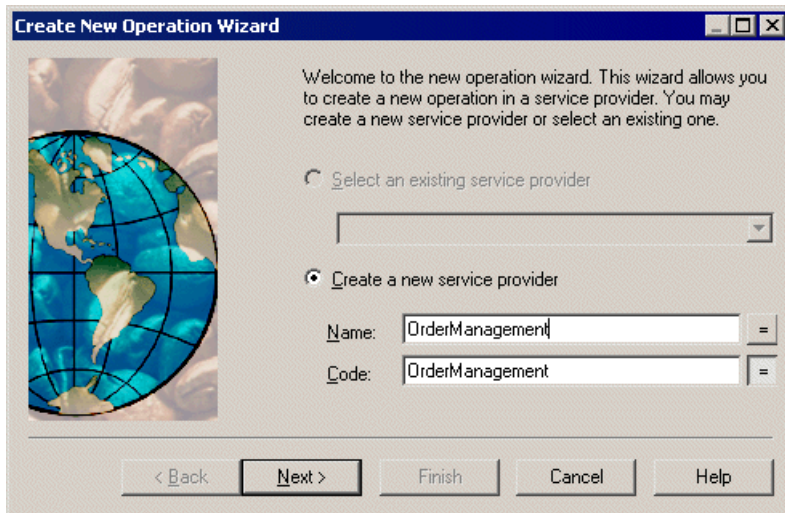
1. Create a process and call it ReceiveOrder.



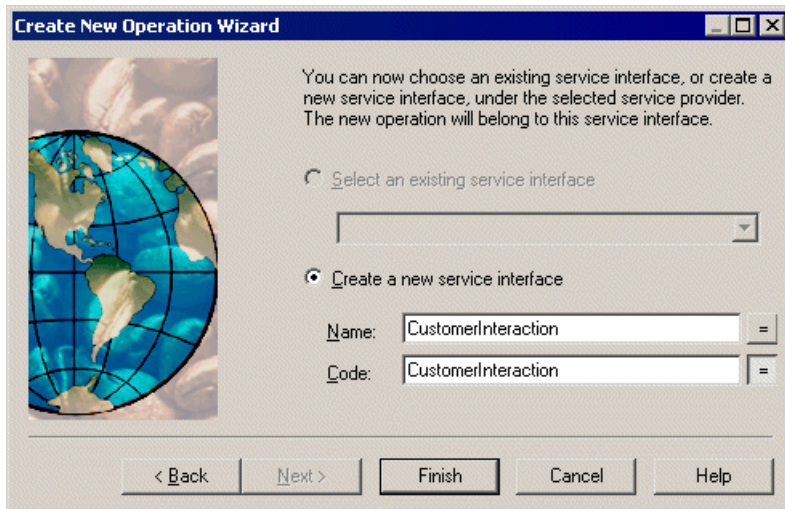
2. Open its property sheet, click the Implementation tab, and select Execute operation from the Type list. The corresponding fields display:



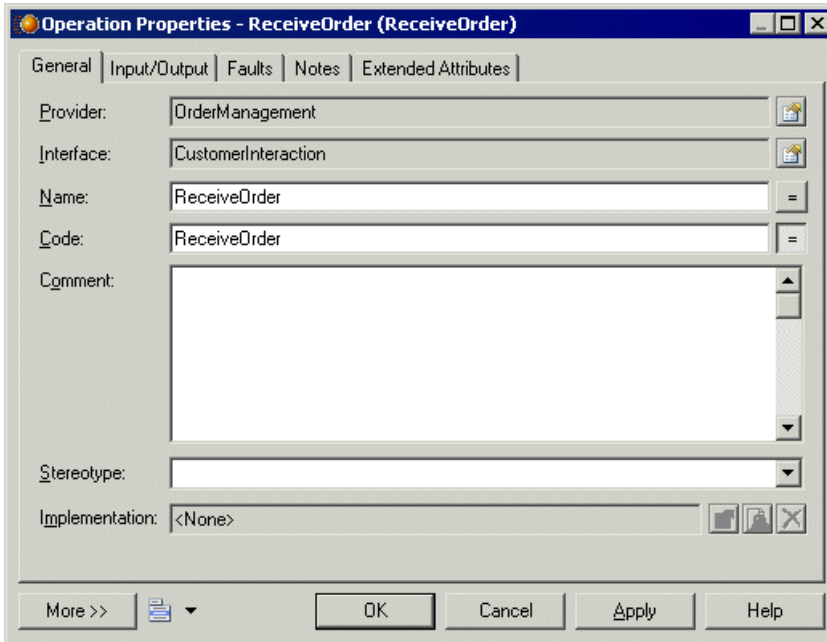
3. Click the Create tool to the right of the Implemented by list to open a wizard to create an operation. First, create a service provider, and call it OrderManagement:



4. Next, create a service interface, and call it CustomerInteraction:



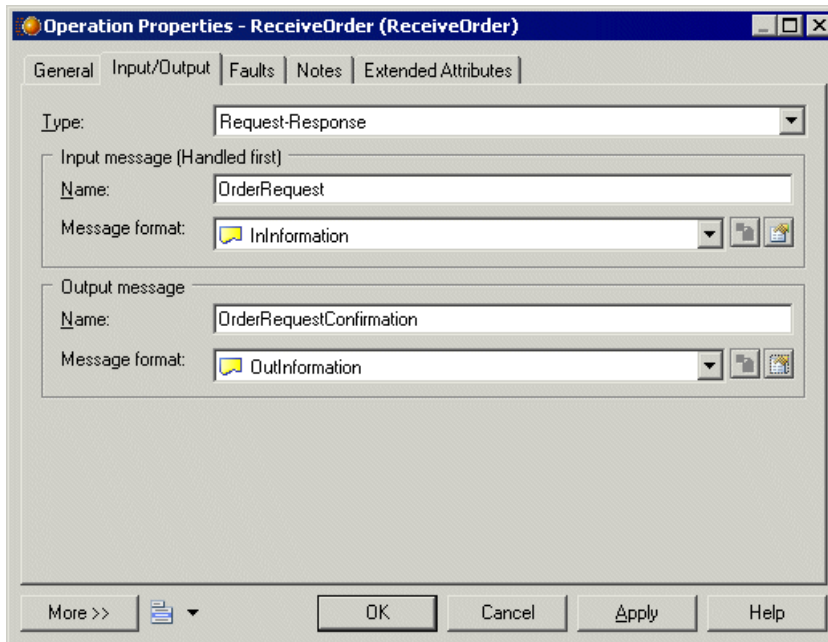
5. Then click Finish to exit the wizard. The operation property sheet opens. Enter ReceiveOrder for the operation name:



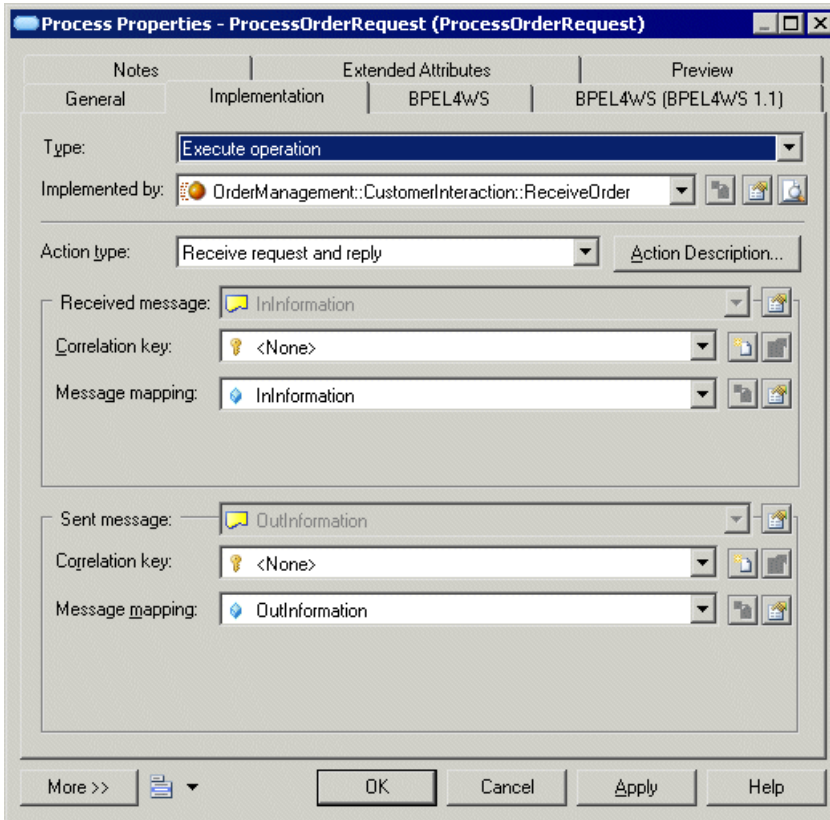
- Click the Input/Output tab and select Request-Response from the Type list, then enter the following information:

Groupbox	Name	Message format
Input message	OrderRequest	InInformation
Outputmessage	OrderRequestConfirmation	OutInformation

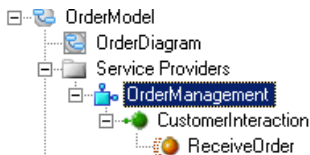
The dialog box displays as follows:



7. Click OK to close the operation property sheet, select Receive request and reply in the Action type list, and then click the New tool to the right of the Message mapping list in the Received message and Sent message groupboxes to create variables for the received and sent messages:



8. Click OK to close the process property sheet. The process symbol displays a small icon to indicate that it is implemented using a web service operation, which displays in the Browser:



Decomposing Processes

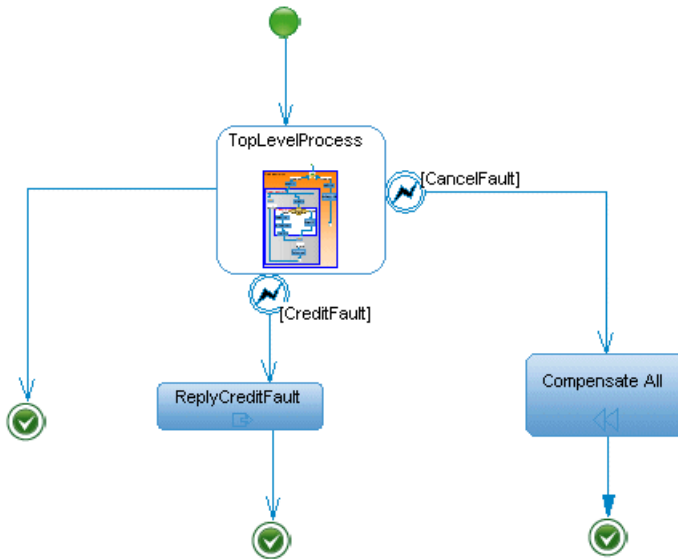
You decompose a process when you want to analyze it in more detail. For example, you could decompose the process "Process order" into the sub-processes "Check availability", "Obtain payment", "Reserve stock items", and so on.

The decomposed process behaves like a specialized package or container. It has its own sub-diagram, which models the control flow or data flow between its sub-processes.

Sub-processes can, themselves, be decomposed into further sub-processes, and so on until you get sufficient detail, or until you have reached the level of atomic tasks.

Decomposed process symbols in a business process diagram carry a plus-sign symbol to indicate that they contain further detail. Alternately, to display the detail available in a sub-diagram, to display the detail available in a sub-diagram, right-click the symbol and select **Composite View > Read-only (Sub-Diagram)**.

In the following example, TopLevelProcess is in composite view mode; the details of its sub-diagram are displayed within its symbol:



Note: To display all processes in the List of Processes, including those belonging to decomposed processes, click the Include Composite Processes tool.

You can create several sub-process diagrams within a decomposed process, but this is not recommended because being in a sub-process diagram means being within the context of a process. Unless you want to design some exception cases like error management for example, it would not be consistent to create multiple sub-process diagrams within a decomposed process.

You cannot create a package or any other business process diagram type in a decomposed process, but you can use shortcuts to packages.

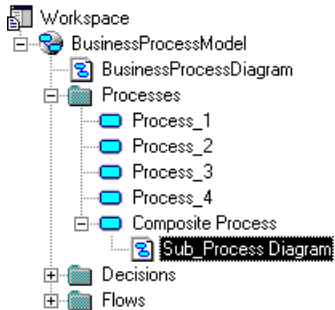
Decomposing an Atomic Process

You can decompose an atomic process in a business process diagram using its icon, context menu or property sheet.

- Press CTRL and double-click the process symbol (this will open the sub- process directly).
- Right-click the process and select Decompose Process in the contextual menu.
- Open the property sheet of the process and, on the General tab, select the Decomposed Process radio button.

For more information about process decomposition in a PHD, see *Chapter 2, Building Process Hierarchy Diagrams* on page 15.

When you create a decomposed process, a sub-process diagram is automatically created below its entry in the browser:



The decomposed process is empty at first, except when you create it from a selection of symbols. Any objects that you create in the sub-process diagram are listed in the Browser under the decomposed process.

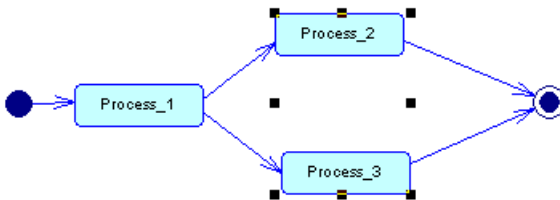
In a BPD, decomposed processes must always have at least one start and one end.

Creating a Decomposed Process from a Selection of Symbols

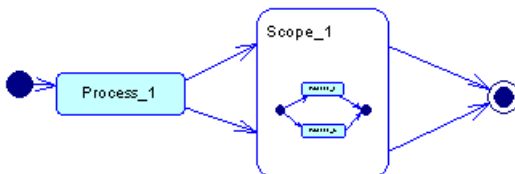
You can create a decomposed process from a selection of symbols in the diagram using the **Tools > Create Composite Process** command. This command is only available from a business process diagram in which you have selected valid objects, such as processes, flows, decisions or synchronizations.

When selected symbols have input/output flows from/to non-selected symbols, a start/end is automatically created inside the decomposed process together with default output/input flows from this start to these selected objects or from these selected objects to the end.

Initial diagram:



New decomposed process:



1. Press the **Shift** key while clicking symbols.
2. Select **Tools > Create Composite Process**.

A new decomposed process named Scope_n by default is created and contains the selected symbols.

The new decomposed process opens in composite view mode (global view of its content within its symbol in the diagram). In addition, any defined symbol custom shapes for are removed, however their relative position, size and format are preserved together with the display preferences of the initial diagram.

Converting a Business Process Diagram to a Decomposed Process

You can convert a diagram to a decomposed process using the Convert Diagram to Process wizard in order to describe the context of a complex process. This option is only available once objects have been created in the diagram. By converting a diagram to a decomposed process, you can then use the decomposed process in another business process diagram.

1. Right-click the diagram background and select **Diagram > Convert to Composite Process**.

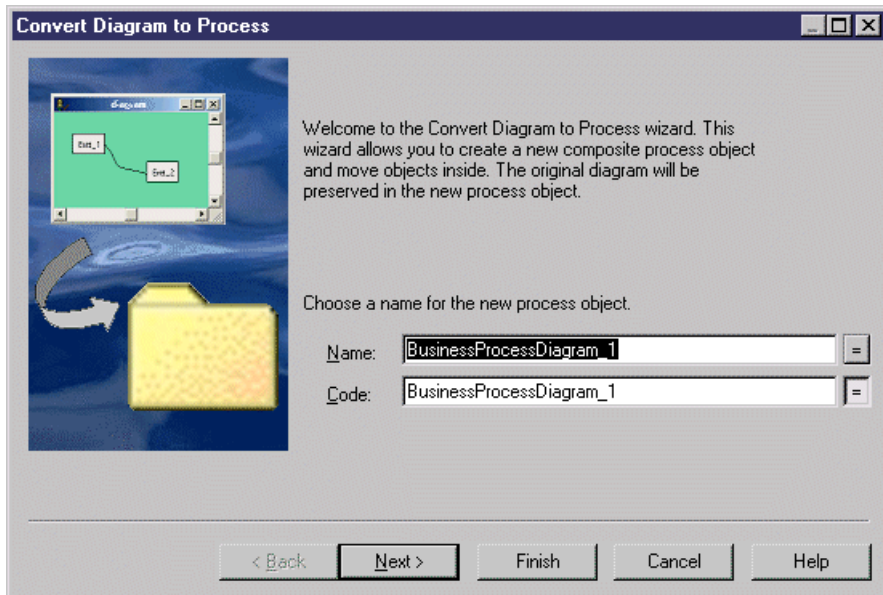
or

Right-click the diagram node in the Browser and select Convert to Composite Process.

or

Select **View > Diagram > Convert to Composite Process** .

The Convert Diagram to Process page is displayed.



2. Specify a name and a code in the Convert Diagram to Process page, and then click Next to open the Selecting Objects to Move page.
3. Select the processes that you want to move to the new decomposed process diagram. Processes that you select will be moved in the Browser to under the new decomposed process. Those that you do not select will remain in their present positions in the Browser and will be represented in the new sub-process diagram as shortcuts.
4. Click Finish to exit the wizard. The new decomposed process and its sub-process diagram will be created, and any objects selected for the move will now appear beneath the decomposed object in the Browser.

Removing a Level of Decomposition

Sometimes, you might decompose a process and then decide that you want, instead, to show the detail of its sub-processes directly.

You can remove this level of decomposition by right-clicking the decomposed process symbol in the diagram or in the Browser and selecting Remove Composite Process Level in the contextual menu.

All the objects contained in the decomposed process (except any starts and ends) are moved one level up in the hierarchy, and those directly beneath the process replace the deleted process in any diagrams. Input and output flows previously connected to the old process are automatically re-linked to the first and last objects respectively from the old control flow.

Working with Sub-Process Diagrams

You can add objects to a sub-process diagram in the same way as you add them to a business process diagram. Each process that you add to a sub-process diagram is a part of its parent decomposed process, and is listed beneath it in the Browser.

You can open the sub-process diagram of a decomposed process in any of the following ways:

- Press **Ctrl** and double-click the decomposed process symbol.
- Double-click the appropriate diagram entry in the Browser.
- Right-click the diagram background and select **Diagram > Go Up One Level** when a sub-process is decomposed into further sub-processes and therefore contains several level of sub-diagrams.

You can move from a sub-process diagram to another diagram in any of the following ways:

- Press **Ctrl+U** to go to the model's default diagram.
- Right-click the process and select Open Diagram in the contextual menu.
- Double-click the sub-process diagram entry under the decomposed process in the Browser.

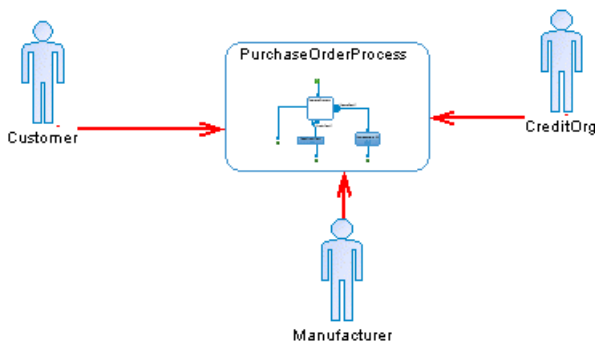
Note: You can set a diagram to be the model's default diagram by selecting the Default Diagram check box in its property sheet, or by selecting the diagram in the Default Diagram list in the model property sheet.

Working with Composite Views

You can change a decomposed process symbol to give a representation of its sub-process diagram. To do so, right-click a decomposed process and select **Composite View > Read-only (Sub-Diagram)**. The decomposed process symbol is expanded. You may need to resize the symbol to see all its content.

Note that if you double click the composite view, you automatically open the sub-process diagram.

In the following example, PurchaseOrderProcess is in composite view mode, and displays the details of its sub-diagram within its symbol:



Working with Data and Resource CRUD Matrices

A *CRUD matrix* is a table that allows you to observe, at a global level, the actions (Create, Read, Update, or Delete) your processes perform on data or resources, and modify or add any missing actions on them. You can open the Resource CRUD Matrix or Data CRUD Matrix using the commands available in the **Tools** menu.

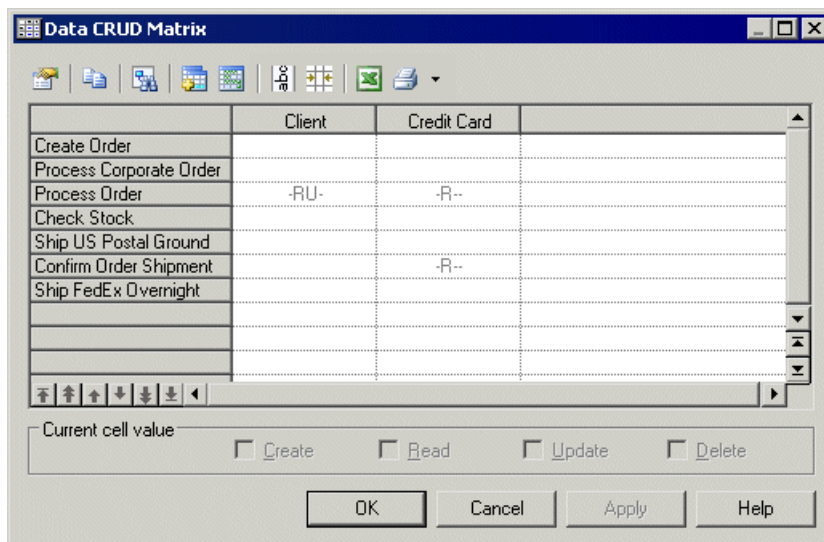
You must have created at least one process and one resource or data object in order to have access to the commands. A CRUD matrix can be created in a choreography diagram with any of the following target languages:

- Analysis languages (except BPMN)
- Collaborative languages

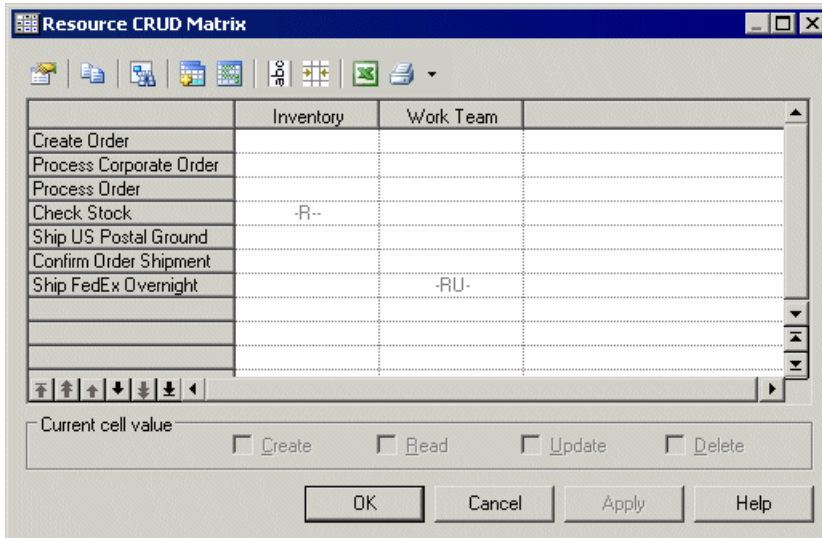
To modify the CRUD values for a process, select the appropriate cell and select or deselect the CRUD check boxes in the Current Cell Value groupbox at the bottom of the matrix window. Changes made in the matrix are reflected in the diagram and property sheets of the affected objects. You cannot select and edit multiple cells.

Note: A process must already be associate with data or resources in order for it to be included in the matrix. You cannot view and modify the CRUD values for objects that are not already related in some way.





In the following example, the Process Order process reads and updates the Client data, and reads the Credit Card data, and the Confirm Order Shipment process reads the Credit Card data:








In the following example, the Check Stock process reads data stored in the Inventory resource and the Ship FedEx Overnight process reads and updates data stored in the Work Team resource:



You can reorder the rows in the matrix by using the arrows at the bottom of the process column. The following tools are available above the matrix:

Tool	Description
	<p>Properties – Opens the property sheet of a:</p> <ul style="list-style-type: none"> • Process, if you select a row header. • Resource or data, if you select a column header. • Resource flow or data, if you select a cell. <p>If parallel resource flows exist between a process and a resource, you will be prompted to choose the resource flow whose properties you want to consult.</p>
	<p>Copy – Copies a CRUD matrix in order to paste it into another application such as:</p> <ul style="list-style-type: none"> • Excel (as CSV). • Word (as text).
	<p>Find Symbol in Diagram – Finds in the diagram the symbol of a:</p> <ul style="list-style-type: none"> • Process, if you select a row header. • Resource, if you select a column header. • Resource flow or process that contains the CRUD values, if you select a cell. <p>If parallel resource flows exist between a process and a resource, you will be prompted to choose the resource flow, whose symbol you want to find in the diagram.</p>
	<p>Select Rows/Columns – Opens a selection box listing all the available objects, which allows you to add or remove rows and columns.</p>

Tool	Description
	Display Only Non-Empty Rows/Columns – Displays only objects sharing a relationship or shows all available objects.
	Vertical/Horizontal Column Header - Toggles between vertical and horizontal orientation of column headers.
	Shrink to Fit - Shrinks row and column headers to fit their contents.
	Export to Excel - Exports the matrix as an MS Excel file. If the specified file already exists, you will be given the option to overwrite it or append a new worksheet in the file.
	Print - Prints the matrix. Click the arrow to the right of the button to view a print preview or to access the Page Setup dialog.

Organization Units (BPM)

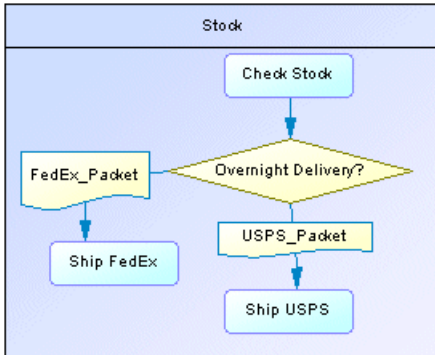
An *organization unit* can represent a company, a system, a service, an organization, a user or a role, which is responsible for a process. It can also be a business partner who uses high level processes.

Note: To enable the display of organization unit swimlanes, select **Tools > Display Preferences**, and select the **Organization unit swimlane** checkbox on the **General** page, or right-click in the diagram background and select Enable Swimlane Mode.

Choreography Diagram

An organization unit can be created in a choreography diagram with any target language. It allows you to assign responsibilities within your system, and displays as a swimlane, which can contain all the symbols of a choreography diagram.

In the following diagram, the Stock organization unit is responsible for the Check Stock process and of the Ship FedEx and Ship USPS processes. The management of these latter processes depends on whether the delivery must be overnight or not:



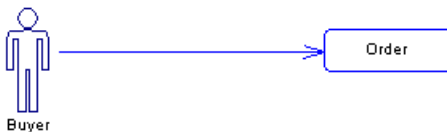
Top-level Diagram

An organization unit can be created in a top-level diagram with any of the following target languages:

- Orchestration languages (BPEL4WS and WS-BPEL only)
- Collaborative language

An organization unit allows you to identify external partners, which interact with your system, and displays as an actor.

In the following example, the Buyer organization unit interacts with the Order process:



Data Flow Diagram

An organization unit can be created in a data flow diagram and displays as an actor to design an external entity that sends or receives data from the system.

Creating an Organization Unit

Create an organization unit to show the participant responsible for the execution of processes.

- Use the **Organization Unit Swimlane** tool in the Toolbox.
- Select **Model > Organization Units** to access the List of Organization Units, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Organization Unit**.

Depending on your diagram, the organization unit will either be displayed as a swimlane or as an actor.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Creating Organization Units with the Swimlane Tool

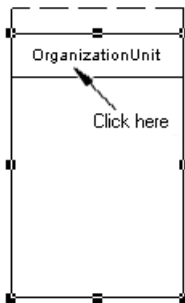
Use the Organization Unit Swimlane tool in the Toolbox to quickly create organization unit swimlanes.

Click in or next to an existing swimlane or pool of swimlanes to add a swimlane to the pool.

Click away from existing swimlanes to create a new pool.

Organization Unit Properties

To view or edit an organization unit's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.



The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.

Property	Description
Stereotype	<p>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</p> <p>An organization unit has the following predefined stereotypes:</p> <ul style="list-style-type: none"> • Role – specifies a role a user plays • User • Group – specifies a group of users • Company • Organization – specifies an organization as a whole • Division – specifies a division in a global structure • Service – specifies a service in a global structure
Parent organization	<p>Specifies another organization unit as the parent to this one.</p> <p>For example, you may want to describe an organizational hierarchy between a department Dpt1 and a department manager DptMgr1 with DptMgr1 as the parent organization of Dpt1.</p> <p>The relationship between parent and child organization units can be used to group swimlanes having the same parent. For more information, see <i>Grouping and Ungrouping Swimlanes</i> on page 58.</p>
Keywords	<p>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</p>

Attaching Processes to Organization Units

Attach processes to organization units to graphically assign responsibility for them. When processes are attached to an organization unit displayed in a swimlane, the organization unit name is displayed in the Organization Unit list of their property sheets.

You attach processes to an organization unit by creating them in (or moving existing ones into) the required swimlane. Alternately, you can select an organization unit name from the Organization Unit list of the process property sheet, and click OK to attach it.

To detach processes from an organization unit, drag them outside the swimlane or select <None> in the process property sheet.

Displaying a Committee Process

A committee process is a decomposed process whose sub-processes are managed by several organization units.

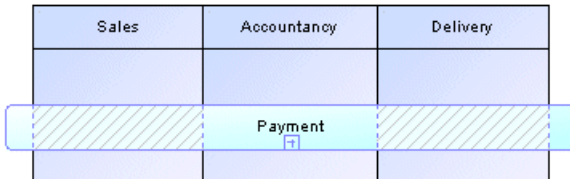
1. Open the property sheet of a decomposed process.
2. Select **Committee Process** from the Organization Unit list and click **OK**.

This value is only available for decomposed processes.

3. In the diagram, resize the decomposed process symbol to cover all the appropriate swimlanes.

The symbol background color changes on the swimlanes depending on whether each is responsible for sub-processes.

In the following example, all sub-processes of Payment are managed in the Accountancy organization unit:



The symbol background of the committee process is lighter and hatched on Sales and Delivery since they do not:

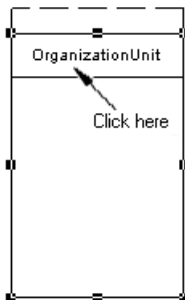
- Manage any sub-processes
- Have any symbol in the sub-process diagram

Note that this display does not appear in composite view mode.

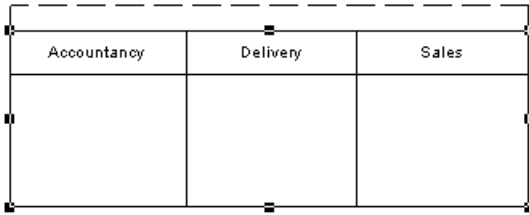
Managing Swimlanes and Pools

Each group of one or more swimlanes forms a pool. You can create multiple pools in a diagram, and each pool is generally used to represent a separate organization.

To select an individual swimlane in a pool, click its header:



To select a pool, click any of its swimlanes or position the cursor above the pool, until you see a vertical arrow pointing to the frame, then click to display the selection frame:



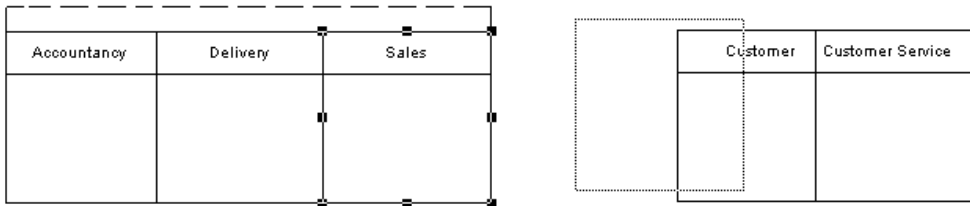
Moving, Copying and Pasting Swimlanes

You can move, copy, and paste swimlanes and pools in the same or in another diagram.

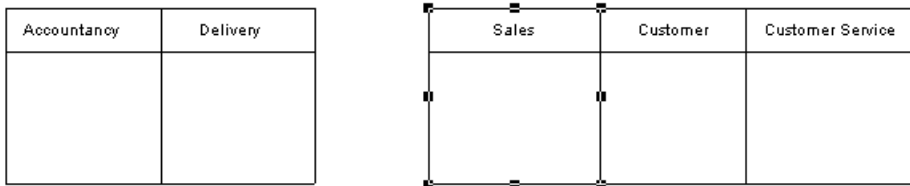
Diagram	What happens...
Same	When you move a swimlane or pool within the same diagram, all symbols inside the swimlane(s) are moved at the same time (even if some elements are not formally attached), so as to preserve the layout of the diagram.
Different	When you move or copy a swimlane or pool to another folder or diagram, the symbols inside the swimlane(s) are not copied.

If a swimlane is dropped on or near another swimlane or pool, it joins the pool.

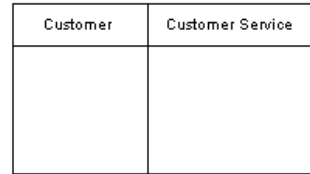
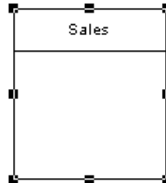
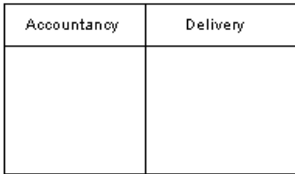
In the following example, Sales forms a pool with Accountancy and Delivery, and is moved to another pool containing Customer and Customer Service:



After the move, Sales has moved from its original pool, and joined the pool containing Customer and Customer Service:



If the moved swimlane is dropped away from another swimlane or pool, it forms a new pool by itself, as in the following example:



If you move linked objects inside a swimlane, the width or height of the swimlane varies with them.

Note: The auto-layout function is unavailable with organization units displayed as swimlanes.

Grouping and Ungrouping Swimlanes

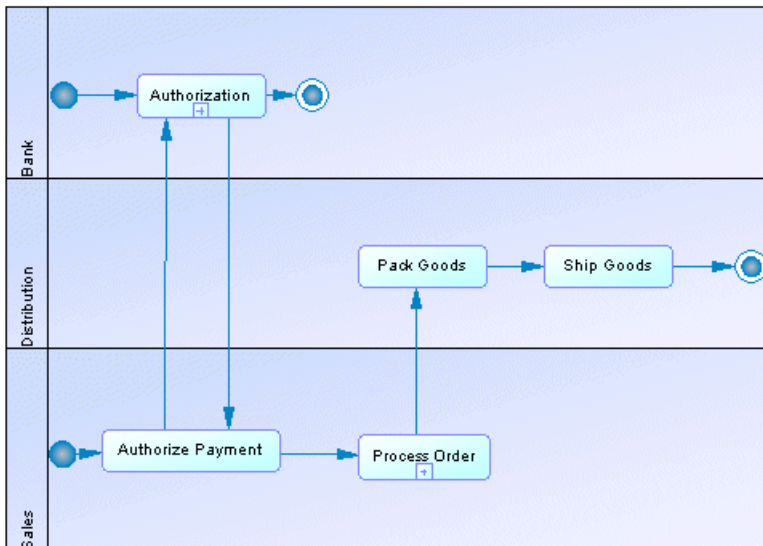
Group organization unit swimlanes within a pool to organize them under a common parent or user-defined name.

To group swimlanes within a pool, right-click the pool, select **Swimlane Group Type**, and then select one of the following options:

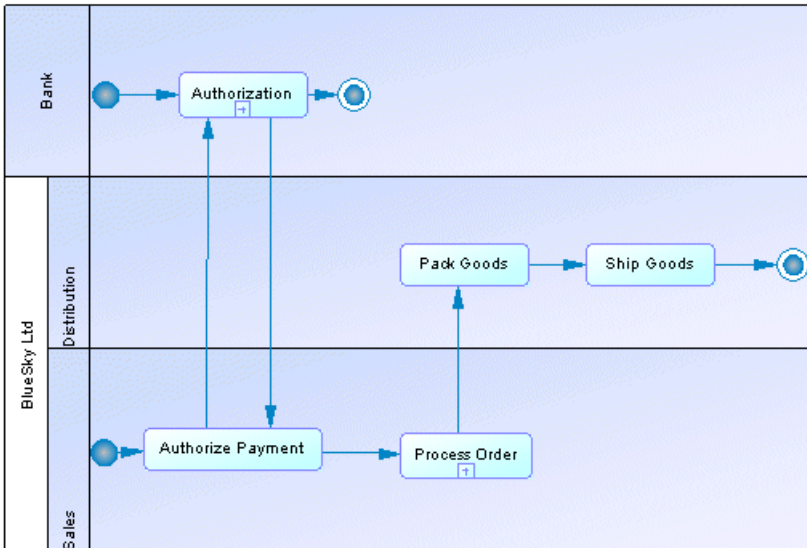
- **By Parent** - to assign the name of the last common parent for the group. This name comes from the Parent Organization Unit field in the swimlanes property sheet.
- **User-Defined** - to assign a name of your choice for the group. Then, you must select at least two attached swimlanes, and select **Symbol > Group Symbols** from the menu bar to display a default name that you can modify.

To ungroup swimlanes, select **Ungroup Symbols** from the pool contextual menu or **Select Symbol > Ungroup Symbols**.

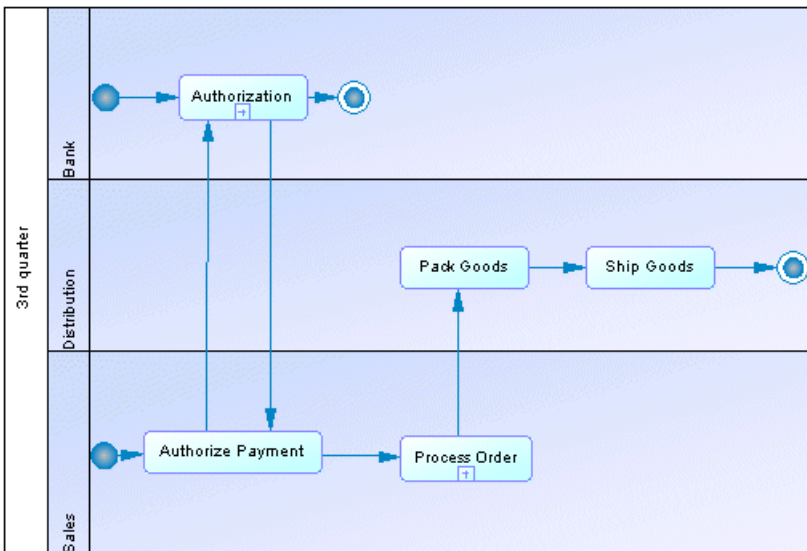
The following example shows a pool without any group:



In the following example, Sales and Distribution are grouped by their BlueSky Ltd common parent:



In the following example, the pool is assigned a user-defined group named 3rd quarter:

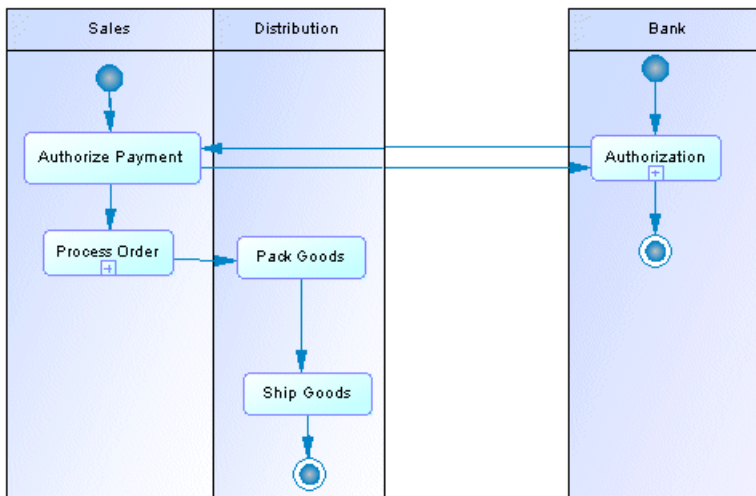


Creating Links Between Pools of Swimlanes

Create links between pools or between processes in separated pools to represent interactions between them.

To create links between pools of swimlanes, simply click the **Flow** tool in the Toolbox and drag a flow from one process in a pool to another in a different pool or from one pool to another.

In the following example, flows pass between Authorize Payment in the Sales swimlane in one pool and Authorization in the Bank swimlane in another pool:



Note: Such links between processes in separate pools are not visible when the swimlanes are not in composite view mode.

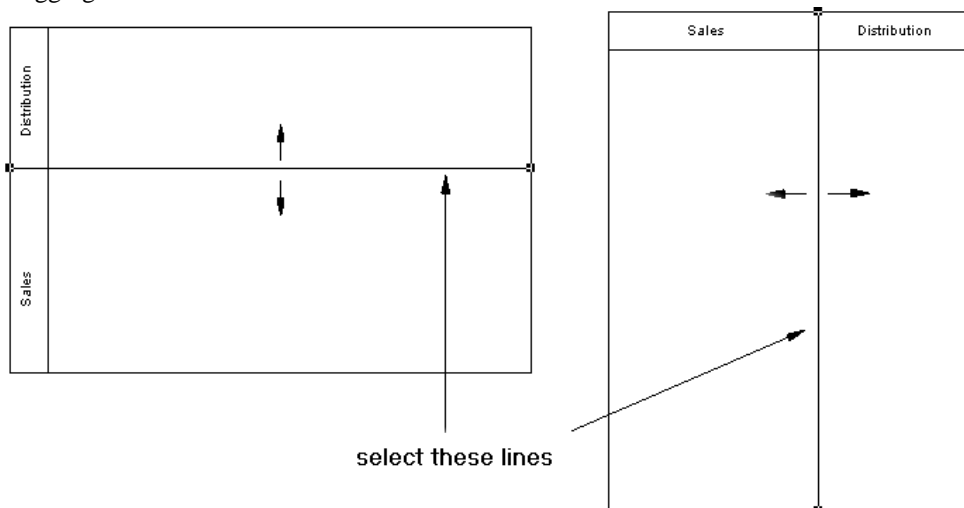
Changing the Orientation of Swimlanes

You can change the orientation of swimlanes so that they run vertically (from top to bottom) or horizontally (from left to right). All swimlanes in a diagram must have the same orientation.

1. Select **Tools > Display Preferences** to open the Display Preferences dialog box.
2. Select the appropriate radio button in the Organization unit swimlane groupbox, and click **OK**.

Resizing Swimlanes

You can resize swimlanes within a pool by clicking the dividing line between them and dragging it.



When you change the width or height of an individual swimlane, all process symbols attached to the swimlane keep their position.

You can resize a pool by selecting one of the handles around the pool, and dragging it into any direction. Any other pools your diagram may contain may also be resized to preserve the diagram layout.

Changing the Format of a Swimlane

You can change the format of swimlanes or a pool using the Symbol Format dialog box. Format changes apply to all swimlanes in the pool.

By default, you use organization units displayed as:

- Actors – in top-level diagrams
- Swimlanes – in choreography diagrams

If you want to modify this default behavior, you can:

- Select **Tools > Display Preferences > General**, and select or deselect the **Organization Unit Swimlane** check box.
- Right-click the diagram background, and select **Enable Swimlane Mode** or **Disable Swimlane Mode**

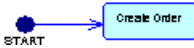
When you switch organization unit representations, the following occurs in the diagram:

Mode	Result
Swimlane to actor	The swimlane is deleted in the diagram, but the corresponding organization unit still exists in the Browser. Right-click the diagram background, select Show Symbols , and select the check box corresponding to the organization unit to display it. Process attachments to the former swimlane still exist in its property sheet.
Actor to swimlane	The actor is deleted in the diagram and replaced with a swimlane if processes were attached to it, otherwise you must display the swimlane using the Show Symbols dialog box.

1. Right-click the swimlane or a pool, and select **Format** to display the Symbol Format dialog box.
2. Enter or select changes in the different tabs, and click **OK** to return to the diagram.

Starts (BPM)

A *start* is a starting point of the flow represented in the diagram.



A start can be created in a choreography diagram with any of the following target languages:

- Analysis languages (except DFD)
- Orchestration languages
- Collaborative languages

In decomposed processes, only one start is allowed per diagram, except for analysis business process diagrams. The Start tool is unavailable if a start symbol already exists. You should not use the same start in two diagrams, and you cannot create shortcuts of starts.

- Activity Diagram - one or more per diagram
- Statechart Diagram - one or more per diagram
- Interaction Overview Diagram - one only per diagram

Note: The start is compared and merged in the Merge Model feature, which checks that there is no additional start in decomposed processes .

Creating a Start

You can create a start from the Toolbox, Browser, or **Model** menu.

- Use the **Start** tool in the Toolbox.
- Select **Model > Starts** to access the List of Starts, and click the **Add a Row** tool.

- Right-click the model (or a package) in the Browser, and select **New > Start**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Start Properties

To view or edit a start's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

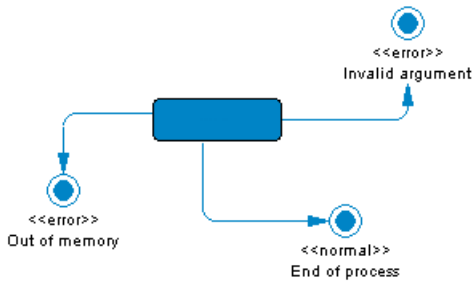
Ends (BPM)

An *end* is a termination point of the flow represented in the diagram.

An end can be created in a choreography diagram with any of the following target languages:

- Analysis languages (except DFD)
- Orchestration languages
- Collaborative languages

You can create several ends within the same diagram, if you want to show divergent end cases, like errors scenarios:



If there is no end, the diagram contains an endless process. However, a decomposed process must always contain at least one end.

You should not use the same end in more than one diagram. You are not allowed to use shortcuts of ends, but graphical synonyms are permitted.

Creating an End

You can create an end from the Toolbox, Browser, or **Model** menu.

- Use the End tool in the Toolbox.
- Select **Model > Ends** to access the List of Ends, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > End**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

End Properties

To view or edit an end's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

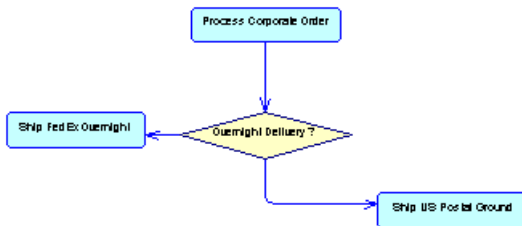
The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Property	Description
Type	Specifies the type of the end used for document purposes. You can create your own type of end in the Type list, or choose one of the following values: <ul style="list-style-type: none"> • Success • Timeout • Business error • Technical error • Compensation
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Decisions (BPM)

A *decision* specifies which path to take, when several paths are possible.



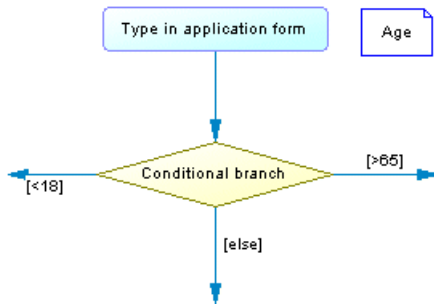
A decision can be created in a choreography diagram with any of the following target languages:

- Analysis languages (except DFD)
- Orchestration languages
- Collaborative languages

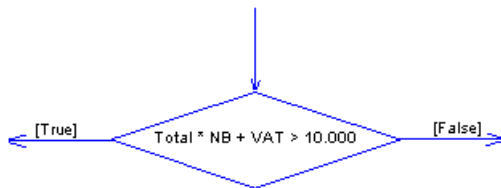
A decision can have one or more input flows and one or more output flows, each labeled with a distinct *guard condition*, a condition that must be satisfied for an associated flow to execute some action. Your guard conditions should avoid ambiguity by not overlapping, yet should also cover all possibilities in order to avoid process freeze.

A decision can represent:

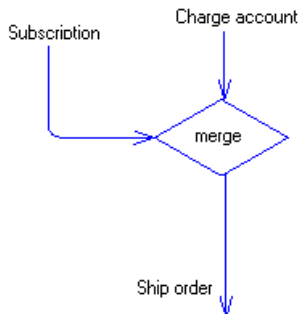
- A conditional branch: one input flow and several output flows. In the following example, the control flow passes to the left if the age given in the application form is <18, to the right if the age is >65, and takes the another route if the age is not mentioned:



You can display a condition on the decision symbol in order to factorize the conditions attached to the flows. In the following example, the condition $Total * NB + VAT > 10.000$ is entered in the Condition tab in the decision property sheet, and True and False are entered in the Condition tabs of the flows:



- A merge: several input flows and one output flow. In the following example, the Subscription and Charge account flows merge to become the Ship order flow:



A decision allows you to create complex flows, such as:

- if ... then ... else ...
- switch ... case ...
- do ... while ...
- loop
- for ... next ...

Note: It is not possible to attach two flows of opposite directions to the same corner on a decision symbol.

Creating a Decision

You can create a decision from the Toolbox, Browser, or **Model** menu.

- Use the **Decision** tool in the Toolbox
- Select **Model > Decisions** to access the List of Decisions, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Decision**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Decision Properties

To view or edit a decision's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Decision Property Sheet Condition Tab

The **Condition** tab contains the following properties:

Property	Description
Alias	Specifies a short name for the condition, to be displayed next to its symbol in the diagram.
Condition (text box)	Specifies a condition to be evaluated to determine how the decision should be traversed. You can enter any appropriate information in this box, as well as open, insert and save text files. You can open the Condition tab by right-clicking the decision symbol, and selecting Condition in the contextual menu.

Synchronizations (BPM)

A *synchronization* enables the splitting or synchronization of control between two or more concurrent actions.

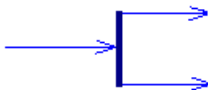
A synchronization can be created in a choreography diagram with any of the following target languages:

- Analysis languages (except BPMN)
- Orchestration languages
- Collaborative languages
- Data Flow Diagram - to design a split/merge.

Synchronizations are represented as horizontal or vertical lines. You can change the orientation of the symbol by right-clicking it and selecting Change to Vertical or Change to Horizontal from the contextual menu.

A synchronization can be either a:

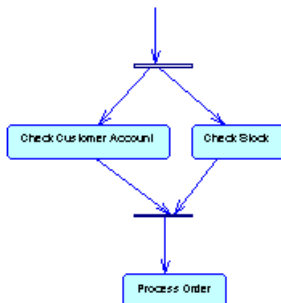
- Fork - Splits a single input flow into several independent output flows executed in parallel:



- Join – Merges multiple input flows into a single output flow. All input flows must reach the join before the single output flow continues:



In the following example, the flow coming into the first synchronization is split into two separate flows, which pass through Check Customer Account and Check Stock. Then both flows are merged into a second synchronization giving a single flow, which leads to Process Order:



Creating a Synchronization

You can create a synchronization from the Toolbox, Browser, or **Model** menu.

- Use the Synchronization tool in the Toolbox.
- Select **Model > Synchronizations** to access the List of Synchronizations, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Synchronization**.

By default, the synchronization symbol is created horizontally. To toggle between horizontal and vertical display, right-click the symbol and select **Change to Vertical** or **Change to Horizontal** in the contextual menu.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Synchronization Properties

To view or edit a synchronization's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Synchronization Property Sheet Action Tab

The **Action** tab contains the following properties:

Property	Description
Action (text box)	Specifies an action to be evaluated to determine how the synchronization should be traversed. You can enter any appropriate information in this box, as well as open, insert and save text files.

Property	Description
Timeout	Specifies the timeout limit. The default value is zero. When the value is not set to zero, it means that a timeout exception occurs if the execution of the activation takes more than the specified timeout limit. You can type any alphanumeric value in the Timeout box (Example: 20 seconds).

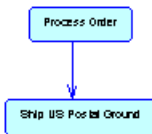
Flows (BPM)

A *flow* is a route the control flow takes between objects (potentially with the exchange of data). The routing of the control flow is made using guard conditions defined on flows. If the condition is true, the control is passed to the next object.

A flow can be created in the following diagrams:

- Choreography diagram with any target languages
- Data flow diagram

In the following example the flow links Process Order to Ship US Postal Ground:



In all languages that support message formats, except orchestration languages, you can associate a message format with a flow in order to define the format of information exchanged between objects. In orchestration languages, the message format is used to specify the format of the message associated with an operation .

A flow can link shortcuts. A flow accepts shortcuts on both extremities to prevent it from being automatically moved when a process is to be moved. In this case, the process is moved and leaves a shortcut, but contrary to the other links, the flow is not moved. Shortcuts of flows do not exist, and flows remain in place in all cases.

The following rules apply:

- *Reflexive flows* (same source and destination process) are allowed on processes.
- Two flows between the same source and destination objects are permitted, and called *parallel flows*.

Note: When flows are compared and merged by the Merge Model feature, they are matched by trigger event first, and then by their calculated name. When two flows match, the trigger actions automatically match because there cannot be more than one trigger action.

Creating a Flow

You can create a flow from the Toolbox, Browser, or **Model** menu.

- Use the **Flow/Resource Flow** tool in the Toolbox
- Select **Model > Flows** to access the List of Flows, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Flow**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Flow Properties

To view or edit a flow's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name and code are read-only. You can optionally add a comment to provide more detailed information about the object.
Source	Specifies the object that the flow leads from. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object. You can also open the property sheet of the source object by clicking the Source button located at the top part of the current object property sheet.
Destination	Specifies the object that the flow leads to. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object. You can also open the property sheet of the destination object by clicking the Destination button located at the top part of the current object property sheet.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Transport	Specifies the way data is conveyed by the flow. This property is used for documentation purposes. You can create your own type of transport in the list, or choose one of the following values: <ul style="list-style-type: none"> • Fax delivery • Mail • Telephone

Property	Description
Flow type	<p>Specifies the type of the flow. You can create your own type of flow in the list, or choose one of the following values:</p> <ul style="list-style-type: none"> • Success - defines a successful flow • Timeout - defines the occurrence of a timeout limit • Technical error • Business error • Compensation - defines a compensation flow <p>The flow type is unavailable if you associate an event with the flow on the Condition tab.</p>
Message format	<p>[Not available for orchestration languages]. Specifies the format of the data exchanged between processes. You can choose from one of the following values:</p> <ul style="list-style-type: none"> • None – defines a simple flow with no exchange of data • Undefined (default value) – defines a flow whose message format is not yet defined. Click the Create tool to the right of the Message Format list to create a message format for your flow.

Note: You can view input and output flows of a process from its property sheet by clicking the Input Flows and Output Flows sub-tabs of the Dependencies tab.

Flow Property Sheet Condition Tab

Parameter	Description
Alias	Short name for the condition, to be displayed next to its symbol in the diagram.
Event	[Only available for BPMN and orchestration languages]. Specifies an event to create an event handler. Select an event from the list or use the tools to the right of the list to create an object, or to view the properties of the currently selected object.
Variable	[Only available for events with a fault stereotype]. Specifies a local variable that retrieves the information associated with the fault event. Select a variable from the list or use the tools to the right of the list to create an object, or view the properties of the currently selected object.
Condition (text box)	Specifies a condition to be evaluated to determine how the flow should be traversed. You can enter any appropriate information in this box, as well as open, insert and save text files. You can open the Condition tab by right-clicking the flow symbol, and selecting Condition in the contextual menu.

Flow Property Sheet Data Tab

The **Data** tab displays the list of data conveyed by the flow without any information on its format. You can add or create data using the Add Objects and Create an Object tools .

You can also migrate data to a source or destination process .

In an Analysis business process diagram, if you have specified data for a message format, the data contained in the flow Data tab should be a sub-set of the data contained in the message format Data tab.

Role Associations (BPM)

A *role association* is a relationship that describes an interaction between a process and an organization unit displayed as an actor.

A role association can be created in a top-level diagram (see *Top-Level Diagram Basics* on page 22) with any of the following target languages:

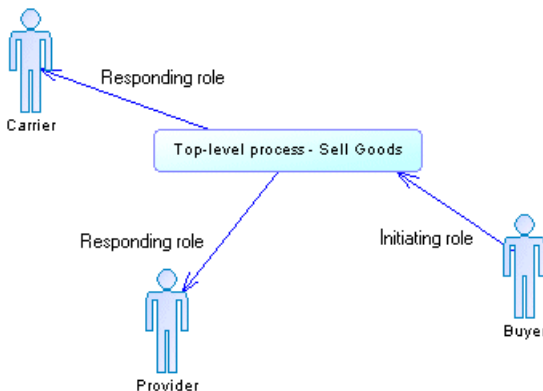
- Orchestration languages (BPEL4WS and WS-BPEL only)
- Collaborative languages

A role association can have the following roles:

- Initiating role – drawn from an organization unit to a process
- Responding role – drawn from the process to the organization unit

If the orientation of the role association is not displayed, you can modify the role association display preferences, by selecting **Tools > Display Preferences > Role Association**.

In the following example, the Buyer interacts with the top-level process via an initiating role, and the top-level process interacts with the Carrier and the Provider via responding roles:



Note: Role associations are not allowed in decomposed processes of top-level diagrams associated with BPEL languages or ebXML. For more information (see *Top-Level Diagram Basics* on page 22).

Creating a Role Association

You can create a role association from the Toolbox, Browser, or **Model** menu.

- Use the **Role Association** tool in the Toolbox.
- Select **Model > Role Associations** to access the List of Role Associations, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Role Association**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Role Association Properties

To view or edit a role association's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Orientation	Specifies the direction of the role association, provided the Direction option is selected in the display preferences. You can choose between: <ul style="list-style-type: none">• Initiating role – the role association leads from the organization unit to the process• Responding role – the role association leads from the process to the organization unit

Property	Description
Source	Specifies the object that the role association leads from. It can be an organization unit or a process. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object. You can also open the property sheet of the source object by clicking the Source button at the top of the tab.
Destination	Specifies the object that the role association leads to. It can be an organization unit or a process. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object. You can also open the property sheet of the destination object by clicking the Destination button at the top of the tab.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Events (BPM)

An *event* is an instantaneous and observable occurrence during the course of a business process, which triggers it to respond. For example, it can be unexpected data returned by a web service or a deadline.

An event can be created in a choreography diagram with any of the following target languages:

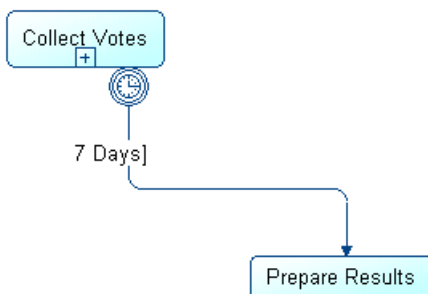
- BPMN language
- Orchestration languages

An event can be associated with a:

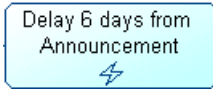
- Flow – to be caught and handled using an event handler (see *Event handlers* on page 77).
- Process with a Generate event implementation type – to trigger an event (see *Specifying implementation types* on page 37).

The same event can be shared between several flows and processes. An event is reusable by nature because it is not dependent on the context.

In the following example, the output flow of the Collect Votes process will be fired after 7 days:



In the following example, the small symbol at the bottom of the Delay 6 days from Announcement process shows that the process triggers a fault type event:



Creating an Event

You can create an event from a flow or process property sheet or from the Browser or **Model** menu.

- Select **Model > Events** to access the List of Events, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Event**.
- Double-click a flow or a process to open its property sheet, click the **Condition** tab (flow) or the **Implementation** tab (process), and then click the **Create** tool to the right of the Event box.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Event Properties

To view or edit an event's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.







The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.

Property	Description
Stereotype	<p>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</p> <p>An event has the following predefined stereotypes:</p> <ul style="list-style-type: none"> • Fault – specifies the occurrence of an error in the normal execution of the process. • Timer – specifies a time event and needs to specify a duration (for example, 1 hour) or a deadline (for example, each Sunday). You can specify a timer value in the Expression box. • Compensation – specifies the invocation of a process compensation that allows you to cancel the actions performed by an already terminated process.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Predefined Events

Whether it is used in association with flows or processes, the event type symbol displays on the flow or on the process as follows:

Event type	Symbol on flow	Symbol on process
Timer		
Fault		
Compensation		

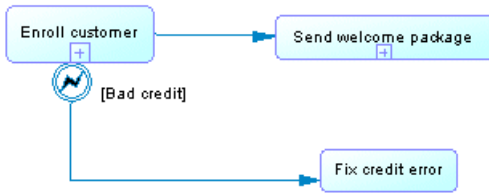
When used in association with a process with a Generate event implementation type, a Timer event implements a Wait activity, a Fault event implements a Throw activity, and a Compensation event implements a Compensate activity (see *Process Properties* on page 34).

Event Handlers

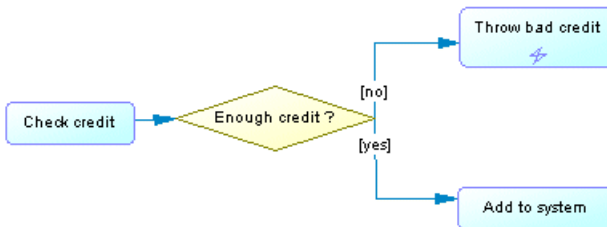
An *event handler* is a way to catch an event, and to handle it using a business process.

You create an event handler by dragging a flow from a source process to a target process, and then associate an event with the flow. The target process specifies the handling of the event and the event type symbol is displayed at the source of the flow.

In the following example, when the Enroll customer composite process completes normally, the flow goes to the Send welcome package composite process. But if an event occurs during its execution, the Enroll customer composite process catches the event and passes control to the Fix credit error process, which acts as a fault handler to its parent process:



We can see that the Throw bad credit event breaks the normal flow of the Enroll customer parent process:



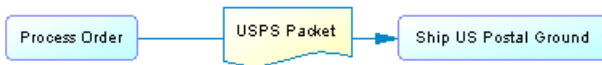
1. Open the flow property sheet and click the Condition tab.
2. Select an event from the Event list and click OK.

You can visualize in the Dependencies tab of the event property sheet, the list of flows that use the event as an event handler and the list of activities that trigger the event.

Message Formats (BPM)

A *message format* can be an XML document or a list of parameters, which defines the format of the data exchanged between processes.

In the following example, the USPS Packet message format associated with the flow, between the Process Order process and the Ship US Postal Ground, defines how to process a package to be shipped through the United States Postal Service, using standard Ground shipment:

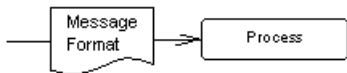


A message format can be created in a choreography diagram with any of the following target languages:

- Analysis languages (Analysis only)
- Collaborative languages
- Orchestration languages

Message Formats in Analysis and Collaborative Languages

When working with the Analysis language and ebXML language, you can associate a message format with a flow or resource flow in order to define the format of information exchanged between processes. The message format is displayed on the flow that uses it:



Message Formats in Orchestration Languages

When working with the orchestration languages, a message format is used to specify the format of the message associated with an operation (see *Operations (BPM)* on page 111).

In some cases, it may be appropriate to decompose a message format into message parts that specify its contents (see *Message Parts (BPM)* on page 81).

Creating a Message Format

You can create a message format from a flow property sheet or from the Browser or **Model** menu.

- Click the **Create** tool next to the Message Format list located at the bottom part of the flow property sheet.
- Select **Model > Message Formats** to access the List of Message Formats, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Message format**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Message Format Properties

To view or edit a message format's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Definition Tab

The **Definition** tab contains the following properties:

Property	Description
Definition type	Specifies how the message format is defined. You can choose between: <ul style="list-style-type: none"> • Embedded file – Enter the definition in the text field. You can open, insert and save text files in this field. • External file – Enter a file in the External definition box. • URL – Enter a Web address in the External definition box. • Message parts – Create message parts in the list (see <i>Message Parts (BPM)</i> on page 81). • XML model - Select an XSM open in the workspace. Use the tools to the right of this field to create a new XSM or open the property sheet of the currently selected model. For detailed information about working with XSMs, see <i>XML Modeling</i>.
External definition	[External file and URL only] Specifies the location path to an external file or an URL.
Message format	[Embedded or External file and URL only] Specifies the format of the message. You can enter your own format or choose one of the following: <ul style="list-style-type: none"> • XML Schema • DTD • RELAX NG

Data Tab

This tab is only available for the Analysis language, and lists the data associated with the message format. You can add or create data and specify both the type and the format of the data conveyed by the flow (see *Specifying data for a flow, a resource flow or a message format* on page 91). If you have specified data for a flow, the data specified for the message format, should be a subset of the flow data (see *Flow Properties* on page 71).

Dependencies Tab

The **Dependencies** tab displays the use of an object or model by the current object. When working with orchestration languages, the Dependencies tab allows you to display, in distinct sub-tabs, the different uses of the current message format:

Sub-tab	Description
Operation Input Message	Displays all the operations that use the message format as input.
Operation Output Message	Displays all the operations that use the message format as output.
Fault Message Links	Displays all uses of the message format as a fault on an operation.
Typed Variables	Displays all the variables that use the message format as a data type.

Message Parts (BPM)

A *message part* represents a sub-division of a message format into separate parts. For example, an invoice form can be modeled as a message format, which includes the following message parts: product information, customer information, and payment information.

A message part can be created in a choreography diagram with any of the following target languages:

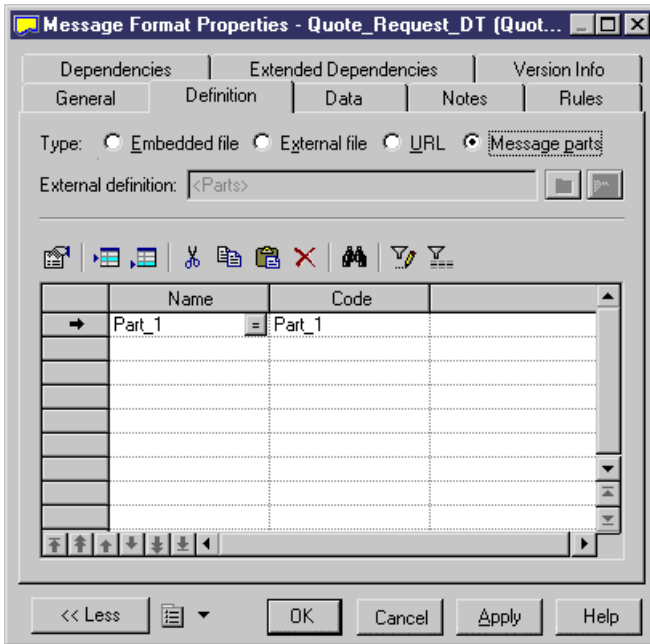
- Analysis languages (Analysis only)
- Collaborative languages
- Orchestration languages

In most languages, a message part lets you describe the message format in a simple way. In orchestration languages, it represents a portion of the WSDL (Web Services Description Language) message.

Creating a Message Part

You create a message part from the message format property sheet.

1. Open the message format property sheet, click the Definition tab, and select the Message Parts radio button in the upper part of the dialog box.
2. Click the Add a Row tool.



3. Type a name and a code for the message part, and click Apply to commit the creation of the new message part.
4. [Optional] Double-click the new message part to specify its properties.
5. Click OK in each of the dialog boxes to return to the model.

Message Part Properties

To view or edit a message part's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Parent	[read-only] Specifies the parent message format.

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Data type	Specifies the data type of the message part. You can choose from a list of simple data types or click the Select Object tool next to the list to select an XML element, a simple or a complex type from the XML models attached to a service provider via an XSD document.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Element type	Specifies whether the variable is an XSD element type. If you have defined a complex type (XSD element) in the Data type list, you should select this check box for the complex type element to be generated. The value of the data type is the name of the element prefixed by the namespace.

Data (BPM)

A *data* object is a piece of information exchanged, at a high conceptual level, between processes using flows or between processes and resources using resource flows.

A data object can be created in a choreography diagram with any Analysis language (except BPMN).

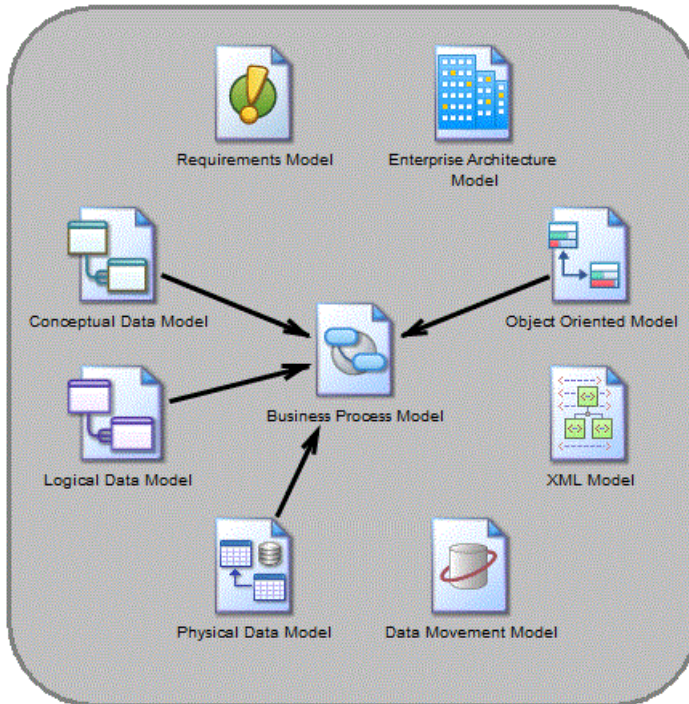
Data objects can be used in conjunction with a:

- Flow or resource flow – to identify the type of data exchanged between processes or between a process and a resource.
- Message format – to identify the type and the format of data exchanged between a resource and a process or between processes.
- Process – to identify the type of action (Create, Read, Update, and Delete) the process performs on the data required for its execution.

You can specify a type for data and decompose them into sub-data. The same data can be shared by several flows, message formats, or processes, but only once each.

Data has no graphical symbol, but you can display a list of data on the flow and resource flow symbols by selecting **Tools > Display Preferences**, clicking **Flow** in the Category list and selecting Data List in the **Center** group box.

You can link data to an object in a CDM, LDM, PDM, or OOM in order to model in detail the nature of the piece of information exchanged (see *Linking Data with Other PowerDesigner Model Objects* on page 85):



Creating Data

You can create data from the Browser or **Model** menu.

- Select **Model > Data** to access the List of Data, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Data**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Data Properties

To view or edit a data's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Type	Specifies the type of the data. The following types are available: <ul style="list-style-type: none"> • Undefined – [default]. • Elementary data – atomic pieces of data, which are comparable to entity attributes, table columns, or class attributes such as a date, name, or ID. • Structured data – more complex data that can be decomposed into sub-data, which are comparable to entities, tables, or classes, such as user, customer, or product.
Definition	Specifies the external PowerDesigner model object that models the data in more detail. Use the tools to the right of the list to browse the complete tree of available objects or view the properties of the selected object. For more information, see <i>Linking Data to an External Model Object</i> on page 86.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Sub-Data Tab

The Sub-Data tab is available only for data objects with the structured data type, and lists the data objects into which the structured data object is decomposed. You can add or create sub-data using the **Add Objects** and **Create an Object** tools.

Note: You can click the Dependencies tab of a sub-data property sheet to display the data of which it is a part.

Linking Data with Other PowerDesigner Model Objects

You can model BPM data objects in more detail by linking them to data items, entities, tables or classes in other PowerDesigner models.

You can link BPM data objects to objects in a CDM, LDM, PDM, or OOM. You can also export BPM data objects to or import data from these other model types. In each case, the BPM

data object and the other model object remain synchronized with the external object displayed in the **Definition** field in the data object property sheet and the data object listed on the **Dependencies** tab of the other model object.

The following table lists the types of objects to which BPM data objects can be linked or exported:

Data Type	CDM	LDM	PDM	OOM
Undefined	Data item or entity	Entity	Table	Class
Elementary	Data item	—	—	—
Structured	Entity	Entity	Table	Class

If you export sub-data objects along with their structured parent, the sub-data are exported as follows:

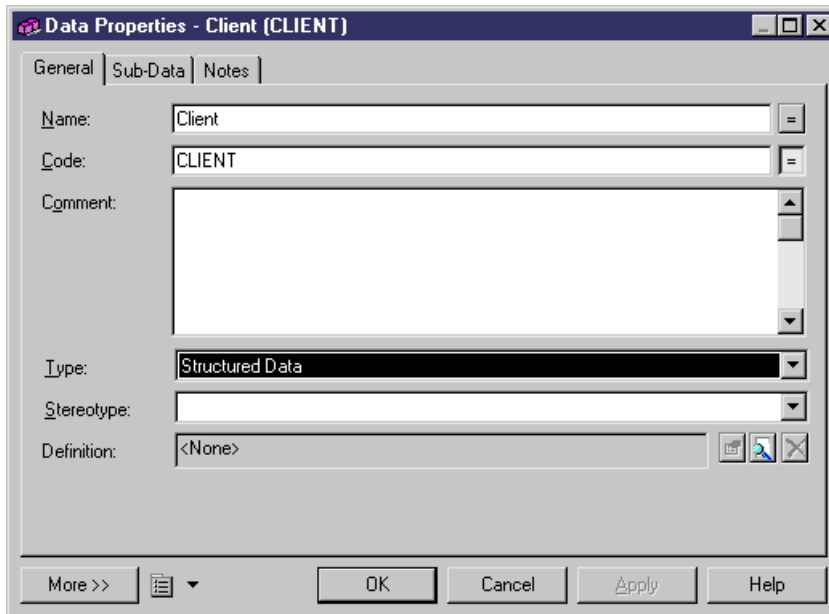
Sub-Data Type	CDM	LDM	PDM	OOM
Undefined or Structured	Entity linked to parent by relationship	Entity linked to parent by relationship	Table linked to parent by reference	Class linked to parent by association link
Elementary	Data item and entity attribute	Attribute of parent entity	Column of parent table	Attribute of parent class

Note: If you export sub-data objects without their parent, then the rules above for exporting data objects are applied.

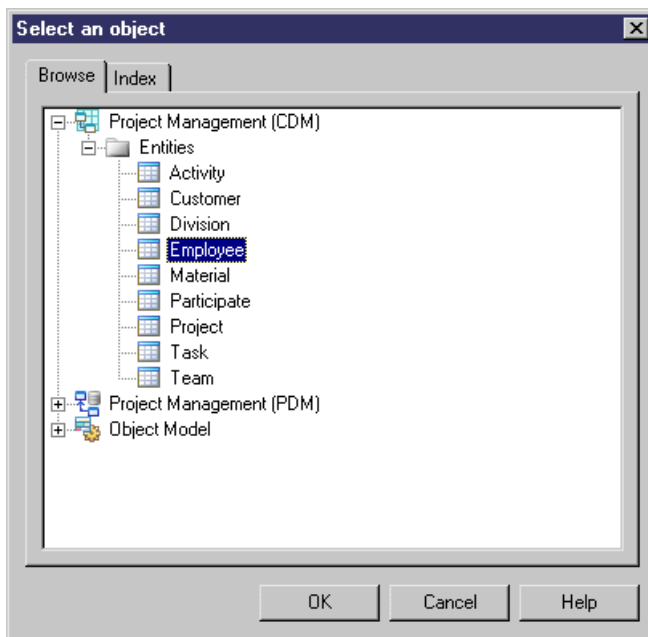
Linking Data Objects to External Model Objects

You can link BPM data objects to existing data items, entities, tables or classes in other PowerDesigner models.

1. Open the data object property sheet, and select the appropriate type from the **Type** list.



2. Click the **Select Definition Object** tool to the right of the **Definition** field to open a dialog which allows you to select an object to associate with the data object from the models open in the workspace:



3. Select the appropriate object in the tree view and click **OK**.

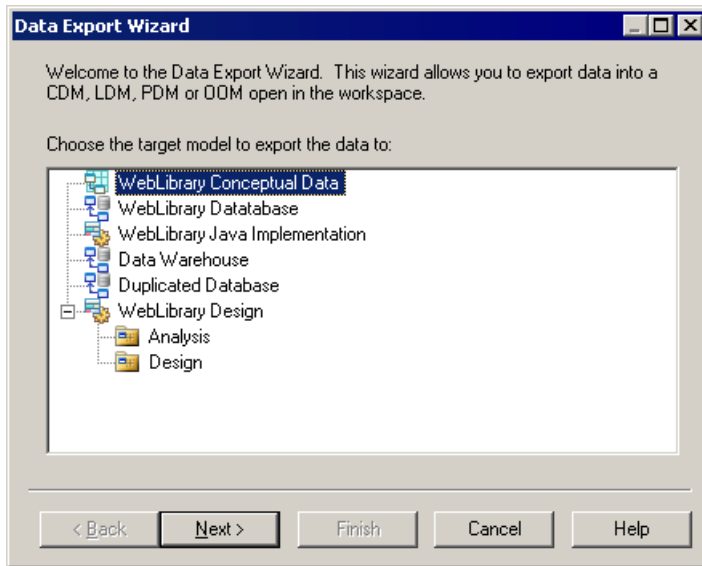
The selected object is displayed in the **Definition** field. You can click the **Properties** tool to the right of the field to open its property sheet or the **Remove Link** tool to sever its association with the data object.

Note that if you subsequently change the type of the data so that it no longer corresponds with the object defined in the **Definition** box, you will be prompted to confirm the change. If you do so, the link between the data and the object is removed.

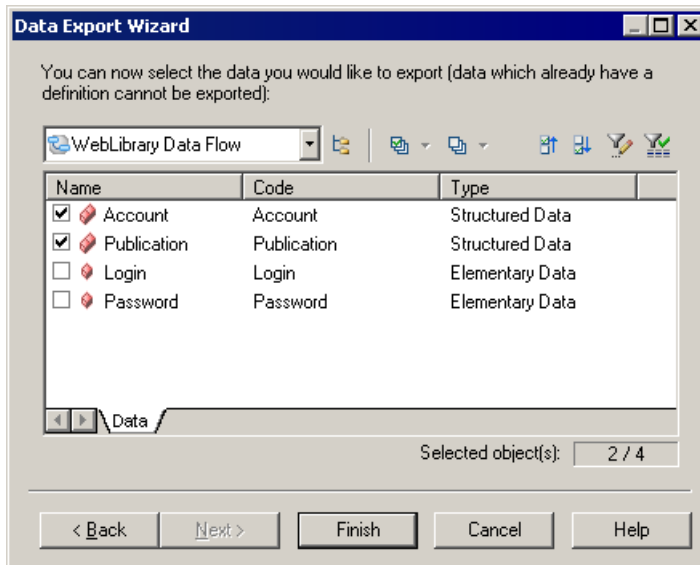
Exporting Data to Other PowerDesigner Models

You can export BPM data objects as data items, entities, tables or classes in other PowerDesigner models. The target model must be open in the workspace, and only data objects which are not already linked to any external model objects are available for export.

1. Select **Tools > Data Export Wizard** to open the Data Export Wizard, which lists all the models and packages open in the workspace to which you can export data:



2. Select the target model or package to which you want to export data and click **Next**.
3. [CDM only] When you export undefined data or sub-data to a CDM, the wizard prompts you to specify whether you want to export them as data items or entities. Select an object type and click **Next**.
4. The data selection page lists all the data objects available to export:



5. Select the data you want to export, and then click **Finish**.

The data is exported as the appropriate objects in the target model.

Note: If you export a data object that has the same name and code as an existing object in the target model, the data will be linked to the existing object.

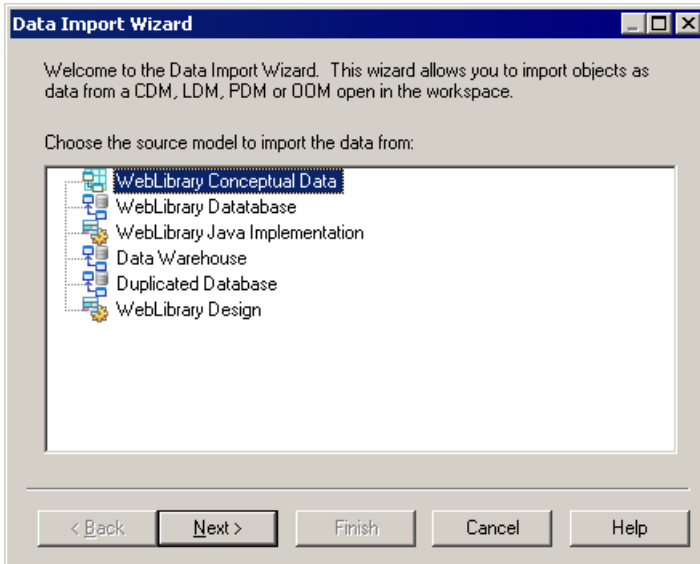
You cannot export a data object more than once. If you want to re-export a data object, you must first delete the link to the external object.

A class attribute or a table column cannot be shared, but sub-data objects can be shared by several data object parents. When you export an elementary or undefined sub-data object as an attribute in the OOM or as a column in the PDM, the link between the sub-data object and the definition object is not saved in the Definition box of its property sheet.

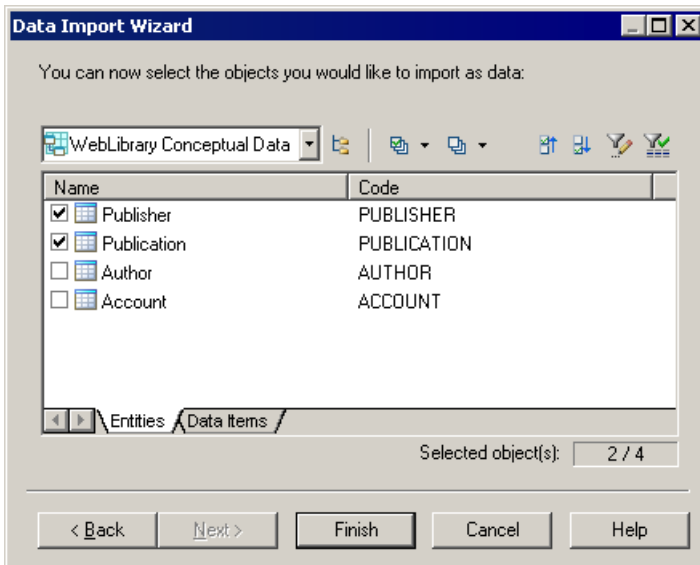
Importing Data from Other PowerDesigner Models

You can import CDM data items and entities, LDM entities, PDM tables, and OOM Classes as data objects in your BPM. The source model must be open in the workspace and only external model objects not already linked to data objects in the model are available for import.

1. Select **Tools > Data Import Wizard** to open the Data Import Wizard, which lists all the models and packages open in the workspace from which you can import data.



2. Select the source model or package from which you want to import data and click **Next**.
3. The data selection page lists all the external model objects available to import:



4. Select the objects you want to import as data, and then click **Finish**.

The data is imported into the BPM. CDM data items are imported as elementary data and all other objects are imported as structured data.

Note: If you import an object that has the same name, code and type as an existing data object in the BPM, the existing data object is reused, unless it already has a definition object. In this

case, the new data is automatically renamed and linked to the selected object in the source model.

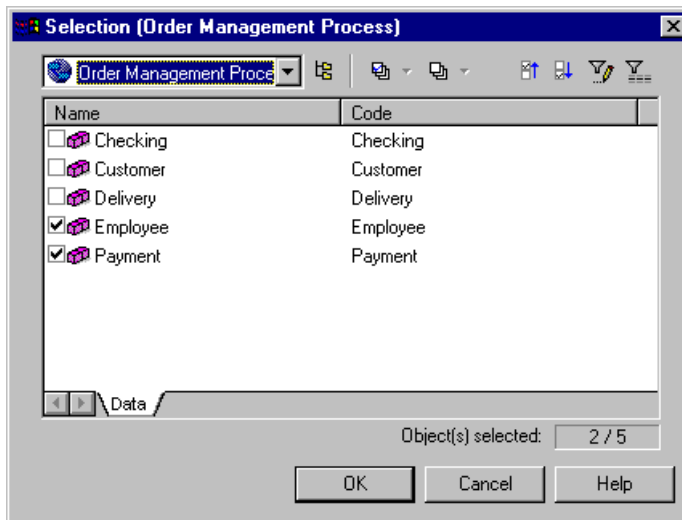
When an entity, a table or a class object is imported as a data object, their data items, columns or attributes are automatically imported as sub-data and linked to their parent structured data.

A PDM reference between two tables is imported as a parent/child relationship between the two imported data. Foreign key column objects are not imported because they are created by the PDM reference between tables.

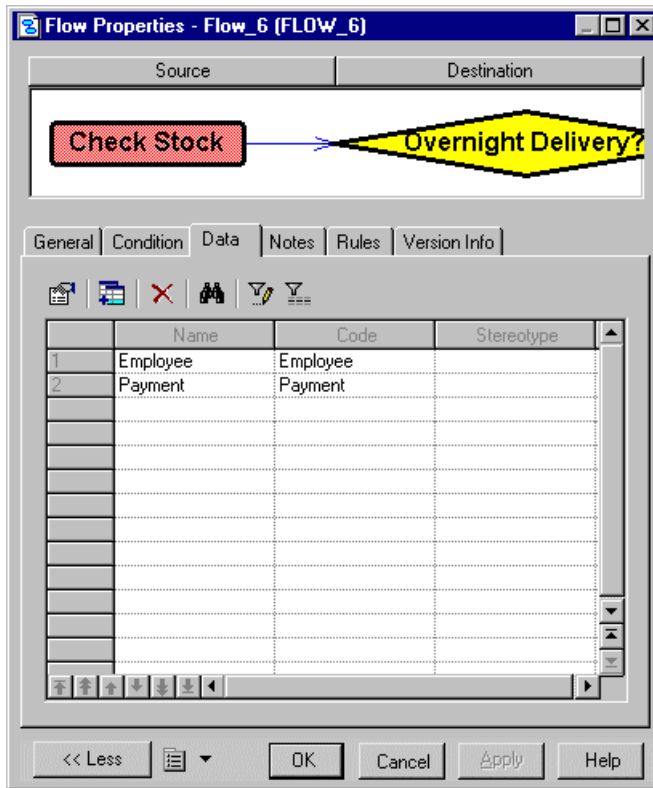
Specifying Data for a Flow, a Resource Flow or a Message Format

You can specify the data conveyed by flows, resource flows and message formats from the Data tab of their property sheets.

1. Open the property of a flow, a resource flow, or a message format, and click the Data tab.
2. Click the Add Data tool to open a data selection list.



3. Select one or more data, and then click OK to close the selection list, and associate the data with the flow, resource flow or message format.



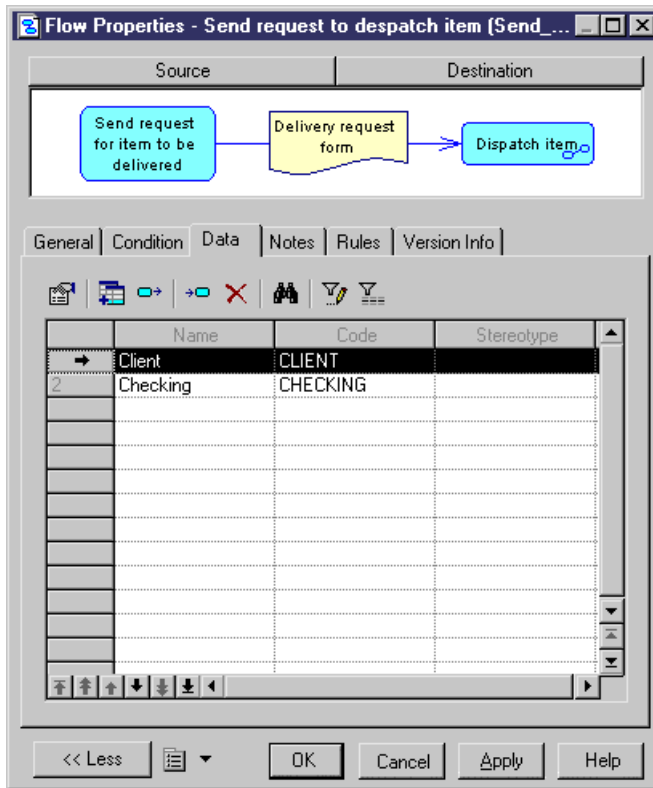
4. Click OK to close the property sheet and return to the model.

Note: You can display the list of data conveyed by a flow or a resource flow on its symbol by selecting **Tools > Display Preferences > Flow** (or Resource Flow) and selecting the Data List radio button.

Migrating the Data of a Flow to a Process

Data objects specified on a flow can be added to its source or destination process, using the Migrate tools in the Data tab of its property sheet.

1. Open a flow property sheet, click the Data tab and select one or more data to migrate.



2. Click the Migrate to destination process or Migrate to source process tool.

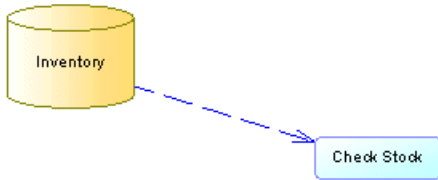
A message box is displayed to indicate to which process the data was migrated. Data migrated to a source process is created with a "Create" access type, while data migrated to a destination process is created with a "Read" access type.

3. Click OK to close the property sheet and return to the model.

Resources (BPM)

A *resource* is a data store. It can be a database, a document, a data, or a component to which a process have access. You access the data stored in a resource using a resource flow.

In the following example, the Check Stock process reads data contained in the Inventory resource:



A resource can be created in a choreography diagram with any of the following target languages:

- Analysis languages
- Collaborative languages

A resource can be created in a data flow diagram to design a data store. For more information, see *Chapter 11, Data Flow Diagram (DFD)* on page 199.

You cannot create shortcuts to a resource.

Creating a Resource

You can create a resource from the Toolbox, Browser, or **Model** menu.

- Use the **Resource** tool in the Toolbox.
- Select **Model > Resources** to access the List of Resources, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Resource**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Resource Properties

To view or edit a resource's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Property	Description
Number ID	<p>Specifies an incrementing number to help you identify resources. You can modify this value at any time by entering an integer greater than 0. Any change you make will not, by default, affect other numbers in the series.</p> <p>When working in a data flow diagram, you can at any time right-click the diagram background and select Renumber Data Store IDs to renumber the resources following their position in the data flow (see <i>Process and Data Store Numbering</i> on page 203).</p>
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

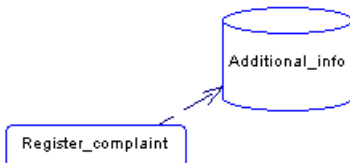
Data Tab

This tab is only available for the Analysis and the Data Flow Diagram languages, and lists the data associated with the resource. Data come from the input and output resource flows (see *Resource Flows (BPM)* on page 95) .

Resource Flows (BPM)

A *resource flow* allows a process to access a resource and describes an interaction between them.

In the following example, the Register_complaint process creates, updates or deletes data contained in the Additional_info resource:

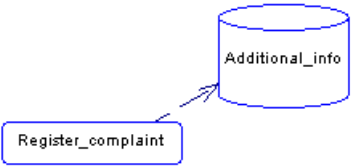
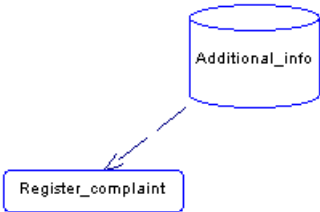
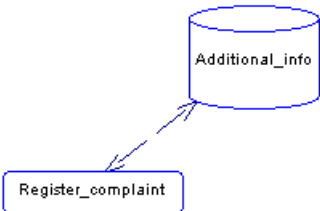


A resource can be created in a choreography diagram with any of the following target languages:

- Analysis languages
- Collaborative languages

A resource flow can also be created in a data flow diagram (see *Chapter 11, Data Flow Diagram (DFD)* on page 199).

The way you draw a resource flow determines the way the process uses the resource:

Type	Description
<p>From process to resource:</p> 	<p>The process creates, updates or deletes data contained in the resource depending on the access modes selected in the resource flow property sheet.</p>
<p>From resource to process:</p> 	<p>Data contained in the resource are read by the process.</p>
<p>Two ways:</p> 	<p>If you select a Read access mode on a resource flow together with one or more other access modes (Create, Update, Delete), the resource flow symbol is bi-directional.</p>

Message Format

When working with the Analysis language and ebXML language, you can associate a message format with a resource flow in order to define the format of information exchanged between a process and a resource (see *Message Formats (BPM)* on page 78).

The following rules apply:

- *Reflexive flows* (same source and destination process) are allowed on processes.
- Two resource flows between the same source and destination objects are permitted, and called *parallel flows*.
- A resource flow cannot link shortcuts.

Creating a Resource Flow

You can create a resource flow from the Toolbox, Browser, or **Model** menu.

- Use the **Flow/Resource Flow** tool in the Toolbox.

- Select **Model > Resource Flows** to access the List of Resource Flows, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Resource Flow**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Resource Flow Properties

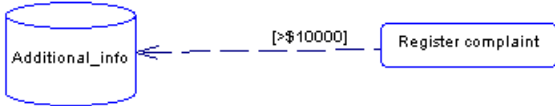
To view or edit a resource flow's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Process / Resource	Specifies the extremities of the resource flow. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.
Message format	[Not available for BPMN and DFD languages]. Specifies the format of the data exchanged between the process and the resource. You can choose from the following values: <ul style="list-style-type: none"> • None – no exchange of data. • Undefined [default] – Click the Create tool to the right of the list to create a message format (see <i>Message Formats (BPM)</i> on page 78).
Access mode	Specifies the way to access data in a resource, and thus defines the resource flow direction. You can choose from the following values: <ul style="list-style-type: none"> • Read – from resource to process. • Create, Update, Delete – from process to resource.

Condition Tab

The **Condition** tab defines the nature of the condition attached to a resource flow, and contains the following properties:

Parameter	Description
Alias	Short name for the condition to be displayed next to its symbol in the diagram.
Condition (text box)	<p>Specifies a condition to be evaluated to determine how the resource flow should be traversed. You can enter any appropriate information in this box, as well as open, insert and save text files. You can open the Condition tab by right-clicking the resource flow symbol, and selecting Condition in the contextual menu. The condition is displayed next to the process symbol:</p>  <p>The diagram illustrates a resource flow. On the right, there is a rounded rectangular process symbol labeled 'Register complaint'. A dashed arrow points from this process to a cylindrical data store symbol on the left labeled 'Additional_info'. Above the arrow, the condition '[>\$10000]' is written.</p>

When there are several flows, each condition is evaluated in order to choose the one the resource flow will transit on.

Data Tab

This tab is only available for the Analysis and the Data Flow Diagram languages and lists the data associated with the resource flow. You can add or create data, and specify which data is conveyed by the resource flow without any information on its format (see *Specifying Data for a Flow, a Resource Flow or a Message Format* on page 91).

Service Providers (BPM)

The business process world is, most of the time, a set of linked services, or repeatable business tasks, that can be accessed when needed over a network, as though they were all installed on your local desktop.

To invoke an external web service, you need the WSDL from the service because it describes the port, service name, operations, and messages that the process needs to communicate with the service. Web service descriptions are modeled in PowerDesigner using service providers, interfaces (see *Service Interfaces (BPM)* on page 110), and operations (see *Operations (BPM)* on page 111).

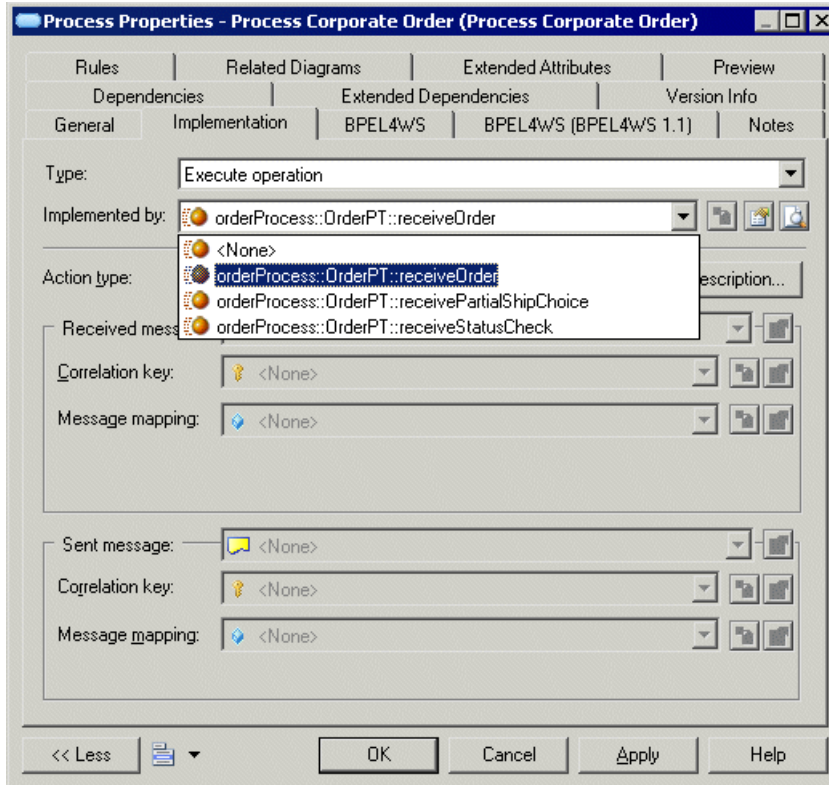
A *service provider* is an object that contains a set of service interfaces. For example a "Travel Agent" service provider may contain the following service interfaces: "TravelAgentToAirline" and "TravelAgentToTraveler".

A service provider can be created in the following diagrams with any orchestration languages:

- Choreography Diagram – the service provider allows you to implement processes with the service operations contained by its service interfaces (see *Example: Using the Execute Operation Implementation Type* on page 40). You can import a WSDL to recover web service description objects or create them manually. You can also import an OOM component or a database web service as a service provider and export service providers

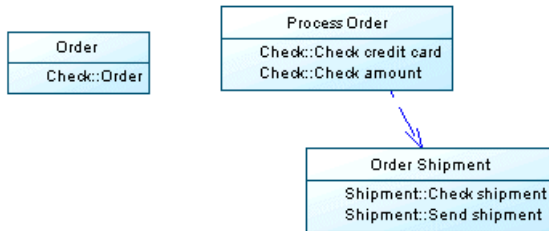
(see *Importing and Exporting Service Providers From/to Another Model* on page 106). The service provider has no graphical symbol in this diagram.

In the following example, the Process Corporate Order process can be implemented by the operations available in the Implemented by list. These are owned by the OrderPT service interface in the orderProcess service provider:



- Process Service Diagram – service providers are displayed with the interfaces and operations they contain, and can be linked with traceability links. These links are for documentation purposes only (see *Chapter 4, Building Process Service Diagrams* on page 127).

The following example shows three service providers with their interfaces and operations. The Process order service provider is dependent on the Order Shipment service provider:



When you copy a service provider, you also copy its associated service interfaces. Shortcuts for service providers are not permitted.

Creating a Service Provider

You can create a service provider from the Browser or from the **Model, Language** or **Tools** menu.

- Select **Model > Service Providers** to access the List of Service Providers, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Service Provider**.
- Select **Language > Import WSDL** to access the Import WSDL dialog box, and select a WSDL to import.
- Select **Tools > Import Service Provider** to access the Service provider Import Wizard, and select an OOM or a PDM object to import.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Service Provider Properties

To view or edit a service provider's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.

Property	Description
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Filename	Specifies the path to the file that contains the whole service definition. The path is set during import and used during file generation. Use the tools to the right of the box to select a WSDL file or open the currently selected WSDL file.
Endpoint URL	Specifies the address at which the service can be reached.
Target namespace	Specifies a URI (Uniform Resource Identifier) reference that uniquely identifies the web service and avoids conflicts with other web services with the same name. By default, it is: urn:<Service Provider Code>.
Prefix	Specifies a prefix for the target namespace. All the schema elements with this prefix in their start-tag will be associated with the namespace. The default value is: "tns" that stands for "This NameSpace". For example: <tns:invoice>, where "tns" is the prefix associated with the XSD document that describes the "invoice" markup.
Implementation	Specifies a link between the service provider and an OOM component or a PDM database web service. Use the tools to the right of the box to select an implementation object, view the properties of the currently selected object, or remove it.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Service Interfaces Tab

This tab displays a list of service interfaces associated with the service provider (see *Service Interfaces (BPM)* on page 110).

XSD Documents Tab

This tab displays a list of XSD documents that define the data schemas describing the service provider (see *XSD Documents* on page 117).

XML Namespaces Tab

This tab displays a list of XML namespaces prefixes used by the WSDL file. An *XML namespace* is a URI (Uniform Resource Identifier) reference used in XML documents, which indicates a location where element and attribute names are declared. An XML document can contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, then the ambiguity between identically named elements or attributes can be resolved.

For example, an XML document can contain references to a customer and an ordered product. Both the customer element and the product element can have a child element named

"ID_number". If you declare a namespace (i.e. which vocabulary an element or attribute name comes from) you differentiate them, and avoid ambiguity.

An XML namespace can be created in a choreography diagram with any orchestration languages.

The namespace declaration syntax is the following:

```
xmlns:prefix="namespaceURI"
```

An XML namespace contains the following parameters:

Parameter	Description
xmlns	Stands for XML Namespace. Indicates an XML namespace declaration.
:prefix	[optional] Shorthand for the full name of the namespace. It qualifies elements belonging to that namespace. You use it when you need to differentiate same namespace references.
namespaceURI	Uniquely identifies a namespace in the XML document.

The XML namespace can be defined in the XML Namespaces tab for each service provider and XSD document in your model.

In the following example, bk is used as a shorthand for the full name of its respective namespace:

```
<BOOKS>  
  <bk:BOOK xmlns:bk="urn:example.tyler.com:BookInfo"  
    <bk:TITLE>Funny Words</bk:TITLE>  
  </bk:BOOK>  
</BOOKS>
```

Data Schema Tab

This tab contains the data schema of the service provider, which can be created manually or come from:

- The imported WSDL
- The XSD document associated with an XML model. If there are more than one XSD document, data schemas are concatenated

The Data Schema tab contains the following properties:

Property	Description
Language	Specifies the type of schema language used for message parts. You can enter your own type of language or choose one of the following XML document languages: <ul style="list-style-type: none"> • DTD • XML Schema • RELAX NG
Data Schema (text box)	Specifies the message part definition details. You can enter any appropriate information in this field, as well as open, insert and save text files.

Importing a Service Provider from a WSDL File

If you have a WSDL file or if you find a WSDL published in a UDDI server, you can import the WSDL to create an abstract definition of a web service using service description objects (service providers, service interfaces, and operations).

Then, you can proceed to the implementation of your processes using operations and associated messages (see *Specifying Implementation Types* on page 37).

The import process analyzes the WSDL file to find the different web services, port types, messages, operations, and parts defined in the script, and converts these elements into BPM objects as follows:

WSDL element	BPM object
WSDL file	Service provider
Port type	Service interface
Operation	Operation
Message	Message format
Part	Message part

1. Select **Language > Import WSDL** to display the Import WSDL dialog box.
2. Enter an URL in the WSDL URL box to specify the location of the WSDL file on the web. The URL is displayed in the Filename box of the service provider property sheet. You can use the tools to the right of the box to browse for a file or browse UDDI (see *Browsing for a WSDL File on a UDDI Server* on page 104).
3. [optional] Click the Preview WSDL button to preview the WSDL file, and the unique key used to locate the UDDI. This button is not available if you select several files to import.

- [optional] Click the Options tab and select the Create XML Model check box, if you want to automatically create an XML model for each schema contained in the WSDL file. This provides you with a graphical representation of the data schema.
- Click OK to import.

A progress box is displayed. If the model in which you are importing already contains data, the Merge Models dialog box is displayed.

For more information about merging models, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

- Click OK to return to the model.

The imported elements are added to your model, are visible in the Browser, and in the Reverse tab of the Output window. If you have selected the Create XML Model option, the XML model(s) corresponding to the WSDL schema(s) are also created in the workspace.

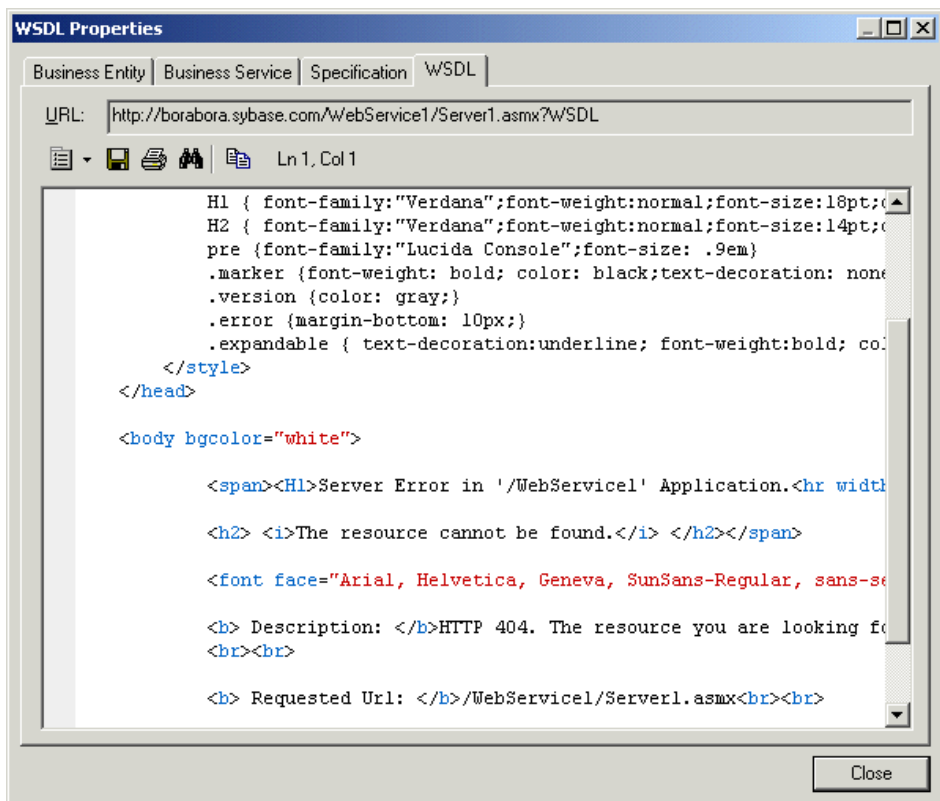
Browsing for a WSDL File on a UDDI Server

You can browse for a WSDL on a UDDI server. You need to have Internet Explorer 5 or higher to use the Browse UDDI feature.

- Select **Language > Import WSDL** to display the Import WSDL dialog box.
- Click the Browse UDDI tool to the right of the WSDL URL box to display the Browse UDDI dialog box.

Business Entity Name	Service Name	WSDL Name	WSDL URL
----------------------	--------------	-----------	----------

3. Enter a URL in the UDDI operator URL list or select one from the list, and select a UDDI version in the UDDI Version list.
4. Select an item to search from the Search In list. You can search for a web service per business entity (company name), web service name, and WSDL name.
5. Enter a keyword in the Search For list and click the Search button. You can search for a name of the item selected in the Search In list. The result is displayed in the Search Result window.
6. [optional] Click the Preview WSDL button to open the WSDL property sheet and click the WSDL tab to display the WSDL.



7. [optional] Click the Business Entity tab to display data about the company and the Business Service tab to display data about the service.
8. Click Close to close the WSDL property sheet.
9. Click OK in each of the dialog boxes to return to the model.

Importing and Exporting Service Providers From/to Another Model

PowerDesigner lets you define the links between the abstract definition of service interfaces and operations in the BPM and their concrete implementation, either in OOM components, or in PDM database web services.

You can import and export service providers in and from a choreography diagram with any orchestration languages.

You can use this feature to:

- Initialize a BPM with existing implementation defined in an OOM or a PDM.
- Use the requirement analysis of the BPM to initialize an OOM, and start implementing these needs.

OOM Implementation

From an OOM, you can import a component with implementation classes owning operations and create a service provider (see *Importing a service provider from an OOM or a PDM* on page 107). You can also export a service provider as a component in the OOM (see *Exporting a service provider from a BPM* on page 108).

During export/import, the following mappings are performed:

OOM object	BPM object
Component (web service, EJB or any other)	Service provider
Web Service implementation class, or a UML interface associated with the component	Service interface
Class (or interface) operation	Operation of the service interface
SOAP Input value (Input box content)	Operation Input message
InputSoapMessageName extended attribute	Operation Input name
SOAP Input schema text	Input message format text

The OOM object is referenced in the Implementation box of the property sheet of the BPM service provider, service interface and operation.

For more information about OOM components, see *Object-Oriented Modeling > Building OOMs > Implementation Diagrams > Components (OOM)*.

PDM Implementation

From a PDM, you can import the WSDL URL corresponding to a database web service and create a service provider. Note that you cannot export a service provider as a database web service.

To import a database web service as a service provider, it must:

- Use the SOAP protocol
- Be generated and deployed into the database

The database server must be running in order to have a WSDL URL.

During import, the following mappings are performed:

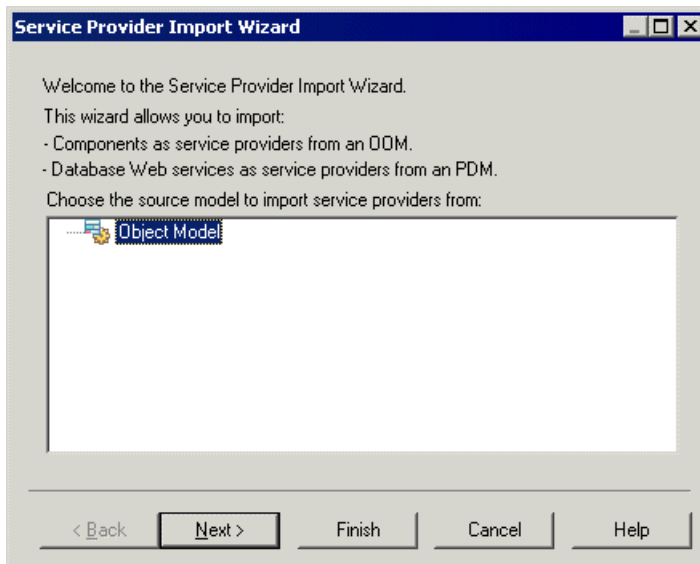
PDM object	BPM object
Database web service	Service provider
Web service operations	Operations in a service interface

For more information about database web services, see *Data Modeling*.

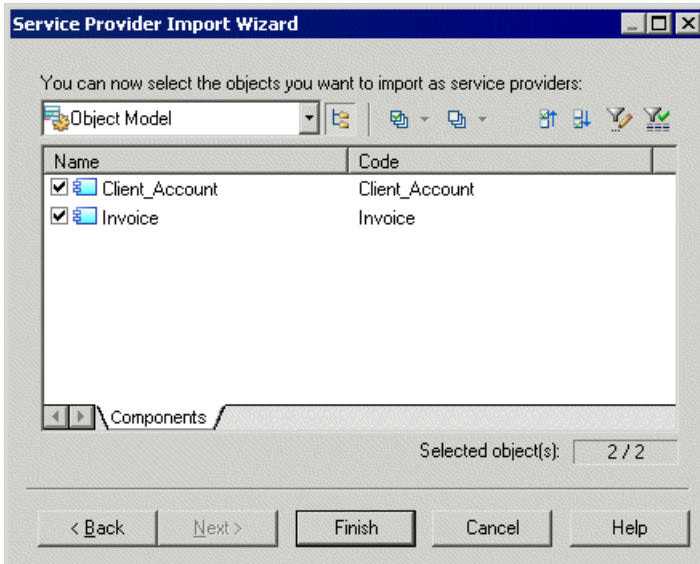
Importing a Service Provider from an OOM or a PDM

The Service Provider Import Wizard is available from the Tools menu when an implementation OOM and/or a PDM with a SOAP web service are open in the workspace.

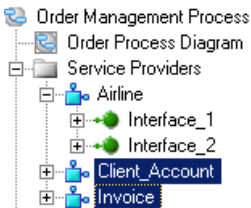
1. Select **Tools > Service Provider Import** to open the Service Provider Import Wizard.



2. Select a model, and click Next to go to the next page.



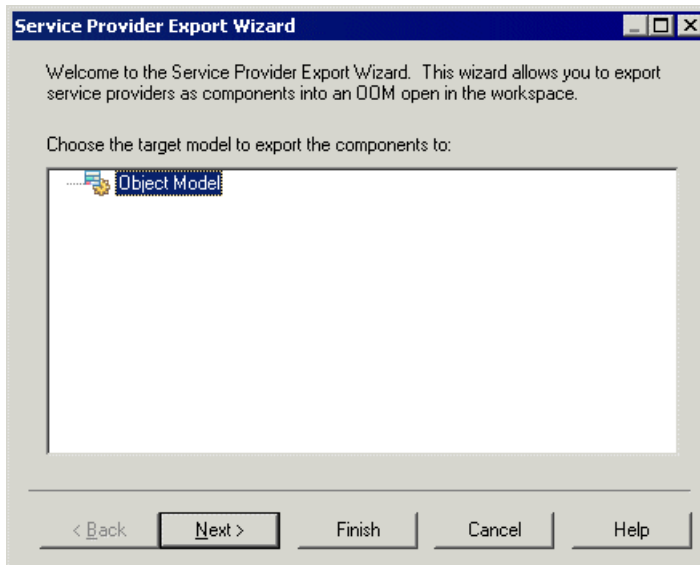
3. Select the OOM components or PDM web services you want to import as service providers and click Finish to begin the import. The components or web services you have selected are imported as service providers in the BPM and display in the Service Providers folder in the Browser:



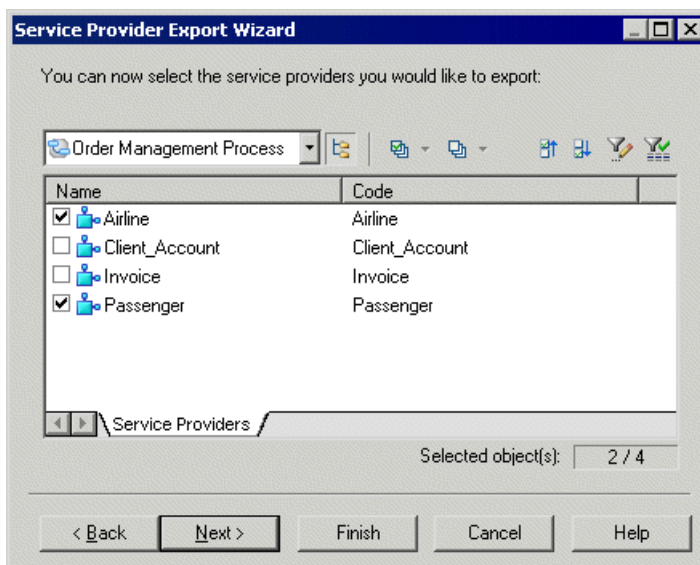
Exporting a Service Provider from a BPM

The Service Provider Export Wizard is available from the Tools menu when at least one service provider is defined in the current BPM and an OOM is open in the workspace.

1. Select **Tools > Service Provider Export** to open the Service Provider Export Wizard.

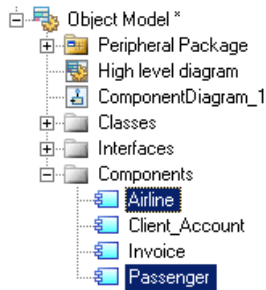


2. Select a target OOM and click Next to go to the next page.



3. Select the service providers you want to export and click Next to choose a component type. If the target OOM language supports web services, the Web Service check box is selected by default, and you can select a web service type or component type from the list.
4. Select the type of component you want to create, and click Finish to exit the wizard.

The service providers you have selected are exported as components in the OOM and display in the Components folder in the Browser:



Service Interfaces (BPM)

A *service interface* is an object that contains a set of operations. For example a "LoanApproval" service interface may contain the following operations: "Request" and "Check".

A service interface can be created in the following diagrams with any orchestration languages:

- Choreography Diagram
- Process Service Diagram

For more information, see *Service Providers (BPM)* on page 98.

A service interface corresponds to the Port Type object in a WSDL file, and belongs to a service provider.

When you copy a service interface, you also copy its associated operations. Shortcuts for service interfaces are not permitted.

Creating a Service Interface

When you import a service provider, you also import its service interfaces. However, you can manually create a service interface.

You create a service interface in any of the following ways:

- Open the **Interfaces** tab in the property sheet of a service provider, and click the **Add a Row** tool.
- Right-click the service provider in the Browser, and select **New > Service Interface**.

The list of service interfaces accessible from the Model menu only allows you to visualize all service interfaces for all service providers within the model. You cannot create service interfaces from this list.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Service Interface Properties

To view or edit a service interface's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Provider	[Read-only] Specifies the service provider owning the service interface. You can click the Properties tool next to the Provider box to display the service provider properties.
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Implementation	Specifies a link between the service interface and an OOM class or interface. Use the tools to the right of the box to select an implementation object, view the properties of the currently selected object, or remove it.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Operations Tab

This tab lists the operations associated with the service interface (see *Operations (BPM)* on page 111).

Operations (BPM)

An *operation* is contained by a service interface, and which comprises input and output elements defined in terms of messages or message parts.

An operation can be created in the following diagrams with any orchestration languages:

- Choreography Diagram
- Process Service Diagram

An operation belongs to a service interface which, in turn, belongs to a service provider (see *Service Providers (BPM)* on page 98). The operation describes the implementation of an

atomic process, and can be sent or received by an activity (see *Example: Using the Execute Operation Implementation Type* on page 40).

The Dependencies tab of the operation property sheet allows you to visualize all the processes implemented by the current operation. Shortcuts for operations are not permitted.

If you copy an operation within the same model, the associated messages are reused, and if you copy it to another model, the messages are also duplicated. If you move:

- An operation to another service interface in the same model - All the links to the processes that use the operation are deleted. However, if you move an operation to another service provider all operation messages are duplicated.
- An operation from the browser to a process in the diagram window - The process is implemented by the operation (see *Example: Using the Execute Operation Implementation Type* on page 40).
- A service provider to another model - Its service interfaces and operations are also moved. The associated message format and process using the operation are not moved with the service provider, a copy of the whole service provider remains in the initial model to preserve these links.

Creating an Operation

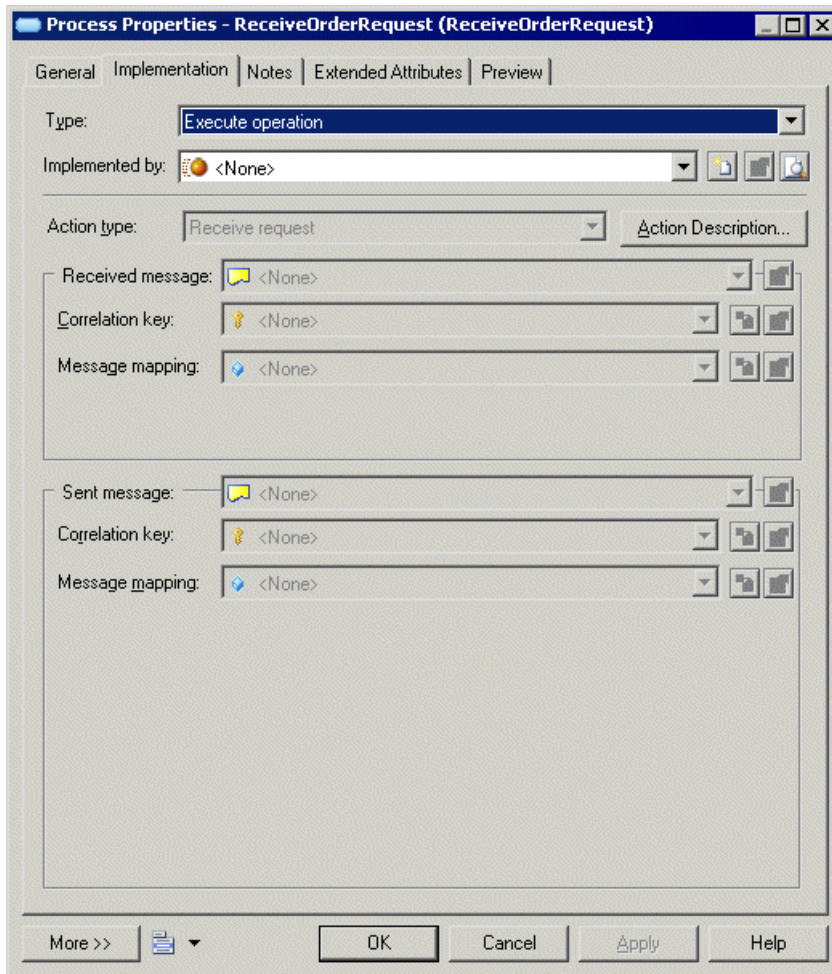
You can create an operation using a Wizard or from the property sheet of, or in the Browser under, a service interface.

- Open the **Operations** tab in the property sheet of a service interface, and click the **Add a Row** tool.
- Right-click the service interface in the Browser, and select **New > Operation**.
- Use the Create New Operation Wizard from the **Implementation** tab of the process property sheet (see *Using the Create New Operation Wizard* on page 112)

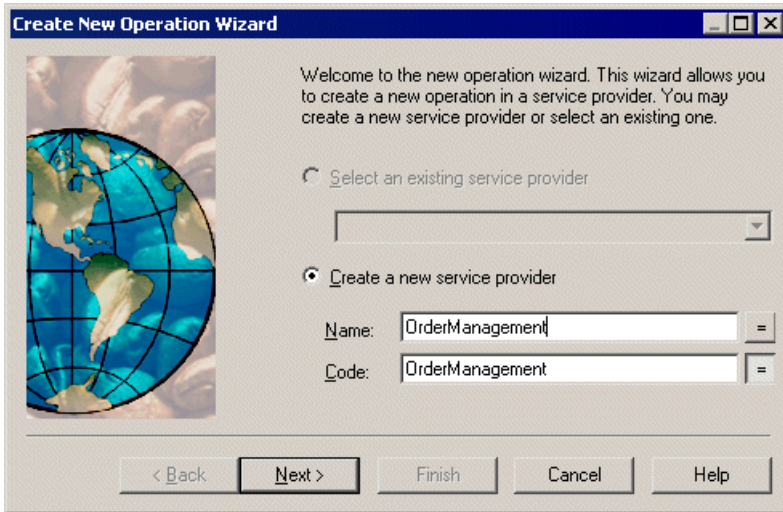
Using the Create New Operation Wizard

When implementing an atomic process (activity), you may need to create the required implementation operation directly from the process.

1. Open a process property sheet, click the Implementation tab, and select Execute Operation from the Type list. The corresponding fields display:



2. Click the Create tool to the right of the Implemented by list to open the Create New Operation Wizard. Note that this tool is unavailable when an operation is already selected in the list:



3. Select one of the following radio buttons:

- Select an existing service provider to select an existing service provider from the list. If no service provider exists in the model, this option is unavailable.
- Create a new service provider, and enter a name for the new service provider.

4. Click the Next button to go to the next page.



5. Select one of the following radio buttons:

- Select an existing service interface to select from the list an existing service interface that belongs to the service provider you have selected. If you chose to create a service provider in the previous page, this option is unavailable.

- Create a new service interface, and enter a name for the new service interface.
6. Click Finish to complete the creation. The operation property sheet opens. Define properties as required.
 7. Click OK to close the property sheet and return to the model.

Operation Properties

To view or edit an operation's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Provider	[Read-only] Specifies the name of the service provider owning the operation. Click the Properties tool next to the Provider box to display the service provider properties.
Interface	[Read-only] Specifies the name of the service interface owning the operation. Click the Properties tool next to the Interface box to display the service interface properties.
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Implementation	Specifies a link between the operation and an OOM operation or a PDM web service operation. Use the tools to the right of the box to select an implementation object, view the properties of the currently selected object, or remove it.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Input/Output Tab

This tab contains the following properties:

Property	Description
Type	<p>Specifies the type of the operation:</p> <ul style="list-style-type: none"> • Undefined – [default] No message. • One Way – the activity calls the web service and no response is expected (input message only). • Request-Response – the activity calls the web service and a response is expected (input and output messages). Fault messages can also be sent as output in case of error. • Solicit-Response – the web service solicits the activity and a response is expected (output and input messages). Fault messages can also be sent as input in case of error. • Notification - the web service solicits the activity and no response is expected (output message only).
Input message	Specifies a name and message format for the input message. Select a format from the list or use the tools to the right of the list to create a format or view the properties of the currently selected format.
Output message	Specifies a name and message format for the output message. Select a format from the list or use the tools to the right of the list to create a format or view the properties of the currently selected format.

This table summarizes the relationships between the input/output messages of the operation and the received messages of the activity:

Operation\Activity	Receive request	Receive request and reply
Undefined	—	—
One-Way	Input is received.	—
Request-Response	Input is received. Output is ignored.	Input is received. Output is sent.
Notification	Output is received.	—
Solicit-Response	Input is ignored. Output is received.	Input is sent. Output is received.

This table summarizes the relationships between the input/output messages of the operation and the sent messages of the activity:

Operation\Activity	Reply	Reply fault	Invoke operation
Undefined	—	—	—
One-Way	—	—	Input is sent.
Request-Response	Input is ignored. Output is sent.	Input is ignored. Output is ignored. Fault is sent.	Input is sent. Output is received.
Notification	—	—	Output is sent.
Solicit-Response	Input is sent. Output is ignored.	Input is ignored. Output is ignored. Fault is sent.	Input is received. Output is sent.

Faults Tab

This tab is only available for Request-Response and Solicit-Response operations and lists the fault links between the operation and a message format. You can add or create a fault using the Add Objects and Create an Object tools.

XSD Documents

An *XSD document* defines the data schema handled by a service provider. It can be created in a choreography diagram with any orchestration languages.

When you reverse engineer or import web services, provided you have selected the Create XML Model option, an XSD document is created for each data schema found in the source WSDL. This XSD document is displayed in the XSD Documents tab of the service provider property sheet.

An XSD document can be associated with an XML model in order to provide a graphical representation of the service provider data schema (see *Attaching an XML model to an XSD document* on page 119).

An XSD document belongs to a service provider.

Creating an XSD Document

You can create an XSD document from the property sheet of, or in the Browser under, a service provider.

- Open the **XSD Document** tab in the property sheet of a service provider, and click the **Add a Row** tool.
- Right-click the service provider in the Browser, and select **New > XSD Document**.

The list of XSD documents accessible from the **Model** menu only allows you to visualize all XSD documents for all service providers within the model. You cannot create XSD documents from this list.

XSD Document Properties

To view or edit an XSD document's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Target namespace	Specifies the URI (Uniform Resource Identifier) reference that uniquely identifies the data schema and avoids conflicts with other data schemas with the same name.
Schema location	Specifies the URI (Uniform Resource Identifier) reference for the location from which the data schema was imported.
Schema model	Specifies the reference to an XML model that represents the data schema. You can select a model from the list or use the tools to the right of the list to create a model and view the properties of the currently selected model.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

XML NameSpaces Tab

This tab displays a list of XML namespaces prefixes used by the WSDL file, which can reference included data schema namespaces or any external namespace. For more information about using this tab, see *Service Provider Properties* on page 100.

Schema Tab

This tab specifies the message part definition details. You can enter any appropriate information in this field, as well as open, insert and save text files. The first lines of the schema display the XML version, encoding format, and namespace details.

Attaching an XML Model to an XSD Document

Attaching an XML model to an XSD document allows you to easily model its data schema in a graphical environment. The XML model schema code is displayed in the Schema tab of the XSD document.

You can attach an XML model to an XSD document in any of the following ways:

- Import or reverse a WSDL file, if you have selected the Create XML Model option. The XSD document created during the import is automatically linked to the XML model. It displays in the Schema model list of the XSD Document property sheet.
- Create a new XSD document in a service provider property sheet, and attach an XML model to the XSD document.

1. Open an XSD document property sheet.
2. Select an XML model among the XML models open in the workspace in the Schema model list.

or

Click the New tool next to the Schema model list. The property sheet of a new XML model displays. Type a name and a code for the new XML model and click OK.

The name of the XML model displays in the Schema model list. The new XML model is also open in the workspace.

3. [optional] Start modeling the data schema in the XML model.

Variables (BPM)

A *variable* is a data container, which holds temporary values that can be passed between processes as input and output parameters, and which are important for their correct execution. For example, variables are useful to determine routing decisions or to build the messages a process has to send.

A variable can be created in a choreography diagram with any orchestration languages, and is associated with a process implemented by an operation.

Variables can be used in conjunction with:

- Processes – to build the process messages
- Correlation keys – to identify a process instance using a set of variables
- Data transformations – to copy data from one variable to another

By default, a variable name or code must be unique within the parent scope (package, composite process, or model) but can be used by any process (activity) defined at the same level. However, two variables can share the same name when they belong to different composite processes contained in the same package.

Moving Variables

When you move variables, the following rules apply:

Move to...	Description
Another model	A copy of the variable stays in the source model, when the variable is used by at least one process in that model, as external shortcuts for variables are not allowed.
Same model	A shortcut for the variable stays in the source process or package, when the variable is used by at least one process in that package.

Creating a Variable

You can create a variable from the Browser or **Model** menu.

- Select **Model > Variables** to access the List of Variables, and click the Add a Row tool
- Right-click the model or package in the Browser, and select **New > Variable**

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Variable Visibility

Note that you can also create variables local to a process. In that case, only the process in which you have created the variables can use them (see *Process Properties* on page 34).

Variable Properties

To view or edit a variable's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Property	Description
Data type	Specifies the data type of the variable. You can choose from a list of simple data types, or type a complex type (XSD element, OOM class, XML object, etc.). [BPEL languages only] Can also specify a message format. You can click the Create tool to create a new message format.
Element type	Specifies whether the variable is an XSD element type. If you have defined a complex type (XSD element) in the Data type list, you should select that check box for the complex type element to be generated. The value of the data type is the name of the element prefixed by the namespace.
Constant	Specifies whether the variable is constant or not during the execution of the process.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Value Tab

This tab specifies the variable value details. You can enter any appropriate information in this field, as well as open, insert and save text files.

The information you enter depends on the variable data type. For example, if you have specified a complex type, such as an XML object in the Data type list, you can enter the corresponding XML schema. Or if you have specified a simple type, such as Duration, you can enter 1 hour.

Correlation Keys (BPM)

A *correlation key* is a set of variables that is used to identify a process instance in order to route the messages that apply to it.

For example, during a flight booking process on the Web, a registered client may have put a ticket in his wish list, but needs further information before booking it. When he comes back to his wish list, the correlation key allows the retrieval of his flight ticket, so he can proceed to the payment.

A correlation key can be created in a choreography diagram with any orchestration languages.

It is associated with a process implemented by an operation (see *Operations (BPM)* on page 111). Depending on the operation type, a process (activity) can have one or two correlation keys:

- One correlation key associated with the input message received by the activity.
- One correlation key associated with the output message sent by the activity.

The Dependencies tab of the correlation key property sheet displays the list of the processes that use the correlation key for reception of messages and the list of the processes that use the correlation key for emission of messages.

Creating a Correlation Key

You can create a correlation key from the Browser or **Model** menu.

- Select **Model > Correlation Keys** to access the List of Correlation Keys, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Correlation key**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Correlation Key Properties

To view or edit a correlation key's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Variables Tab

This tab lists the variables (see *Variables (BPM)* on page 119) that define the current correlation key.

You can add variables to a correlation key to gather variables that are related to the same communication. The correlation key can then be associated with a process implemented by an operation.

Data Transformations

A *data transformation* is an object that allows you to copy data from a source container to a target container, and also to calculate the value of an expression and store it in a variable.

For example, a manufacturer may be asked the price of one of its products, which he calculates depending on one or more parameters, such as quantity, delivery location, and so on. These parameters are input data on which the manufacturer will perform a transformation, and then store the result as target data.

A data transformation can be created in a choreography diagram with any orchestration languages.

It is used in conjunction with:

- Assign activities – to design a sequence of atomic assign tasks (see *Process Properties* on page 34)
- Correlation keys – to perform a mapping between a message, and a variable, which identifies a process instance (for example, a customer ID) (see *Correlation Keys (BPM)* on page 121)

A data transformation can have one or more source containers (*Input Variable*), but has always only one target container (*Assigned Variable*). In addition, the value to be copied from the source to the target must be type-compatible.

Creating a Data Transformation

You can create a data transformation from the **Browser** or **Model** menu.

- Select **Model > Data Transformations** to access the List of Data Transformations, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Data Transformation**.

For general information about creating objects, see *Core Features Guide > The PowerDesigner Interface > Objects*.

Data Transformation Properties

To view or edit a data transformation's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Assigned variable	Specifies the variable that receives the result of the transformation. You can select an object from the list, or use the tools to the right of the list to create an object, browse the available objects, or view the properties of the currently selected object. [BPEL languages only] Specifies an organization unit to represent the partner who sends the message.
Assigned Part	[Only process languages supporting messages in variables] Specifies the message part that receives the result of the transformation. The assigned variable has to be typed by a message format. You can click the Properties tool to view the properties of the selected object.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

Transformation Tab

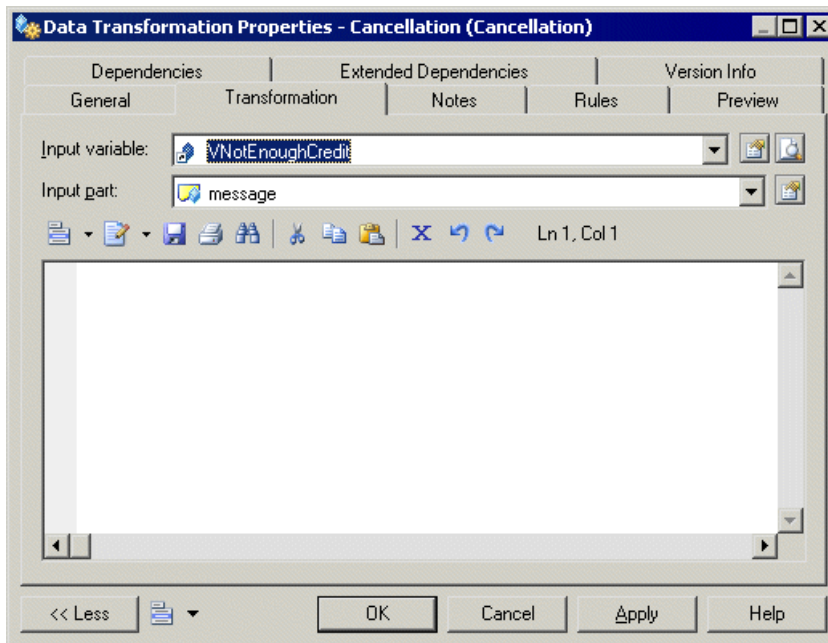
The **Transformation** tab contains the following properties:

Properties	Description
Input variable	Specifies a variable or an organization unit (to identify the partner to whom the message is sent). You can select an object from the list, or use the tools to the right of the list to browse the available objects, or view the properties of the currently selected object. If you need to specify more than one variable or organization unit as inputs, you have to use the Transformation text box and leave the Input variable list empty.
Input part	Specifies a message part when the input variable is typed by a message format. You can click the Properties tool to view the properties of the selected object.
Transformation (text box)	Specifies the transformation details using XPath language (for simple transformations) or XSLT language (for more complex transformations). You can enter any appropriate information in this field, as well as open, insert and save text files.

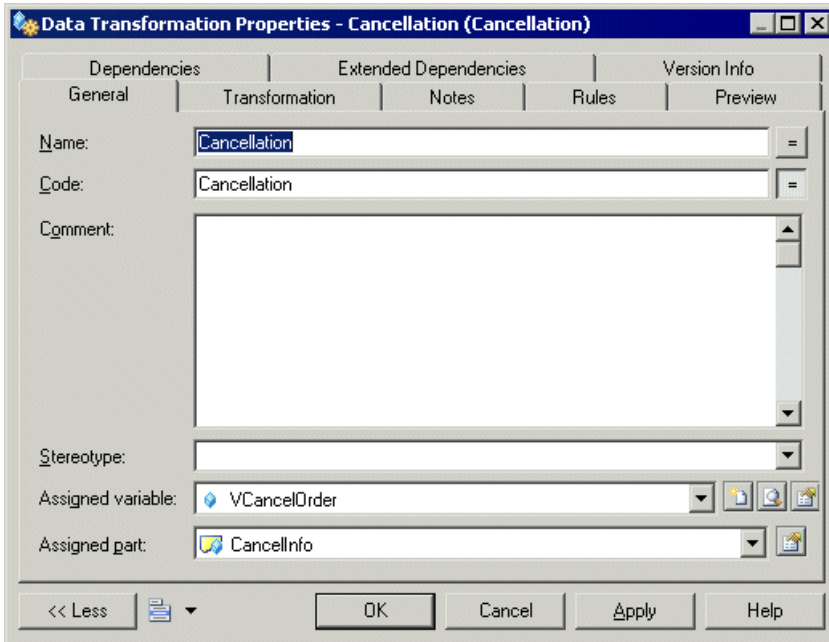
Example: Defining a Data Transformation

You define a data transformation from its property sheet by selecting:

- One or more source container (*Input variable*), which contains the source to transform and copy, in the Transformation tab. In the following example, the Cancellation data transformation contains an InputVariable and an Input part within that variable called VnotEnoughCredit and message. You can use the transformation text box to write simple transformations using XSLT or more complex transformations using Xpath. In that case the Input variable list remains empty:



- A target container (*Assigned variable*), which contains the result of the transformation in the General tab. In the following example, the Cancellation data transformation contains an assigned variable and an assigned part within that variable, called VCancelOrder and CancelInfo, because the source container also has a part (Input Part):



Building Process Service Diagrams

A *process service diagram* provides a graphical view of the services, operations, and interfaces available in your system.

Note: To create a process service diagram in an existing BPM, right-click the model in the Browser and select **New > Process Service Diagram**. You cannot create a new BPM with a process service diagram as the first diagram.

The PSD is only available with orchestration languages.





You can create and import service providers (see *Service Providers (BPM)* on page 98) in a PSD (see *Importing a Service Provider from a WSDL File* on page 103 and *Importing a Service Provider from an OOM or a PDM* on page 107).

In the following example the Process Order service provider depends on the Order Shipment service provider:



Process Service Diagram Objects

PowerDesigner supports all the objects necessary to build process service diagrams.

Object	Tool	Symbol	Description
Service provider			Service that gathers a set of service interfaces and operations. For more information, see <i>Service Providers (BPM)</i> on page 98.
Traceability Link			Unidirectional link between two service providers to specify a dependency (documentation purposes only).

Simulating a Business Process Model with SIMUL8

Simulation helps you to better understand the expected performance of your business processes before their implementation, by providing you with useful analysis metrics and assistance in business process optimization.

SIMUL8 is a flow simulation program that lets you view your process in action, by showing how its control flow moves around the organization. It can reveal key bottlenecks, over-utilized resources, or under-resourced elements of your system, and lets you fine-tune your simulation.

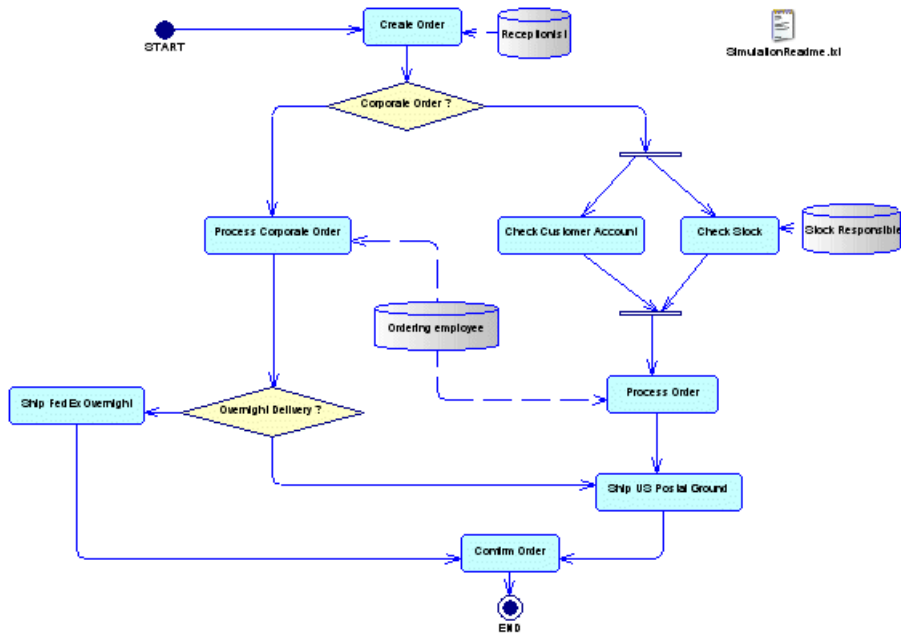
PowerDesigner supports the principal objects and parameters for SIMUL8 version 9.0 and higher.

Note: You can simulate any analysis or orchestration BPM. However, to obtain the most valuable simulation results, we recommend that you simulate Analysis language BPMs only. For more information about process languages, see *Chapter 1, Getting Started with Business Process Modeling* on page 3.

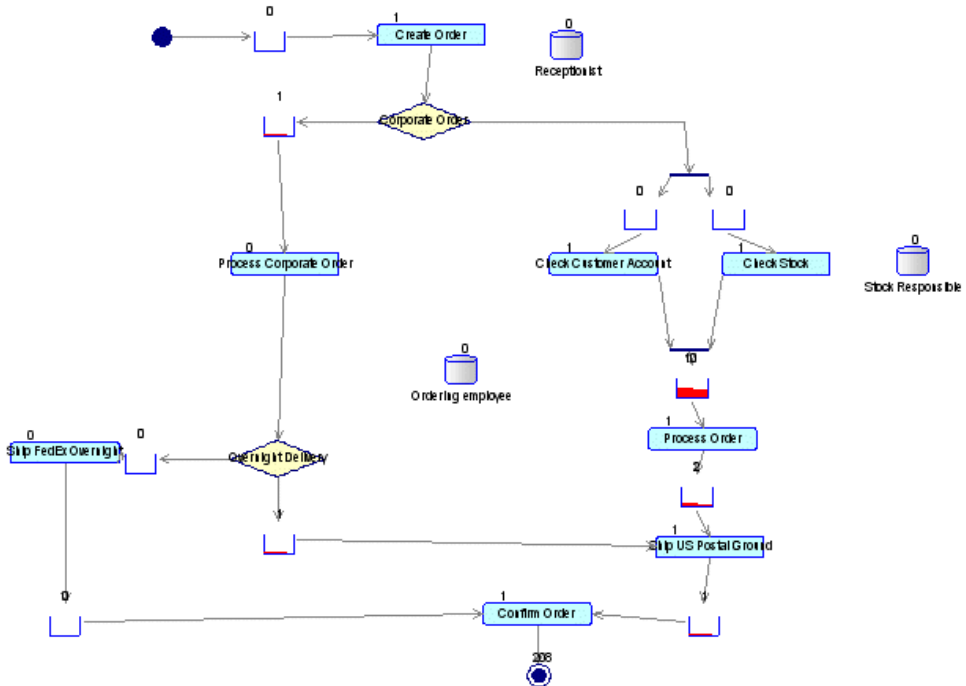
Simulation will provide more useful results when:

- The business process being analyzed is well defined and repetitive.
- An operational (logical or quantitative) decision is being taken.
- Activities and events show some interdependency and variability.
- The cost to experiment on the actual system is greater than the cost to perform a simulation.

For example, PowerDesigner ships with the following BPM (Examples directory) to model the future `Create Order` business process:



You can simulate this model with SIMUL8, and analyze your simulation results:



You rapidly see that some work items have been blocked in queues by the `Process Order` and `Process Corporate Order` work centers which are waiting for the availability of the `Ordering employee` resource. You can open the `SimulationReadme.txt` file to review how you can adjust the simulation parameters to remove work items blocked in queues.

Modeling for Simulation

PowerDesigner supports the modeling of the principal objects required to simulate your BPM with SIMUL8.

To simulate your business process, we recommend that you use the following workflow:

1. Create an analysis BPM with a business process diagram, and attach the SIMUL8 extension to the new model (see *Creating a BPM* on page 5).
2. Populate your diagram by creating a choreography of objects, such as processes, resources, flows, decisions, etc. (see *Choreography Diagram Objects* on page 30).
3. [optional] Review each object's default simulation properties (see *Reviewing SIMUL8 Default Properties* on page 134), and edit them if appropriate.

4. Export your BPM diagram to SIMUL8, and then run a simulation (see *Exporting a BPM to SIMUL8* on page 134).
5. Analyze the simulation results, and if necessary, edit simulation properties for certain objects, and run the simulation again (see *Analyzing Results and Fine-Tuning the Simulation* on page 135). You can repeat this step until you are satisfied with your simulation results.
6. Re-import your simulated BPM to PowerDesigner to synchronize the SIMUL8 changes with your BPM, and, if necessary, create any appropriate modeling objects (see *Synchronizing SIMUL8 Changes back to PowerDesigner* on page 137). You can repeat steps 4 to 6 until you are satisfied with your BPM.
7. [optional] Generate code for an orchestration engine, such as BPEL4WS or SOA to model the implementation of your processes (see *Generating Process Language Files from a BPM* on page 169).

BPM / SIMUL8 objects conversion

PowerDesigner exports and imports objects to and from SIMUL8 as follows:

BPM object and properties	SIMUL8 object and properties
Diagram with Window color display preference	<i>Model</i> with Fill color property.
[No equivalent in BPM]	<i>Work item</i> - specifies the work which is performed in the organization being simulated. For example patients in a hospital, invoices in an Accounts department.
Atomic process (see <i>Processes (BPM)</i> on page 32)	<i>Work center</i> - specifies the place where the work is performed (see <i>SIMUL8 Work Center Properties</i> on page 138).
Composite process (see <i>Processes (BPM)</i> on page 32)	<i>Component</i> - specifies a single object containing one or more existing standard objects or other components.
Implemented by process (see <i>Processes (BPM)</i> on page 32)	<i>Component</i> [if the process is implemented by a composite process] or <i>work center</i> .
Resource (see <i>Resources (BPM)</i> on page 93)	<i>Resource</i> - are required at work centers in order for the work center to work on a work item (see <i>SIMUL8 Resource Properties</i> on page 140).

BPM object and properties	SIMUL8 object and properties
Start (see <i>Starts (BPM)</i> on page 62)	<p>Can be either a :</p> <ul style="list-style-type: none"> • <i>Work entry point</i> - specifies a place where work to be done appears in your simulation for the first time (see <i>SIMUL8 Work Entry Point Properties</i> on page 141). • <i>Work center</i> with a Zero working time [if the start is contained in a composite process] see (<i>SIMUL8 Work Center Properties</i> on page 138).
End (see <i>Ends (BPM)</i> on page 63)	<p>Can be either a :</p> <ul style="list-style-type: none"> • <i>Work exit point</i> - specifies a place where work that is complete leaves your simulation (see <i>SIMUL8 Work Exit Point Properties</i> on page 142). • <i>Work center</i> with a Zero working time [if the end is contained in a composite process] (see <i>SIMUL8 Work Center Properties</i> on page 138).
Flow (<i>Flows (BPM)</i> on page 70)	<i>Routing in/out</i> properties of a work center - specifies the path taken by each individual work item through the simulation (see <i>SIMUL8 Route Properties</i> on page 143).
[No equivalent in BPM]	<i>Queue</i> - specifies a place where work to be done can wait until appropriate resources or work centers are available to process it. Properties of queues are imported to the flow that contains the queue (see <i>SIMUL8 Route Properties</i> on page 143). A queue is generated for each link between SIMUL8 objects, except for work centers generated from decisions or synchronizations.
Resource flow (see <i>Resource Flows (BPM)</i> on page 95) with Read access property	<i>Required resource property</i> of a work center - specifies a resource that must be available before a work center can start processing a work item (see <i>SIMUL8 Required Resource Properties</i> on page 140).
Decision (see <i>Decisions (BPM)</i> on page 65)	<i>Work center</i> [without queue] (see <i>SIMUL8 Work Center Properties</i> on page 138).
Synchronization (see <i>Synchronizations (BPM)</i> on page 68)	<i>Work center</i> [without queue] (see <i>SIMUL8 Work Center Properties</i> on page 138).

Note:

- PowerDesigner free symbols are preserved in SIMUL8.

- Organization units, files, packages, message formats, parts and data are not supported in SIMUL8.

For more information about SIMUL8, see: <http://www.SIMUL8.com>.

Reviewing SIMUL8 Default Properties

PowerDesigner provides default values for simulation properties that allow you to rapidly simulate your BPM.

If you need to customize some of the default simulation properties to suit your personal needs, we recommend that you begin with the following:

Domain	Property to review
Time unit and processing time	[diagram] Time unit, Simulation running time [process] Duration
Declaration and assignment of resources	[resource] Available number [process] Priority
Probability estimation on conditional flows	[flow] Routing out percent
Cost/revenue estimation	[all objects] Finance. The SIMUL8 Professional Profit plug-in allows you to add financial information to your simulation. At the end of the simulation, select Finance > Income Statement to display the financial results of your model.

Simulating a BPM

You can simulate one or more business process diagrams by exporting each one of them to a SIMUL8 model, and running a simulation. You can analyze the simulation results, and use them to adjust simulation parameters. You can then re-import the simulated model in PowerDesigner to synchronize your changes with your BPM where you can create additional modeling objects.

Exporting a BPM to SIMUL8

You can export your BPM to SIMUL8 to run a simulation of your model, and analyze the results. When you export a BPM to SIMUL8, you generate one .XS8 file for each BPM diagram selected.

1. Select **Tools > Simulation > Export SIMUL8 File** to open a standard generation dialog.
2. Specify a directory in which to generate the SIMUL8 file.

3. [optional] Select the **Check Model** option to verify the validity of your model before generation.
4. Select one or more diagrams to generate from the Business Process Diagrams sub-tab. Each diagram you select is generated as a separate .XS8 file.
5. [optional] Click the **Generated Files** tab, and specify which files will be generated. By default, all files are generated, and PowerDesigner remembers for any subsequent generation the changes you make.
6. [optional] Click the **Tasks** tab, and select the **Open the first SIMUL8 model in SIMUL8** option, if you want the first SIMUL8 model to open automatically after you close the Generated Files dialog.
7. Click **OK** to generate.

A Progress box is displayed, and the SIMUL8 files are generated in the destination directory. The Generated Files dialog opens to display the generated .XS8 files.

8. Select an .XS8 file, and click the **Edit** button to open the file in the main SIMUL8 simulation window (if you have selected the Open the first SIMUL8 model in SIMUL8 option, you can close the dialog, and the first SIMUL8 model will open automatically).
9. Click the **Run** tool in the SIMUL8 toolbar to run the simulation, and then analyze the simulation results (see *Analyzing Results and Fine-Tuning the Simulation* on page 135).
The process control flow moves around the organization, and can reveal any bottlenecks, over-utilized resources, or under-resourced elements. The clock in the corner of the window shows the passage of time.

Note: When you export a diagram contained in a hierarchy of packages, the hierarchy is preserved in the Windows Explorer.

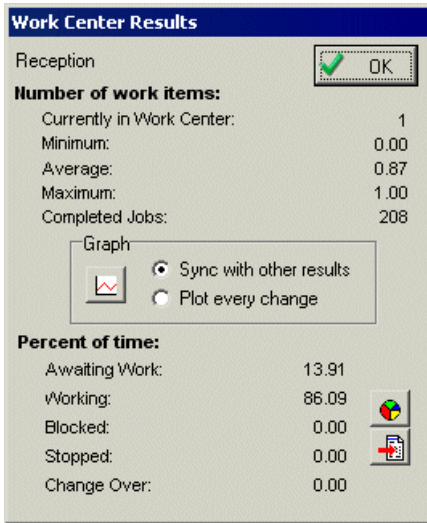
Analyzing Results and Fine-Tuning the Simulation

Simulating your BPM can provide you with information to answer "what if" questions regarding your system. You can analyze the simulation results in a variety of formats, and then open simulation objects property sheet to adjust your simulation parameters, and suggest improvements in your business process.

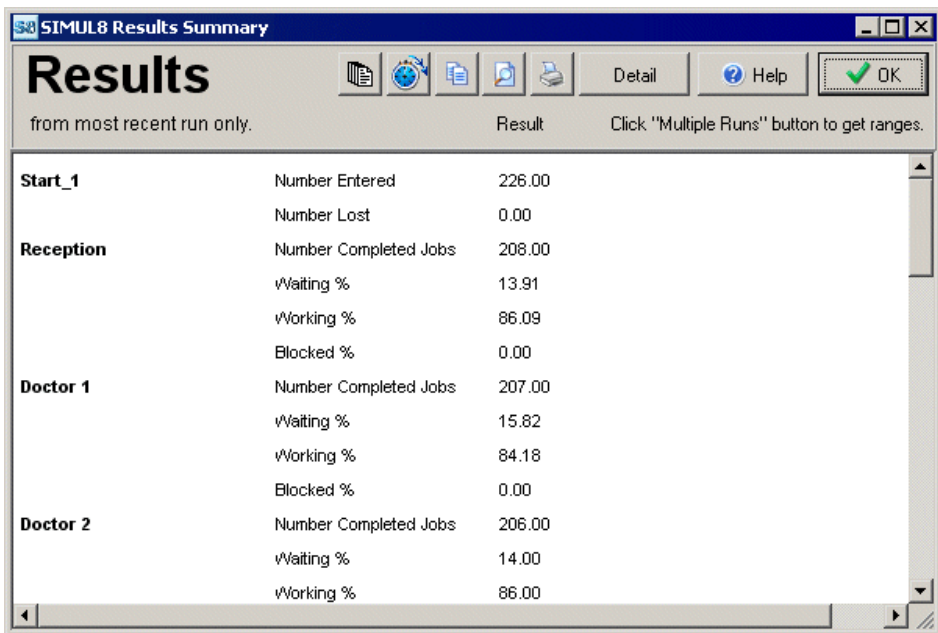
Results Analysis

You can analyze your simulation results in SIMUL8 in a variety of ways, depending on the information you need. You can:

- Display results per object - open a simulation object property sheet, and click the **Results** button, or select **Results > Object Type** :



- Display a results summary of a set of objects at the end of each simulation run - select **Results > Results Summary**:



- Export results to a text or Excel file - click the **Copy** tool in the Results Summary dialog, and paste the copied data into a text or Excel file. You can also select **Results > Results Export** to export results summary data to a number of applications.

Simulation Fine-Tuning

Depending on your simulation results, you may have to adjust parameters in the simulation objects' property sheets, and re-run your simulation.

Once your fine-tuning is complete, you must save your simulation model as an .XS8 file to enable you to import it back to PowerDesigner (see *Synchronizing SIMUL8 Changes back to PowerDesigner* on page 137).

Synchronizing SIMUL8 Changes Back to PowerDesigner

Your BPM should be the basis of your modeling work. When you export your BPM to SIMUL8, you should only change simulation parameters in SIMUL8. If your simulation results lead you to create new work centers or resources, or to modify the control flow of your business process in another way, you should always perform these changes in PowerDesigner.

1. Select **Tools > Simulation > Import SIMUL8 File** to open the Import SIMUL8 Files dialog.
2. Click the **Add** button, select the .XS8 file to import from the standard Open dialog, and click **Open** to return to the Import SIMUL8 Files dialog.

You can multi-select files to import using the **Ctrl** or **Shift** keys. All files will be imported in the same BPM.

3. Click **OK** to close the Import SIMUL8 Files dialog.

The import begins, and the Merge Models dialog opens to let you control the differences between your BPM and the imported SIMUL8 model.

For more information about merging models, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

4. Click **OK** to close the dialog.

Your synchronized BPM opens in the diagram window.

5. Update your BPM as necessary in response to the simulation results. For example, you may decide to create additional processes or resources.

Recovering a BPM from a SIMUL8 file

You can recover a BPM from a SIMUL8 file by importing the SIMUL8 model, saved as an .XS8 file, into a new analysis BPM diagram. This can be because you no longer have the BPM used to generate the SIMUL8 model, or because you have a SIMUL8 model, and decide to perform your modeling work in PowerDesigner.

1. Select **File > Import > SIMUL8 File** to open the New Business Process Model dialog.
2. Select the Analysis process language, and click the **Share** radio button.
3. [optional] Click the **Select Extensions** tab, and select any extensions to attach to your new BPM.

4. Click **OK** to open the Import SIMUL8 file dialog.
5. Click the **Add** button, select a SIMUL8 file to import, and click **Open** to return to the Import SIMUL8 file dialog.

You can multi-select files to import using the **Ctrl** or **Shift** keys. All files will be imported in the same BPM.

6. Click **OK** to start the import process. When the import is complete, your BPM diagram opens in the diagram window, and you can continue to model your business processes.

SIMUL8 Object Properties

When preparing a BPM for simulation, you use standard BPM objects with additional properties.

SIMUL8 Work Center Properties

SIMUL8 work center property sheets contain all the standard process tabs, along with the Simulation tab.

The following simulation properties apply to atomic and reuse processes only:

Name	Description
Replicate	Specifies an alternative number of processes which perform the same tasks. Using the Replicate number is a way to copy the process. Default value: 1 Scripting name: Replicate
Capital cost	Specifies the accumulated data on the flows of money at the end of simulation. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement . Default value: 0 Scripting name: FinanceCapitalCost
Cost by time unit	Specifies the usage cost of the process by time unit. Default value: 0 Scripting name: FinanceCostByTimeUnit
Cost by unit	Specifies the usage cost of the process by work unit. Default value: 0 Scripting name: FinanceCostByUnit

Name	Description
Priority	<p>Specifies that the process with the highest priority will be given the resource first (from 0 to 100), if two processes both require the same resource before they can start working.</p> <p>Default value: 50%</p> <p>Scripting name: ResourcePriority</p>
Release	<p>[If unchecked] The resource must wait for the work item, if the process cannot send the work item to the next simulation object.</p> <p>Default value: true</p> <p>Scripting name: ResourceRelease</p>
Distribution	<p>Specifies a method for simulating the variations that occur in timing in the process. You can choose between one of the following values:</p> <ul style="list-style-type: none"> • Average [default] • Exponential • Fixed • Normal • Uniform <p>Scripting name: TimingDistribution</p>
Lower bound	<p>Specifies the lower bound for the uniform timing distribution type. Samples from a uniform distribution are equally spread between the lower bound and the upper bound.</p> <p>Default value: 10</p> <p>Scripting name: TimingBoundLower</p>
Upper bound	<p>Specifies the upper bound for the uniform timing distribution type. Samples from a uniform distribution are equally spread between the lower bound and the upper bound.</p> <p>Default value: 11</p> <p>Scripting name: TimingBoundUpper</p>
Standard deviation	<p>Specifies the normal timing distribution type. For the average distribution type, the standard deviation value is set to: average value / 4.</p> <p>Default value: 0</p> <p>Scripting name: TimingStandardDeviation</p>

SIMUL8 Required Resource Properties

The resource requirements of work centers are contained in resource flow property sheets, which contain all the standard resource flow tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Required resource	<p>Specifies the way in which the resource is used by the work center. You can choose one of the following values:</p> <ul style="list-style-type: none">• Require Release [default]- The resource must be available for the process to work, and released as soon as the task is complete.• Require Only - The resource must be available for the process to work.• Release Only - The resource is released by the work item as soon as the task is complete.• Display Only - Specifies the location of the resource when displayed at this work center. <p>Scripting name: ResourceRequire</p>
Minimum number of resources	<p>Specifies the minimum number of this type of resource required at the process. Modify the default value if you need more than one unit of this resource to perform tasks at this process.</p> <p>Default value: 1</p> <p>Scripting name: ResourceMinNumber</p>
Maximum number of resources	<p>Specifies the maximum number of this type of resource required at the process. Modify the default value to be higher than the minimum value, if the process can work faster with more resources.</p> <p>Default value: 1</p> <p>Scripting name: ResourceMaxNumber</p>

SIMUL8 Resource Properties

SIMUL8 resource property sheets contain all the standard process tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Available number	<p>Specifies the number of this type of resource used at processes to enable them to perform the work on work items.</p> <p>Default value: 10</p> <p>Scripting name: NumberAvailable</p>

Name	Description
Capital cost by unit	Specifies the capital cost by resource unit. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement . Default value: 0 Scripting name: FinanceCapitalCostByUnit
Cost by time unit and by unit	Specifies the cost by time unit and by unit. Default value: 0 Scripting name: FinanceCostByUnit ByUnitTime

SIMUL8 Work Entry Point Properties

SIMUL8 work entry point property sheets contain all the standard start tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Capital cost	Specifies the capital cost. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement . Default value: 0 Scripting name: FinanceCapitalCost
Capital cost by unit	Specifies the capital cost by work unit. Default value: 0 Scripting name: FinanceCapitalCostByUnit
Time distribution type	Specifies work feeding by using different statistical distributions. You can choose from one of the following values: <ul style="list-style-type: none"> • Exponential [default] • Average • Fixed • Normal • Uniform Scripting name: InterArrivalTimeDistribution

Name	Description
Average time	Specifies the average time between two consecutive work items (in time units). Default value: 10 Scripting name: InterArrivalTimeAverage
Lower bound time	Specifies the lower bound for the uniform timing distribution type. Samples from a uniform distribution are equally spread between the lower bound and the upper bound. Default value: 10 Scripting name: InterArrivalTimeBoundLower
Upper bound time	Specifies the upper bound for the uniform timing distribution type. Samples from a uniform distribution are equally spread between the lower bound and the upper bound. Default value: 11 Scripting name: InterArrivalTimeBoundUpper
Time standard deviation	Specifies the standard deviation for the normal timing distribution type. For the average distribution type, standard deviation value is set to: average value / 4. Default value: 0 Scripting name: InterArrivalTimeStandardDeviation

SIMUL8 Work Exit Point Properties

SIMUL8 work exit point property sheets contain all the standard end tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Halt simulation at limit	Specifies that the simulation stops when the simulation limit is reached. Default value: False Scripting name: HaltSimulationAtLimit
Simulation limit	Specifies the maximum number of work items to process when the "Halt simulation at limit" option is selected. Default value: 10000 Scripting name: SimulationLimit

Name	Description
Capital cost	Specifies the capital cost. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement . Default value: 0 Scripting name: FinanceCapitalCost
Revenue per unit	Specifies the revenue per unit. Default value: 0 Scripting name: FinanceRevenuePerUnit

SIMUL8 Route Properties

SIMUL8 route property sheets contain all the standard flow tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Routing out percent	[Decision output flow] Specifies that the work items coming out from the decision are distributed to destinations according to the specified percentage value. Default value: 100 Scripting name: RoutingOutPercent
Add queue	Specifies that a queue is added to the flow when its source is a start of the main simulation diagram to prevent the loss of work items. Default value: True Scripting name: AddQueue
Initial item count	Specifies the initial count of items in the queue at the start of the simulation run. Default value: 0 Scripting name: QueueInitialItemCount
Capacity	Specifies the maximum count of work items that can be in the queue (-1 = no limit). When the maximum count is reached, further items are blocked (stay in the objects that feed the queue). Default value: -1 Scripting name: QueueCapacity

Name	Description
Min wait time	Specifies the minimum time a work item must stay in the queue. Default value: 0 Scripting name: QueueMinWaitTime
Finance capital cost	Specifies the capital cost of the queue. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement . Default value: 0 Scripting name: QueueFinance CapitalCost
Finance cost by time unit	Specifies the usage cost of the queue by unit and time unit. Default value: 0 Scripting name: QueueFinanceCostByTimeUnit

SIMUL8 Diagram Properties

SIMUL8 diagram property sheets contain all the standard diagram tabs, along with the Simulation tab.

The **Simulation** tab contains the following properties:

Name	Description
Diagram scale	Specifies the percent scale value applied to the symbol coordinates from the top left corner of the diagram. Default value: 100 Scripting name: DiagramScale
Time unit	Specifies the time unit used for timing values in objects property sheet. For time units under seconds, decimals of units must be used (for example 0.001 = 1 millisecond). Default value: Seconds Scripting name: TimeUnit
Simulation running time	Specifies the number of time units the simulation will run while collecting results information. Default value: 2400 Scripting name: SimulationRunningTime

Name	Description
Finance currency symbol	<p>Specifies the currency used by the financial properties of objects. Use "E" for Euro. Financial results can be viewed in SIMUL8, by selecting Finance > Income Statement.</p> <p>Default value: \$</p> <p>Scripting name: FinanceCurrencySymbol</p>
Finance overhead cost	<p>Specifies fixed costs. Non-object based costs will be included in financial results, which can be viewed in SIMUL8, by selecting Finance > Income Statement.</p> <p>Default value: 0</p> <p>Scripting name: FinanceOverheadCost</p>
Finance overhead revenue	<p>Specifies fixed revenues. Non-object based revenues will be included in financial results, which can be viewed in SIMUL8, by selecting Finance > Income Statement.</p> <p>Default value: 0</p> <p>Scripting name: FinanceOverheadRevenue</p>

PART II

Working with BPMs

The chapters in this part provide information about PowerDesigner features that allow you to check, generate from, and reverse-engineer to your business process models .

The business process model is a very flexible tool, which allows you quickly to develop your model without constraints. You can check the validity of your BPM at any time.

A valid BPM conforms to the following kinds of rules:

- Each object name or code must be unique in a BPM
- Each process must have at least one input flow and at least one output flow
- Each data created in the model must be used

Note: We recommend that you check your business process model before generating code or another model from it. If the check encounters errors, generation will be stopped. The **Check model** option is enabled by default in the Generation dialog box.

You can check your model in any of the following ways:

- Press F4, or
- Select **Tools > Check Model**, or
- Right-click the diagram background and select Check Model from the contextual menu

The Check Model Parameters dialog opens, allowing you to specify the kinds of checks to perform, and the objects to apply them to. The following sections document the BPM-specific checks available by default. For information about checks made on generic objects available in all model types and for detailed information about using the Check Model Parameters dialog, see *Core Features Guide > The PowerDesigner Interface > Objects > Checking Models*.

Package Checks

PowerDesigner provides default model checks to verify the validity of packages.

Check	Description and Correction
Existence of several data with same definition object	<p>Several data should not be linked to the same definition object within the same namespace.</p> <ul style="list-style-type: none"> • Manual correction: Link the data to different definition object from the data property sheet • Automatic correction: None

Process Checks

PowerDesigner provides default model checks to verify the validity of processes.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Input or output flow missing	<p>Each process should have at least one input flow and at least one output flow.</p> <ul style="list-style-type: none"> Manual correction: Add any missing flows to the process Automatic correction: None
Composite process without start or end	<p>A composite process must contain at least one start and at least one end.</p> <ul style="list-style-type: none"> Manual correction: Add a start and an end in the sub-process diagram Automatic correction: None
Process implementation	<p>A process cannot be implemented by an implemented process.</p> <ul style="list-style-type: none"> Manual correction: Select a process which is not an implemented process Automatic correction: None
Existence of several data with the same definition object	<p>Several data should not be linked to the same definition object within the same namespace, as data can be created in a composite process.</p> <ul style="list-style-type: none"> Manual correction: Link the data to different definition object from the data property sheet Automatic correction: None

Check	Description and Correction
Process with incoherent data accesses	<p>The data attached to a flow should also be attached to the source and destination processes.</p> <ul style="list-style-type: none"> • Manual correction: Migrate the data of the flow to the source and destination processes • Automatic correction: Automatically migrates the data of a flow to the source and destination processes
Undefined data accesses	<p>The data accesses of a process should have one of the following values: Create, Read, Update, Delete.</p> <ul style="list-style-type: none"> • Manual correction: Add a data access for the data in the Data tab of the process property sheet • Automatic correction: None

Decision Checks

PowerDesigner provides default model checks to verify the validity of decisions.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> • Manual correction - Modify the duplicate name or code. • Automatic correction - Appends a number to the duplicate name or code.
Invalid decision	<p>A decision represents a conditional branch when a unique flow is split into several output flows, or it represents a merge when several input flows are merged into a unique output flow. That is why a decision must have more than one input flow or more than one output flow.</p> <ul style="list-style-type: none"> • Manual correction: Add any missing flows on the decision • Automatic correction: None

Check	Description and Correction
Event condition coherence	<p>All conditions on outgoing flows must be of the same type. Condition types can be a Boolean expression or an event. A flow defined from a decision object to a receive activity is considered as an event that corresponds to the reception of a message.</p> <ul style="list-style-type: none"> • Manual correction: Assign the same conditions to all outgoing flows • Automatic correction: None

Synchronization Checks

PowerDesigner provides default model checks to verify the validity of synchronizations.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> • Manual correction - Modify the duplicate name or code. • Automatic correction - Appends a number to the duplicate name or code.
Incomplete synchronization	<p>A synchronization represents a fork when a unique flow is split into several output flows executed in parallel, or it represents a join when several input flows are joined and they wait until all flows reach the join before continuing as a unique output flow. That is why a synchronization must have more than one input flow, or more than one output flow.</p> <ul style="list-style-type: none"> • Manual correction: Add any missing flows to the synchronization • Automatic correction: None

Flow Checks

PowerDesigner provides default model checks to verify the validity of flows.

Check	Description and Correction
Flow without source or destination	<p>A flow must have a source and a destination object.</p> <ul style="list-style-type: none"> Manual correction: Assign a source or a destination to the flow Automatic correction: None
Flow undefined message format	<p>A flow should have a defined message format or the message format set to <None>.</p> <ul style="list-style-type: none"> Manual correction: Define the message format for the flow or delete it Automatic correction: None
Flow incoherent message format	<p>The message format of a flow coming out of a composite process (child process) must also exist on the flow going to the end inside the child process. The message format of a flow coming in a composite process must also exist on the flow going out from the start inside the child process.</p> <ul style="list-style-type: none"> Manual correction: Add any missing message formats to the appropriate flows of the decomposed processes Automatic correction: None
Invalid event condition	<p>Outgoing flows from start, decision and synchronization objects cannot have fault, compensation, or signal event. Besides, outgoing flows from synchronization objects cannot have message type events. Also, a decision object can only have flows with Message, Signal or Timer event. You can mix Message and Timer events or Signal and Timer events on the same decision.</p> <ul style="list-style-type: none"> Manual correction: Removes the inappropriate Event condition on the flow or change the source extremity of the flow Automatic correction: Removes the inappropriate Event condition on the flow
Invalid exception flow destination	<p>A flow with Exception stereotype must target a process that is implemented by an operation and whose Action Type is Receive Request.</p> <ul style="list-style-type: none"> Manual correction: Change the flow stereotype or select a process that is implemented by an operation and whose Action Type is Receive Request Automatic correction: None

Resource Checks

PowerDesigner provides default model checks to verify the validity of resources.

Check	Description and Correction
Name/Code contains terms not in glossary	[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary. <ul style="list-style-type: none">• Manual correction - Modify the name or code to contain only glossary terms.• Automatic correction - None.
Name/Code contains synonyms of glossary terms	[if glossary enabled] Names and codes must not contain synonyms of glossary terms. <ul style="list-style-type: none">• Manual correction - Modify the name or code to contain only glossary terms.• Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	Object names must be unique in the namespace. <ul style="list-style-type: none">• Manual correction - Modify the duplicate name or code.• Automatic correction - Appends a number to the duplicate name or code.
Isolated resource	A resource must be linked to at least one process. <ul style="list-style-type: none">• Manual correction: Link the resource to a process• Automatic correction: None

Resource Flow Checks

PowerDesigner provides default model checks to verify the validity of resource flows.

Check	Description and Correction
Name/Code contains terms not in glossary	[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary. <ul style="list-style-type: none">• Manual correction - Modify the name or code to contain only glossary terms.• Automatic correction - None.

Check	Description and Correction
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Resource flow extremities	<p>A resource flow must always link a process to a resource or a resource to a process.</p> <ul style="list-style-type: none"> Manual correction: Assign a process and a resource to the resource flow extremities Automatic correction: None
Resource flow undefined access mode	<p>A resource flow should have a defined access mode (Read, Create, Update or Delete).</p> <ul style="list-style-type: none"> Manual correction: Assign an access mode to the resource flow Automatic correction: None

Organization Unit Checks

PowerDesigner provides default model checks to verify the validity of organization units.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.

Check	Description and Correction
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Circular dependency through parent property	<p>An organization unit cannot be the parent of itself or cannot have for parent one of its children.</p> <ul style="list-style-type: none"> Manual correction: Change the organization unit in the Parent box in the organization unit property sheet Automatic correction: None

Start Checks

PowerDesigner provides default model checks to verify the validity of starts.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Existence of output flow	<p>Each start object must have at least one output flow.</p> <ul style="list-style-type: none"> Manual correction: Create a flow from the start Automatic correction: None

End Checks

PowerDesigner provides default model checks to verify the validity of ends.

Check	Description and Correction
Name/Code contains terms not in glossary	[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary. <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	[if glossary enabled] Names and codes must not contain synonyms of glossary terms. <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	Object names must be unique in the namespace. <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Existence of input flow	Each end object must have at least one input flow. <ul style="list-style-type: none"> Manual correction: Create a flow to the end Automatic correction: None

Message Format Checks

PowerDesigner provides default model checks to verify the validity of message formats.

Check	Description and Correction
Name/Code contains terms not in glossary	[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary. <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.

Check	Description and Correction
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Message format definition uniqueness	<p>Message format definitions should be unique in the model.</p> <ul style="list-style-type: none"> Manual correction: Delete the duplicate message format definition Automatic correction: None

Data Checks

PowerDesigner provides default model checks to verify the validity of data.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.

Check	Description and Correction
Unused data	<p>The data you created is not used in the model.</p> <ul style="list-style-type: none"> Manual correction: Attach the data to an object in the model Automatic correction: None

Service Provider Checks

PowerDesigner provides default model checks to verify the validity of service providers.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Empty service provider	<p>Each service provider should own at least one service interface.</p> <ul style="list-style-type: none"> Manual correction: Create a service interface in the Interfaces tab of the service provider property sheet Automatic correction: None

Service Interface Checks

PowerDesigner provides default model checks to verify the validity of service interfaces.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Empty service interface	<p>Each service interface should own at least one operation.</p> <ul style="list-style-type: none"> Manual correction: Create an operation in the Operations tab of the service interface property sheet Automatic correction: None

Operation Checks

PowerDesigner provides default model checks to verify the validity of operations.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.

Check	Description and Correction
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Input or output message missing	<p>Depending on the type of the operation, input and/or output messages are required. A Notification operation requires an output message, a One-Way operation requires an input message, and a Request-Response or a Solicit Response operation requires both input and output messages.</p> <ul style="list-style-type: none"> Manual correction: Change the operation type to be coherent with the message definition or define the missing message Automatic correction: Updates the operation type to be coherent with the current message definition, except when both input and output messages are missing

Variable Checks

PowerDesigner provides default model checks to verify the validity of variables.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.

Check	Description and Correction
Undefined data type	<p>The variable should have a defined data type.</p> <ul style="list-style-type: none"> Manual correction: Set a correct data type for the variable in the Data Type list of its property sheet Automatic correction: None
Variable used out of scope	<p>The variable must be used in the scope where it is defined. When a variable is used in a different package or composite process from the one where it is defined, a shortcut is created. The package or composite process that owns the shortcut must be a child of the package or composite process that owns the variable object. In other cases, the variable is not visible, as it is not defined in the parent scope.</p> <ul style="list-style-type: none"> Manual correction: Move the variable under the common parent or duplicate it Automatic correction: Moves the variable under the common ascendant
Data type coherence	<p>A variable mapped to a message should be of the same type as the message.</p> <ul style="list-style-type: none"> Manual correction: Change the type of the variable to be the same as the messages to which it is mapped Automatic correction: Changes the variable type when it is mapped only once to a message or mapped several times but to the same message

Data Transformation Checks

PowerDesigner provides default model checks to verify the validity of data transformations.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.

Check	Description and Correction
Empty transformation expression	<p>The transformation expression should not be empty.</p> <ul style="list-style-type: none"> Manual correction: Define a transformation expression in the Transformation tab of the data transformation property sheet Automatic correction: None
Empty assigned variable	<p>The target variable of a transformation must not be undefined.</p> <ul style="list-style-type: none"> Manual correction: Select a variable object in the Assigned Variable list of the data transformation property sheet Automatic correction: Creates a variable object and associates it with the data transformation

Correlation Key Checks

PowerDesigner provides default model checks to verify the validity of correlation keys.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Empty list of variables	<p>A correlation key must have at least one variable.</p> <ul style="list-style-type: none"> Manual correction: Attach a variable to the correlation key Automatic correction: None

Check	Description and Correction
Correlation key used out of scope	<p>A correlation key must be used within the scope of its definition.</p> <ul style="list-style-type: none"> Manual correction: Choose only correlation keys defined under the parent scope of the process Automatic correction: Moves out of scope correlation keys to common ancestor and leaves a shortcut at the initial location
Unused correlation key	<p>The correlation key should be used by an activity.</p> <ul style="list-style-type: none"> Manual correction: Use the correlation key object in a process implemented by an operation or delete the useless correlation key Automatic correction: None

Event Checks

PowerDesigner provides default model checks to verify the validity of events.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Empty stereotype	<p>An event should have a defined stereotype.</p> <ul style="list-style-type: none"> Manual correction: Define a stereotype in the Stereotype box of the event property sheet Automatic correction: None

Choreography Task Checks

PowerDesigner provides default model checks to verify the validity of choreography tasks

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Initiating or Responding participant missing	<p>Each choreography task must have both an initiating and responding participant specified.</p> <ul style="list-style-type: none"> Manual correction: On the General tab of the choreography task property sheet, specify the missing participant. Automatic correction: None
Initiating or Responding participant not linked to related node	<p>If the choreography task is associated with a conversation node, then the participants specified on the task must be the same as those on the node.</p> <ul style="list-style-type: none"> Manual correction: On the General tab of the choreography task property sheet, change the participants associated with the task to those associated with the node. Automatic correction: None
Initiating message missing	<p>Each choreography task must have an initiating message specified.</p> <ul style="list-style-type: none"> Manual correction: On the General tab of the choreography task property sheet, specify an appropriate initiating message. Automatic correction: None

Conversation Node Checks

PowerDesigner provides default model checks to verify the validity of conversation nodes.

Check	Description and Correction
Name/Code contains terms not in glossary	<p>[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - None.
Name/Code contains synonyms of glossary terms	<p>[if glossary enabled] Names and codes must not contain synonyms of glossary terms.</p> <ul style="list-style-type: none"> Manual correction - Modify the name or code to contain only glossary terms. Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	<p>Object names must be unique in the namespace.</p> <ul style="list-style-type: none"> Manual correction - Modify the duplicate name or code. Automatic correction - Appends a number to the duplicate name or code.
Inconsistent participants with parent node	<p>A sub-node must not be attached to participants that are not attached to its parent node.</p> <ul style="list-style-type: none"> Manual correction: Change the participants on the child node to those of the parent node. Automatic correction: Changes the participants on the child node to those of the parent node.
Correlation key missing	<p>Each conversation node must have a correlation key specified.</p> <ul style="list-style-type: none"> Manual correction: Specify a correlation key in the conversation node property sheet. Automatic correction: None

Communication Link Checks

PowerDesigner provides default model checks to verify the validity of communication links.

Check	Description and Correction
Name/Code contains terms not in glossary	[if glossary enabled] Names and codes must contain only approved terms drawn from the glossary. <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - None.
Name/Code contains synonyms of glossary terms	[if glossary enabled] Names and codes must not contain synonyms of glossary terms. <ul style="list-style-type: none"> • Manual correction - Modify the name or code to contain only glossary terms. • Automatic correction - Replaces synonyms with their associated glossary terms.
Name/Code uniqueness	Object names must be unique in the namespace. <ul style="list-style-type: none"> • Manual correction - Modify the duplicate name or code. • Automatic correction - Appends a number to the duplicate name or code.

Generating and Reverse Engineering Process Languages

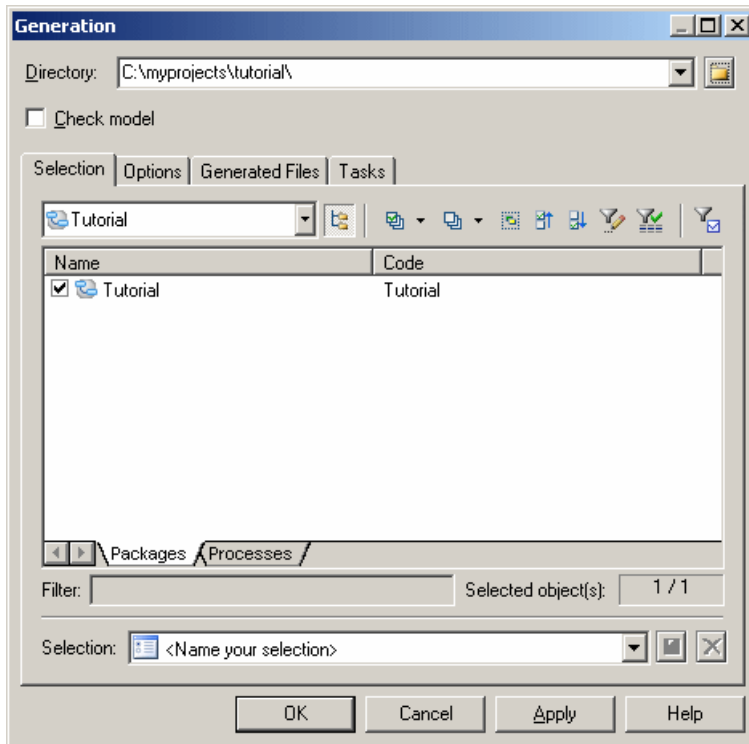
PowerDesigner can generate objects from a BPM and reverse engineer into an existing or new BPM.

Generating Process Language Files from a BPM

PowerDesigner supports the generation of the process language files.

Process language	What is generated
Analysis	No files generated as this language is used for modeling purposes only
BPMN 1.0	No files generated as this language is used for modeling purposes only
ebXML BPSS v 1.01 ebXML BPSS v1.04	.XML files
BPEL4WS 1.1 WS-BPEL 2.0	.XML files, .WSDL files
Sybase WorkSpace Business Process 2.x	.XML files, .XSD files, svc_xyz (service) files
Service Oriented Architecture	No files generated as this language is used to design the orchestration of processes without being linked to any particular platform or orchestration language

1. Select **Language > Generate *language* Code** to open the Generation dialog:



2. Enter a directory in which to generate the files, and specify whether you want to perform a model check (see *Chapter 6, Checking a BPM* on page 149).
3. [optional] Click the **Selection** tab and specify the objects that you want to generate from. By default, all objects are generated.
4. [optional] Click the **Options** tab and set any necessary generation options. For more information about these options, see the appropriate language chapter.

For information about editing the options that appear on this and the **Tasks** tab, see *Customizing and Extending PowerDesigner > Object, Process, and XML Language Definition Files*.

5. [optional] Click the **Generated Files** tab and specify which files will be generated. By default, all files are generated.

For information about customizing the files that will be generated, see *Customizing and Extending PowerDesigner > Extension Files > Templates and Generated Files (Profile)*.

6. [optional] Click the **Tasks** tab and specify any additional generation tasks to perform. For more information about these tasks, see the appropriate language chapter.
7. Click **OK** to begin generation.

A Progress box is displayed. The Result list displays the files that you can edit. The result is also displayed in the Generation tab of the Output window, located in the bottom part of the main window.

All files are generated in the destination directory.

Note: You can attach an extension file (.XEM) to your model to extend the generation process. For more information, see *Customizing and Extending PowerDesigner > Extension Files*.

Reverse Engineering Source Files into a BPM

Reverse engineering is the process of examining and recovering source code from a file that is then used to build or update a BPM.

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 the existing object, or to keep the existing object in the model.

You can reverse the following types of files into a BPM:

Types of file you can reverse	Subfamily	Family
ebXML BPSS 1.01 and 1.04 (Business Process Specification Schema)	ebXML	Collaborative
BPEL, WSDL	BPEL4WS WS-BPEL	Service Orchestration

Reverse Engineering into a New BPM

You can reverse engineer process language files into a new BPM.

1. Select **File > Reverse Engineer > Process Language** to display the New Business Process Model dialog box.
2. Select a process language in the list and click the **Share** radio button.
3. [optional] Click the **Select Extensions** tab, and select any extensions you want to attach to the new model.
4. Click **OK** to go to the appropriate, language-specific Reverse Engineering window. For ebXML, a standard file selection box is displayed. For detailed information, see the appropriate language chapter.
5. Click **OK** to start reverse engineering.

A progress box is displayed. The processes are added to your model.

Reverse Engineering into an Existing BPM

You can reverse engineer process language files into an existing BPM.

1. Select **Language > Reverse Engineer a file** to display the Reverse Engineer dialog box.
2. [not ebXML] Select to reverse engineer files or directories from the Reverse Engineering list.
3. [not ebXML] Click the Add button in the Selection tab to display a standard Open dialog box.
4. Select the files or directory you want to reverse and click Open.

Note: For those languages that support multi-selection, you select several files simultaneously using the **Ctrl** or **Shift** keys. You cannot select several directories.

The Reverse Engineer dialog box displays the files you selected.

5. [optional-BPEL4WS or WS-BPEL] Click the Options tab and specify any appropriate options.
6. Click OK to begin reverse engineering.

A message in the Output window indicates that the specified file is fully reverse engineered and the Merge Models window opens.

7. Review the objects that you will be importing, and the changes that they will make to the model.

For more information on merging models, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

8. Click OK to merge the selected changes into your model.

Generating Other Models from a BPM

You can generate another BPM from your BPM. You can use the BPM to BPM generation to generate an analysis or collaborative BPM into an implementation BPM designed for the different orchestration process languages supported in PowerDesigner (BPEL4WS, WS-BPEL, and SOA).

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. For example, once the Analyst team has designed the analysis model, the model can be submitted to the Development team for implementation.

1. Select **Tools > Generate Business Process Model (Ctrl+Shift+B)** to open the BPM Generation Options Window:
2. On the General tab, select a radio button to generate a new or update an existing model, and complete the appropriate options.
3. [optional] Click the Detail tab and set any appropriate options. We recommend that you select the Check model checkbox to check the model for errors and warnings before generation.
4. [optional] Click the Target Models tab and specify the target models for any generated shortcuts.
5. [optional] Click the Selection tab and select or deselect objects to generate.
6. Click OK to begin generation.

Note: For detailed information about the options available on the various tabs of the Generation window, see *Core Features Guide > Linking and Synchronizing Models > Generating Models and Model Objects*.

Generating an Orchestration BPM

You can generate an orchestration BPM from an analysis, a collaborative or an orchestration BPM in any of the following ways:

- Use the generation feature (see *Chapter 8, Generating Other Models from a BPM* on page 173).
- Use the change target language feature – replaces the content of your current model by performing the appropriate transformations on the whole model, preventing you from

doing any object selection. (For more information, see *Changing the Process Language of a BPM* on page 12).

When generating an orchestration BPM, some transformations are performed on the model to make it compliant with the new target language. These transformations are logged into the Output window.

Generating an Orchestration BPM from an Analysis BPM

The following transformations are executed when you generate an orchestration BPM from an analysis BPM:

- **Creation of top-level processes** - The orchestration BPM requires a top-level process. When a graph of activities is defined under a package or a model, a top-level process is automatically created and the whole graph of activities is moved under it. For each unrelated set of activities, a top-level process is created. An activity (start, end, process, decision, and synchronization) is related to another one if a flow exists between them or if they are displayed in the same diagram. The diagrams are also moved under the composite process and their contents are preserved.
- **Merge of multiple start objects** - The orchestration BPM does not support multiple start objects in the composite process. All start objects that appear under composite processes and all start objects that appear simultaneously in the same top-level diagram are merged to become one.
- **Shortcut of process in graphs** - A shortcut can be used in an analysis BPM to reuse an existing process. In an orchestration BPM, such a shortcut is replaced with a duplication of the target object, as processes implemented by another process are not supported. For orchestration languages that do not support process reusability, the call of a reusable process is replaced with a duplication of the process. If the reusable process is an unloaded external shortcut, the activity process that calls the shortcut is preserved and detached from the shortcut.
- **Message format on flows** - The orchestration BPM does not support the association of a message format with a flow, as the exchange of information is no longer managed by flows in this type of BPM view. All message formats are automatically detached from flows.
- **Flow type on flows** - In an analysis BPM, a flow can have one of the following flow types: Success, Timeout, Technical Error and Business Error. The Timeout, Technical Error and Business Error flow types are replaced with event objects with Timer or Fault stereotype. Event objects are associated with the flow to define event handlers.
- **Data** - Data objects are automatically replaced with variables. The data attachment to flow or message format is lost.

Generating an Orchestration BPM from a Collaborative BPM

You can generate an orchestration BPM from a collaborative BPM, in order to focus on the implementation of one side of the collaboration you have defined in the collaborative BPM.

The following transformations are executed when you generate an orchestration BPM from a collaborative BPM (in addition to those listed in *Generating an orchestration BPM from an analysis BPM* on page 174):

- **Business Transaction** - A Business Transaction represents a simple exchange of information between two partners. It is automatically replaced with an operation in the orchestration BPM. All operations are created under the same service provider and the same service interface. The RequestDocument message format is considered as the input message, and the ResponseDocument message format (if any) is considered as the output document. The Receipt and Acceptance Acknowledgements are lost.
- **Organization unit and Binary Collaboration** - A Binary Collaboration stores the choreography of exchanges between the two partners (organization units) in a collaborative BPM. Binary Collaborations are converted into two top-level processes, which represent the collaboration implementation from the Responding Role point of view, and from the Initiating point of view. If you want to generate one process only, you must deselect an organization unit in the Selection page of the generation dialog box. The choreography described inside the Binary Collaboration process is duplicated in each top-level process. Each Business Transaction Activity is converted into a process implemented by the operation that matches the Business Transaction. The process receives messages from partners if the Business Transaction was associated with the organization unit that corresponds to the top-level process. Otherwise the process sends messages to partners.
- **MultiParty Collaboration** - The MultiParty Collaboration is not generated. Only Binary Collaborations defined inside the MultiParty Collaboration are generated following the rules described in section Organization unit and Binary Collaboration.

Generating an Orchestration BPM from an Orchestration BPM

You can generate an orchestration BPM from another orchestration BPM in order to execute your BPM in a specific application or to use a language standard format. Most of the time you will perform the following types of generation:

- Reversed BPEL file > Sybase WorkSpace Business Process BPM
- SOA > Sybase WorkSpace Business Process or BPEL

The following transformations are performed when you generate a Sybase WorkSpace Business Process BPM from an orchestration BPM (SOA or any of the BPEL languages):

- Operation - Each language can restrict the type of operations that can be attached to a process. In Sybase WorkSpace Business Process, the following operation types are supported:
 - One-Way
 - Request-Response operation
- Top-level process - In Sybase WorkSpace Business Process, the top-level process is not supported. The flow chart is moved to the top-level diagram and must contain at least a start, a process and an end. In case of multiple top-level processes, these are all associated with the same start and end.
- Package - In Sybase WorkSpace Business Process, packages are not supported, as the model can contain only one process.
- Additional transformations - The following additional transformations are performed from a BPEL BPM to a Sybase WorkSpace Business Process BPM:

BPEL	WorkSpace Business Process
Flow with <<link>> stereotype.	Not supported.
Single process with multiple output flows.	Split between the single process and the multiple output flows.
Single process with multiple input flows.	Join between the single process and the multiple input flows.
Start with multiple output flows.	Choice between the start and the multiple output flows.
End with multiple input flows.	Join between the end and the multiple input flows.
"Receive request and reply" action type for operation.	"Receive request" action type for operation.
Switch decision.	SingleRule decision with "If-Then-Else" expression.
Event handler (flow chart of processes to handle an event).	Composite process (containing the flow chart) with appropriate event handler stereotype.

Changing Target from Analysis to Data Flow Diagram

You use the Change Current Process Language command from the Language menu in order to generate a data flow diagram from an Analysis business process diagram, when you need to represent the processes, at a high level of abstraction, in your information system from the viewpoint of data.

It automatically replaces the content of your current model by performing the appropriate transformations on the whole model, preventing you from doing any object selection.

For more information on the Change Current Process Language command, see *Changing the Process Language of a BPM* on page 12.

The following transformations are executed when you generate a data flow diagram from an Analysis business process diagram:

- Decision objects are replaced with split/merge objects
- Organization unit objects are replaced with external entity objects
- Resource objects are replaced with data stores
- Organization unit objects in top-level diagrams linked to processes using role association objects are replaced with external entity objects
- Composite processes with start and end objects are replaced with external entity objects or process shortcut, depending on the input and output flows of the composite process. Data items are preserved

Importing Visio Diagrams into PowerDesigner

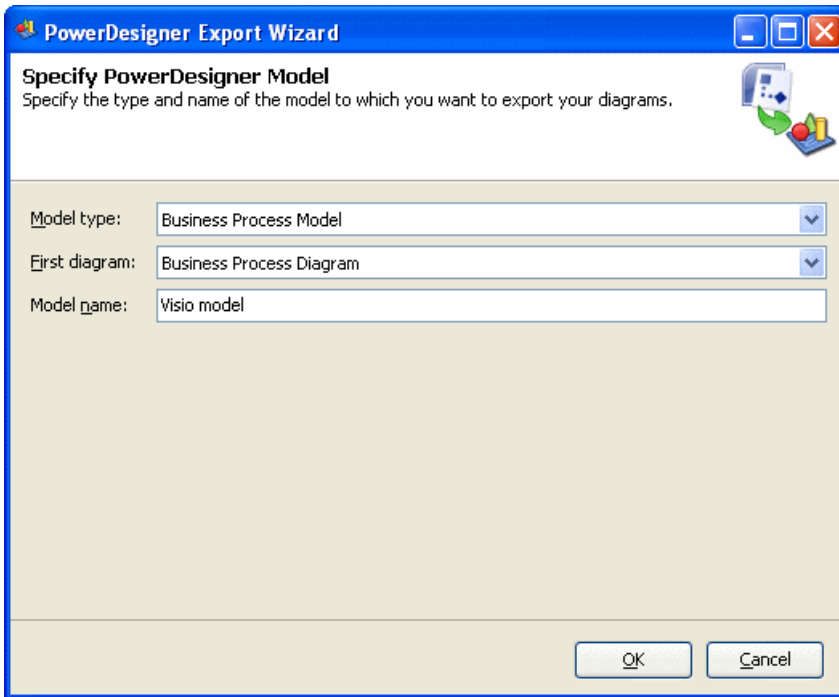
Importing your Visio diagrams into PowerDesigner's rich metadata environment enables you to link your architectural objects with the objects that will implement them, and to profit from PowerDesigner's powerful impact and lineage analysis features. You must have installed Visio 2002 or higher and have selected to install the Visio plug-in from the PowerDesigner installer.

Note: Only Visio diagrams created from the following standard templates can be imported into PowerDesigner, and only objects available on the standard stencils will be imported. Custom properties will be imported as extended attributes.

You can import the following diagrams into a PowerDesigner BPM or EAM:

Visio Template	PowerDesigner Diagram
Audit Diagram	BPM Analysis/ Business Process Diagram
Basic Flowchart	BPM Analysis/ Business Process Diagram
Cross-Functional Flowchart	BPM Analysis/ Business Process Diagram
Business Process/ Data Flow Diagram Software/ Data Flow Diagram	BPM Data Flow Diagram
Event Driven Process Chain Diagram	BPM Business Process Diagram
ITIL Diagram	BPM Business Process Diagram
Work Flow Diagram	BPM Business Process Diagram
Flowchart/ SDL Diagram	BPM Business Process Diagram
Organization Chart	EAM Organization Chart Diagram
Software/ Enterprise Application	EAM Application Architecture Diagram
Network/ Basic Network / Detailed Network Diagram	EAM Technology Infrastructure Diagram
Active Directory	EAM Organization Chart Diagram
LDAP Directory	EAM Organization Chart Diagram

1. Open your diagram in Visio and select **PowerDesigner > Export to PowerDesigner Model** to open the PowerDesigner Export wizard:



2. Specify the type of model to which you want to export your diagram, enter a name for the model to be created, and then click **OK** to start the export.
3. When the export is complete, click **OK** to close the wizard.

The diagram is opened as a new BPM or EAM in PowerDesigner.

PART III

Process Language Definition Reference

The chapters in this part provide information specific to the business process languages supported by PowerDesigner.

Business Process Modeling Notation (BPMN)

Business Process Modeling Notation (BPMN) 2.0 from the Object Management Group (OMG) is a standardized graphical notation for modeling business processes. It is intended to provide a notation that is readily understandable by all business users (including business analysts, technical developers, and those who will manage and monitor the processes after implementation) and to create a standardized bridge between business process design and XML-based business execution languages, such as BPEL4WS and Sybase Integration Orchestrator.

BPMN 2.0 provides the following diagrams:

- Conversation diagrams - which provide an overview of the communications between participants.
- Choreography diagrams - which focus on the detail of the conversation between two or more participants, and which are often linked to specific conversation nodes.
- Collaboration diagrams - which focus on the messages that pass between participants. You can show participants as black boxes or with processes inside them. PowerDesigner supports collaboration diagrams as standard business process diagrams with a BPMN-specific toolbox.
- Process diagrams - which focus on the sequence flow in a single process in a participant. PowerDesigner supports process diagrams as standard business process diagrams with a BPMN-specific toolbox.

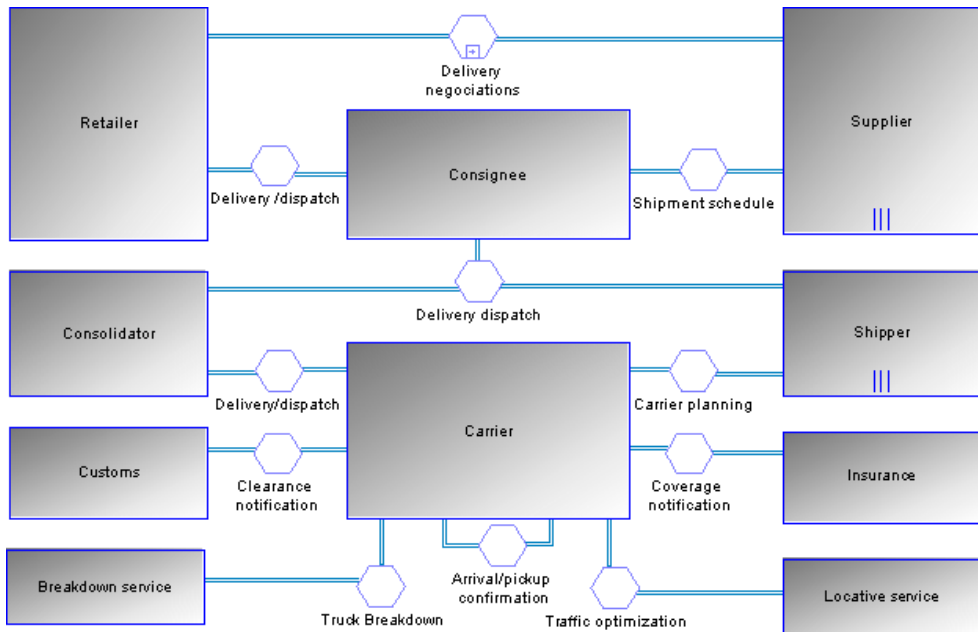
Note: When creating reports for your BPMN 2.0 models, we recommend that you start from one of the BPMN 2.0 report templates, which provide a framework for organizing all the extensions in logical groups.

We recommend that you perform a model check on your completed model (or after major changes) to verify the validity of your diagrams.

Conversation Diagrams (BPMN)




A conversation diagram focuses on the communications between participants. You cannot create or display processes or choreographies in this diagram.

In the following example, the various conversations associated with deliveries from a supplier to a retailer are analyzed:



Note: PowerDesigner does not support the display of processes within participant symbols in a conversation diagram.

The following tools are available in this diagram:

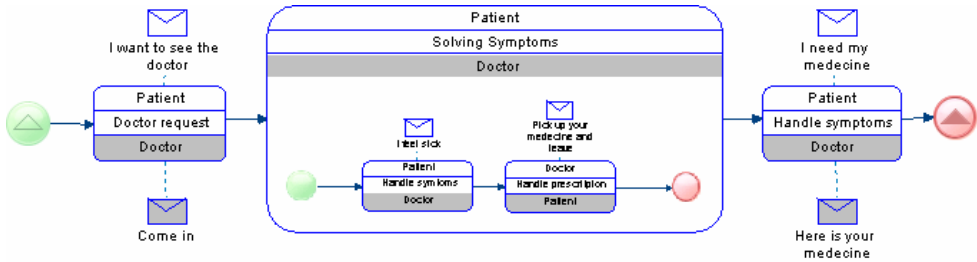
Tool	Description
	Participant - Organization, business unit, or role (see <i>Participants (BPMN)</i> on page 189).
	Conversation Node - Sits between two participants and collects the messages exchanged between them (see <i>Conversation Nodes (BPMN)</i> on page 189).
	Conversation Link - Links participants via a conversation node (see <i>Flows and Links (BPMN)</i> on page 197). Click in one participant and draw a link to another participant to automatically create a conversation node between them.

Choreography Diagrams (BPMN)

A choreography diagram is used to analyze how participants exchange information to coordinate their interactions. A choreography diagram can be used to expand and analyze in detail the exchange of messages associated with a conversation node in a conversation diagram.








In the following example, the exchange of messages between a patient and a doctor is analyzed:

CHAPTER 10: Business Process Modeling Notation (BPMN)



Note: PowerDesigner does not support the display of participant swimlanes nor the display of collaboration diagrams within choreography tasks. In addition, you cannot create intermediate events in choreography diagrams, and only one initiating and one responding participant are supported for atomic choreography tasks (though multiple participants are calculated for a task containing sub-tasks).

The following tools are available in this diagram:

Tool	Description
 ...	Start Events - Initiate a process (see <i>Events (BPMN)</i> on page 191). The various types of start events each have their own tools.
 ...	End Events - Conclude a process (see <i>Events (BPMN)</i> on page 191). The various types of end events each have their own tools.
	Choreography Task - Interaction between two participants (see <i>Choreography Tasks (BPMN)</i> on page 190).
 ...	Gateways - Merge or split the sequence flow (see <i>Gateways (BPMN)</i> on page 194). The various types of gateways each have their own tools.
	Message - Message sent to the choreography task by a participant (see <i>Messages (BPMN)</i> on page 197). If your task has participants specified, you can click on the task symbol to create a message and message flow in one step.
	Message Flow - Links a message to a participant in the choreography task (see <i>Flows and Links (BPMN)</i> on page 197). The task must have participants defined before you can attach a message to it.
	Sequence Flow - Links two elements (events, activities, gateways) in a process (see <i>Flows and Links (BPMN)</i> on page 197).

Note: You can change the type of an event or activity by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Associating a Conversation Node with a Choreography Diagram or Task

You can associate a conversation node with a choreography diagram or with a choreography task in order to model the choreography of the messages that flow through it. Choreography diagrams and tasks associated with a conversation node are initialized with the participants linked to the node.

There are various ways to create the link:

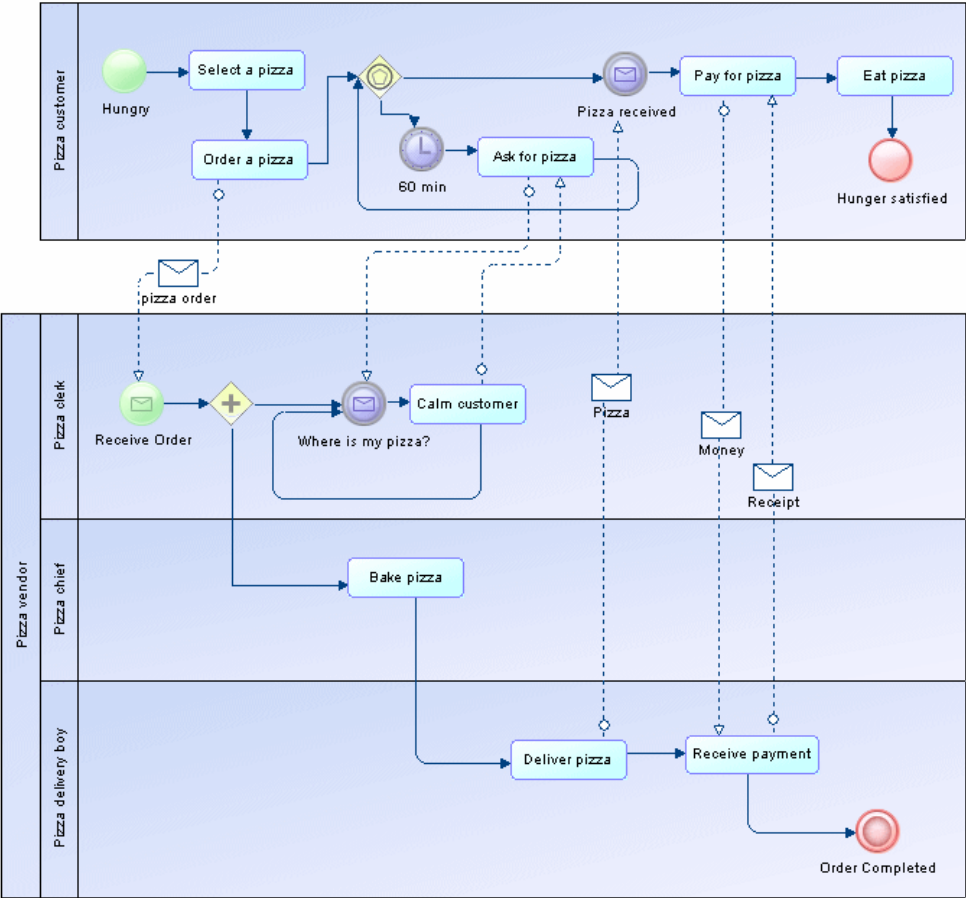
- To associate a choreography diagram with a conversation node from the choreography diagram, right-click the diagram background and select **Diagram > Properties**, and then select the appropriate node in the **Related node** list on the **General** tab of the diagram property sheet. Any choreography tasks you create in the diagram will be initialized with the participants associated with the node.
- To associate a choreography task with a conversation node from the task property sheet, select the appropriate node in the **Related node** list on the **General** tab of the task property sheet. The task participants will be set to the participants associated with the node.
- To associate a conversation node with an existing choreography diagram from the conversation node symbol, right-click the symbol and select **Related Diagram > DiagramName**. Alternately, you can create a new choreography diagram from a conversation node, by selecting **Related Diagram > New**. In both cases, to complete the link, you must open the choreography diagram property sheet and select the node in the **Related node** list. Any choreography tasks you create in the diagram will be initialized with the participants associated with the node.

Collaboration and Process Diagrams (BPMN)

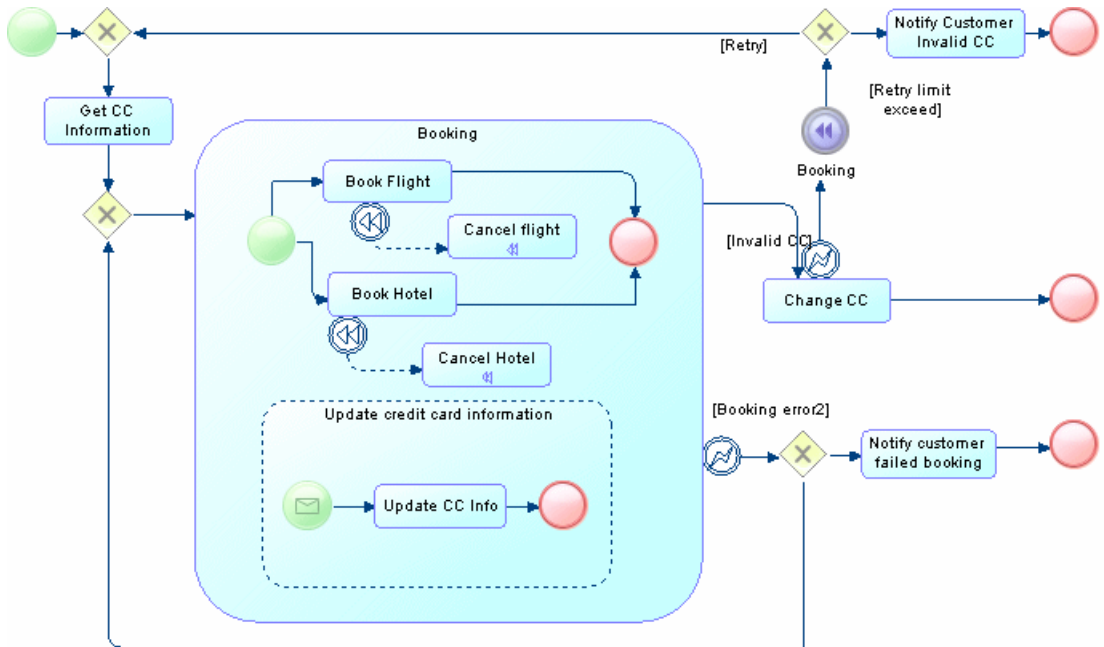
A collaboration diagram analyzes the sequence flow of processes and the exchange of messages between participants (represented as swimlanes and pools). Each pool contains an implicit process with a start event and one or more end events. A process diagram analyzes the sequence flow in a single process in a participant (which can be shown or implicit). PowerDesigner supports collaboration diagrams and process diagrams as standard business process diagrams with a BPMN-specific Toolbox.

In the following example collaboration diagram, the interactions between the staff of a pizza restaurant and a customer are analyzed:









CHAPTER 10: Business Process Modeling Notation (BPMN)





In the following example process diagram, the booking process internal to a travel agency is analyzed:



The following tools are available in collaboration and process diagrams:

Tool	Description
 ...	Start Events - Initiate a process (see <i>Events (BPMN)</i> on page 191). The various types of start events each have their own tools.
 ...	Intermediate Events - Trigger further activity during a process (see <i>Events (BPMN)</i> on page 191). The various types of intermediate events each have their own tools.
 ...	End Events - Conclude a process (see <i>Events (BPMN)</i> on page 191). The various types of end events each have their own tools.
 ...	Activities - Work performed within a process (see <i>Activities (BPMN)</i> on page 195). The various types of activities each have their own tools.
 ...	Participant - Organization, business unit, or role represented as swimlanes and pools (see <i>Participants (BPMN)</i> on page 189).
 ...	Gateways - Merge or split the sequence flow (see <i>Gateways (BPMN)</i> on page 194). The various types of gateways each have their own tools.
 ...	Data Objects - Information item used in a process (see <i>Data (BPMN)</i> on page 196). The various types of data objects each have their own tools.
 ...	Message Flow - Links a participant to another participant and passes a message between them. You can also draw message flows from an activity contained within a participant to another participant or to one of its activities (see <i>Flows and Links (BPMN)</i> on page 197).

Tool	Description
	Sequence Flow - Links two elements (events, activities, gateways) in a process (see <i>Flows and Links (BPMN)</i> on page 197).
	Data Association - Links a data object to an activity or event (see <i>Flows and Links (BPMN)</i> on page 197).

Note: You can change the type of an event, activity, or gateway by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Participants (BPMN)

Participants represent companies, departments, or roles who are involved in a collaboration. Participants are represented as swimlanes in collaboration and process diagrams and as square nodes in conversation diagrams. In choreography diagrams, participants do not have a separate symbol but are displayed on the top or bottom band of the choreography task symbol.

Participants are based on standard BPM organization units (see *Organization Units (BPM)* on page 52), and have the following additional properties:

Property	Description
Multi-instance	The participant symbol represents multiple instances of the specified role.

Conversation Nodes (BPMN)

A conversation node is a hexagon symbol in a conversation diagram that links two participants and regroups a set of message exchanges that share the same correlation.

Conversation nodes can be decomposed and contain their own conversation diagrams in the same way as standard BPM processes (see *Decomposing Processes* on page 44).

Conversation Node Properties

Conversation nodes have the following properties:

Property	Description
Name	Specifies the name of the item, which should be clear and meaningful, and should convey the item's purpose to non-technical users.
Code	Specifies the technical name of the item, which is used for generating code or scripts.
Comment	Specifies a descriptive comment for the object.

Property	Description
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Composite status	Specifies whether the task is a simple conversation or a sub-conversation (which can itself contain conversation nodes, listed on the Conversation Nodes tab). If you revert from a sub-conversation back to a communication, then any conversations that you have created inside it will be deleted. For detailed information about decomposing objects, see <i>Decomposing Processes</i> on page 44.
Reusable	Specifies whether the conversation node may be reused in other contexts.
Reuse conversation	Specifies the conversation node that is being reused in this context.
Correlation key	[atomic conversations only] Specifies the correlation key (set of correlation properties drawn from the message) used to associate the conversation to a particular instance of a process (see <i>Correlation Keys (BPM)</i> on page 121). Each flow connected to the node must have the same key as the node.

Choreography Tasks (BPMN)

A choreography task represents an interaction, a set of message exchanges between two participants. The name of the choreography task and each of the participants are displayed in the various bands of the its symbol

Choreography tasks can be decomposed and contain their own choreography diagrams in the same way as standard BPM processes (see *Decomposing Processes* on page 44).

Choreography Task Properties

Choreography tasks have the following properties:

Property	Description
Name	Specifies the name of the item, which should be clear and meaningful, and should convey the item's purpose to non-technical users.
Code	Specifies the technical name of the item, which is used for generating code or scripts.
Comment	Specifies a descriptive comment for the object.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

Property	Description
Related node	Specifies the conversation node with which the choreography task is associated. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected conversation node.
Composite status	Specifies whether the task is a choreography task or choreography sub-process (which can itself contain choreography tasks, listed on the Choreography Tasks tab). If you revert from a choreography sub-process back to a choreography task, then any tasks that you have created inside it will be deleted. For detailed information about decomposing objects, see <i>Decomposing Processes</i> on page 44.
Reusable	Specifies whether the task may be reused in other contexts.
Reuse task	Specifies the choreography task that is being reused in this context.
Initiating and Responding participants	[atomic tasks only] Specify the participants that interact through the choreography task. The initiating participant and her message are colored white and the responding participant and her message are colored grey. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected participant. Select the Multiple checkbox to specify that there is more than one initiating or responding participant.
Initiating and Return messages	[atomic tasks only] Specify the messages that the participants exchange through the choreography task. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected message.
Loop characteristics	Specifies that the task is a loop or multiple-instance (parallel or sequential) choreography task.

Events (BPMN)

An event is something that happens during the course of a process. Events include the start and end of an activity, and any other intermediate happening (such as a change of state or receipt of a message) which will affect its sequence or timing. You can create events in collaboration, process, and choreography diagrams.

The following types of events are available:

- None - Untyped events, which indicate start points, state changes, and final states.
- Message - Receiving and sending messages.
- Timer - Cyclic timer events, points in time, time spans, or timeouts.
- Escalation - Escalating to a higher level of responsibility.

- Conditional - Reacting to changed business conditions or integrating business rules.
- Link - Off-page connectors. Two corresponding link events equal a sequence flow.
- Error - Catching or throwing named errors.
- Cancel - Reacting to cancelled transactions or triggering cancellation.
- Compensation - Handling or triggering compensation.
- Signal - Signalling across different processes. A signal thrown can be caught multiple times.
- Multiple - Catching one out of a set of events. Throwing all events defined.
- Parallel multiple - Catching all out of a set of parallel events.
- Terminate - Triggering the immediate termination of a process.

Creating Events

Each type of event can be created in one or more different contexts:

- Start events:
 - Top-level - Create the event with its Toolbox tool.
 - Event sub-process interrupting - Create the event with its Toolbox tool within an event sub-process.
 - Event sub-process non-interrupting - Create an interrupting event sub-process, right-click the symbol, and select **Change to Non-Interrupting**.
Alternately, open the event property sheet and deselect the **Interrupting** checkbox.
- Intermediate events:
 - Catching - Create the event with its Toolbox tool.
 - Boundary interrupting - Right-click a sequence flow and select **Add Boundary Event > Type**.
Alternately, open the property sheet of a sequence flow, click the **Condition** tab, click the **Create** tool to the right of the **Event** field, and select the appropriate type of event in the **Stereotype** field of the event property sheet.
 - Boundary non-interrupting - Create a boundary interrupting event, right click the symbol, and select **Change to Non-Interrupting**.
Alternately, open the event property sheet and deselect the **Interrupting** checkbox.
 - Throwing - Create a catching event, right-click the symbol, and select **Change to Throwing Event**.
Alternately, open the event property sheet, click the **Implementation** tab, and select **Generate event** in the **Type** field.
- End events - Create the event with its Toolbox tool.

Event Types

The following table shows the symbols of all the types of events available in each context:

Type	Start			Intermediate			End	
	Top-Level	Event Sub-Process		Catch-ing	Boundary			Throw-ing
		Inter-rupting	Non-In-terrupting		Inter-rupting	Non-In-terrupting		
None								
Message								
Timer								
Escala-tion								
Condi-tional								
Link								
Error								
Cancel								
Compen-sation								
Signal								
Multiple								
Parallel Multiple								
Termi-nate								








Events are based upon and share the properties of starts (see *Starts (BPM)* on page 62), processes, (see *Processes (BPM)* on page 32), or ends (see *Ends (BPM)* on page 63), as appropriate.

Note: You can change the type of an event by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Gateways (BPMN)

Gateways control the sequence flow of the process, and can merge or split the flow as dictated by the gateway conditions. You can create gateways in collaboration, process, and choreography diagrams.

The following kinds of gateways are available:

Symbol	Description
	Basic/Exclusive gateway - When splitting, routes the flow to one outgoing branch. When merging, waits for one incoming branch to complete before triggering the outgoing flow.
	Parallel gateway - When splitting, activates all outgoing branches simultaneously. When merging, waits for all incoming branches to complete.
	Inclusive gateway - When splitting, activates one or more branches. When merging, waits for all incoming branches to complete before merging.
	Event-based gateway - Followed by catching events or receive tasks and routes the flow to whichever of these happens first.
	Exclusive event-based gateway - Starts a new process instance for each occurrence of a subsequent event.
	Parallel event-based gateway - Starts a new process instance for the occurrence of all subsequent events.
	Complex gateway - Treats complex merging or branching behavior not covered by other gateways.














Gateways are based upon, and share the properties of decisions (see *Decisions (BPM)* on page 65).

Note: You can change the type of a gateway by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Activities (BPMN)

Activities are work that is performed within a process. You can create activities in collaboration and process diagrams.

The following types of activities are available:

Symbol	Description
	Abstract task - Basic unit of work.
	Send task - Sends a message to a participant external to the process. Once the message has been sent, the task is completed.
	Receive task - Waits for a message to arrive from a participant external to the process. Once the message has been received, the task is completed.
	User task - A human performer performs the task with the assistance of a software application and is scheduled through a task list manager of some sort.
	Manual task - A task that is performed without the aid of any business process execution engine or any application. For example, a telephone technician installing a telephone at a customer location.
	Business rule task - Sends input to a business rules engine and receives the output of the engine's calculations.
	Service task - Uses a Web service or automated application.
	Script task - Executed by a script interpreted by a business process engine.
	Transaction - Set of activities that logically belong together, and which might follow a specific transaction protocol.
	Call activity - Wrapper for a globally defined sub-process or task that is reused in the current process.
	Sub-process - An activity whose internal details have been modeled using activities, gateways, events, and sequence flows.
	Event sub-process - Placed into a process or sub-process and is activated when its start event is triggered, and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.
	Ad hoc sub-process - A specialized type of sub-process that is a group of activities that have no required sequence relationships, and whose sequence and number are determined by the performers of the activities.

Activities are based upon and share the properties of standard BPM processes (see *Processes (BPM)* on page 32). They can be decomposed and contain their own collaboration diagrams in the same way as processes (see *Decomposing Processes* on page 44).








Note: You can change the type of an activity by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Data (BPMN)

Data are physical or information items that are created, manipulated, or otherwise used during the execution of a process. You can create data objects in collaboration and process diagrams.

Note: PowerDesigner does not support the association of data objects with sequence flows.

The following kinds of data are available:

Symbol	Description
	Data object - Information flowing through the process.
	Data input - External input for the entire process, which can be read by an activity.
	Data output - Variable available as the result of the entire process.
	Collection data object - Collection of information, such as a list of order items.
	Collection data input
	Collection data output
	Data store - Place where the process can read or write data, such as a database or filing cabinet, and which persists beyond the lifetime of the process instance.

Note: You can change the type of a data object by selecting a different stereotype from the **Stereotype** list on the **General** tab of its property sheet, or by right-clicking its symbol and selecting the appropriate **Change to...** command.

Correlation Keys and Correlation Properties (BPMN)

Correlation keys are sets of correlation properties used to associate a message to a particular instance of a process.

BPMN correlation keys and correlation properties are based upon and have the same properties as standard correlation keys (see *Correlation Keys (BPM)* on page 121) and variables (see *Variables (BPM)* on page 119).

Messages (BPMN)

A message represents the content of a communication between two participants, and is passed along a message flow. In choreography diagrams, an initiating message is automatically colored white, and a non-initiating message is automatically colored grey.





Messages are based upon and share the same properties as standard BPM message formats (see *Message Formats (BPM)* on page 78).

To view all the messages exchanged between participants in your model, right-click the model node in the Browser and select **New > Message Flow Matrix**. You can create and delete messages directly in this matrix.

Flows and Links (BPMN)

BPMN provides various kinds of flows and links to connect objects in collaboration, conversation, and choreography diagrams.

The following types of flows and links are available:

Tool	Description
	Sequence Flow - Connects events, activities, and gateways in processes in collaboration, process, and choreography diagrams.
	Message Flow - Connects participants in collaboration diagrams. Connects message symbols to participant bands on choreography tasks in choreography diagrams.
	Data Association - Connects data objects to activities or events in collaboration and process diagrams (see <i>Resource Flows (BPM)</i> on page 95).
	Conversation Link - Connects participants in conversation diagrams.

Sequence and Message Flow Properties

Sequence and Message Flows are based on and have the same properties as standard flows (see *Flow Properties* on page 71). Message flows have the following additional properties:

Property	Description
Correlation key	Specifies the correlation key (set of correlation properties drawn from the message) used to associate the message to a particular instance of a process (see <i>Correlation Keys (BPM)</i> on page 121). Each flow must have the same key as the conversation node to which it is connected.
Correlation property	Specifies the correlation property that acts as the unique identifier for this instance of the message (see <i>Variables (BPM)</i> on page 119).

Note: To set a sequence flow to the default flow, right-click the flow and select **Set to Default Flow**. The source of a default flow must be of an origin of an inclusive, exclusive, or complex gateway or an activity.

Conversation Link Properties

Conversation links have the following properties:

Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
Stereotype	Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
Participant	Specifies the participant to which the link is joined.
Conversation node	Specifies the conversation node to which the link is joined. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected conversation node.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

CHAPTER 11 Data Flow Diagram (DFD)

The *Data Flow Diagram* (DFD) is a graphical representation of the flow of data through an information system. It enables you to represent the processes in your information system from the viewpoint of data. The DFD lets you visualize how the system operates, what the system accomplishes and how it will be implemented, when it is refined with further specification.

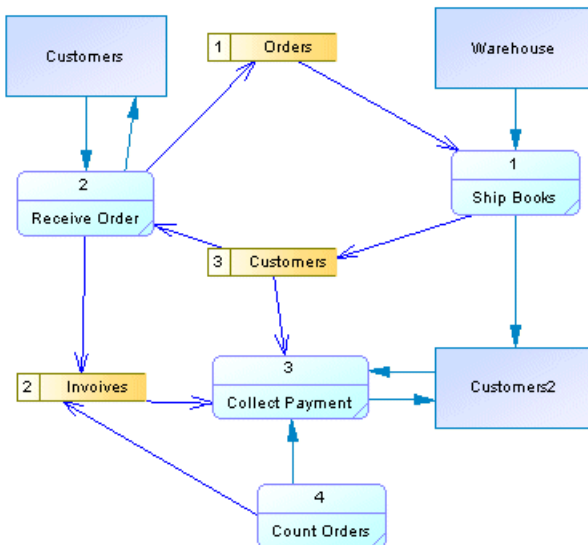
Data flow diagrams are used by systems analysts to design information-processing systems but also as a way to model whole organizations. You build a DFD at the very beginning of your business process modeling in order to model the functions your system has to carry out and the interaction between those functions together with focusing on data exchanges between processes. You can associate data with conceptual, logical, and physical data models and object-oriented models.

There are two types of DFDs, both of which support a top-down approach to systems analysis, whereby analysts begin by developing a general understanding of the system and gradually break components out into greater detail:

- Logical data flow diagrams - are implementation-independent and describe the system, rather than how activities are accomplished.
- Physical data flow diagrams - are implementation-dependent and describe the actual entities (devices, department, people, etc.) involved in the current system.

DFDs can also be grouped together to represent a sub-system of the system being analyzed.


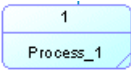
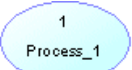





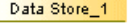






A data flow diagram can look as follows:



PowerDesigner support for DFD includes:

- Support for the Gane & Sarson and Yourdon notations, which you choose between by selecting **Tools > Model Options**.
- Automatic processes and data stores numbering (see *Process and Data Store Numbering* on page 203).
- Data flow diagram balancing (see *Data Flow Diagram Balancing* on page 205).
- Data Flow Diagram specific validation rules (**F4**) - PowerDesigner may perform automatic corrections to your model or output errors and warnings that you will have to correct manually.

In addition to PowerDesigner's standard palette, a Data Flow Diagram Toolbox is available to let you rapidly create objects specific to the diagram type:

Concept	Tool	Gane & Sarson	Yourdon	Description
Process				Location where data is transformed. See <i>Process</i> on page 201.
Flow				Oriented link between objects, which conveys data. See <i>Flow</i> on page 201.
Data store				Repository of data. See <i>Data store</i> on page 202.
External entity				Source or destination of data. See <i>External entity</i> on page 202.
Split/Merge				Splits a flow into several flows or merges flows from different sources into one flow. See <i>Split/merge</i> on page 202.

This chapter outlines the specifics of PowerDesigner's support for data flow diagrams, and should be read in conjunction with *Chapter 3, Building Business Process Diagrams* on page 21.

Creating a Data Flow Diagram

You create a data flow diagram from the business process diagram with the Data Flow Diagram process language attached.

1. Select **File > New Model**, and choose to create a business process model from the Model type list.
2. Select **Business Process Diagram** as the first diagram.
3. Select **Data Flow Diagram** from the Process language list and click **OK**.

Designing for Data Flow Diagram

This section explains the DFD's objects and how to design each object in the PowerDesigner Business Process Model.

You design Data Flow Diagram objects using business process diagram objects with a specific stereotype.

Process

A *process* is an activity, which transforms and manipulates input data to produce output data.

For example, in a model about the publication of books, selecting a manuscript is a process. Data is sent to the selection process in the form of a manuscript. During selection, the manuscript is transformed either into a manuscript that goes directly to the printer, or into a manuscript that must wait before it is printed.

Flows outgoing from processes can go to external entities, data stores, split/merges, or other processes.

You design a DFD process using a BPM process.

The Data tab in the property sheet displays the process CRUD accesses to data.

Synchronization of these data accesses with the data transported by the incoming and outgoing flows of the process is made through the balancing check. For more information see *Data flow diagram balancing* on page 205.

Flow

A *flow* conveys data between processes, external entities, and data stores. It represents data in motion, which can be computerized components, such as messages or bits, or non-computerized components, such as eggs or cake for example.

A flow cannot link two data stores or two external entities without going through a process or a split/merge.

Flows outgoing from data stores are interpreted as a read access to information in the data store.

Flows incoming to data stores are interpreted as a write, update, or delete access to information in the data store.

You design a DFD flow using a BPM flow (or a resource flow object when the destination object is a data store) with a Flow stereotype. Flows incoming to and outgoing from data stores must be created with the Resource Flow tool when using the DFD specific toolbox.

The Data tab in the property sheet displays the data transported by the flows.

Data Store

A *data store* is the location where data resides permanently or temporarily. It responds to requests for storing and accessing data, but cannot initiate any actions. It represents data at rest, which can be computerized components, such as files or databases, or non-computerized components, such as names and addresses in an address book for example.

You design a data store using a resource object with a Data Store stereotype. Flows incoming to and outgoing from data stores must be created with the Resource Flow tool when using the DFD specific toolbox.

External Entity

An *external entity* sends or receives data from the system. It can represent a person, a machine, an organization etc., that is external to the system being modeled. Flows outgoing from external entities go to processes.

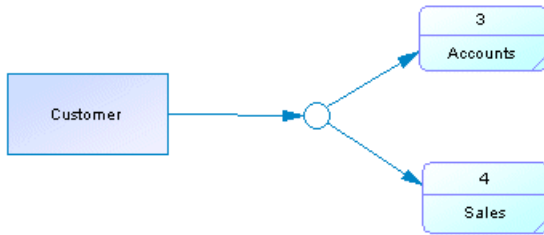
You design an external entity using an organization unit (actor symbol) with an External Entity stereotype. Organization units represented as swimlanes are not available in the Data Flow Diagram process language.

Split/merge

A *split/merge* allows you to either splits a flow into several flows so that it can send data to different destinations, or merges flows from different sources into one flow.

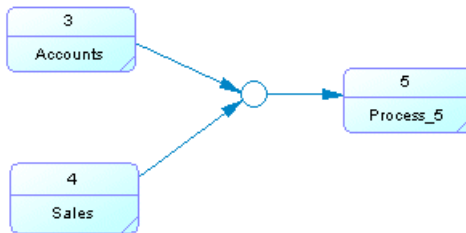
A split/merge that sends data to several destinations has a single incoming flow and multiple outgoing flows. It can mean that a complex packet of data is being split into several more elementary data packets. Each of which is being sent to different parts of the system. It can also mean that duplicate copies of data packets are being sent to different parts of the system, for example.

The following example shows a split/merge with a single incoming flow and multiple outgoing flows:



A split/merge that joins flows from different sources has multiple incoming flows and a single outgoing flow. It means that several data packets are joining together to form more complex packets of data.

The following example shows a split/merge with multiple incoming and a single outgoing flow:



Flows outgoing from split/merges can go to data stores, processes and other split/merges.

You design a split/merge using a synchronization object with a Split/Merge stereotype.

Process and Data Store Numbering

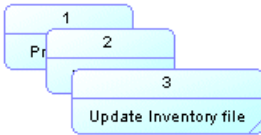
Process and data store numbering is a convenient way of referencing processes and data stores in a DFD. For example, in a lively discussion about processes or data stores in a DFD it is easier to mention a process or a data store by its number instead of its name, which can sometimes be long or complex.

1 is the numbering default value. You can modify this value at any time and type an integer greater than 0 in the process or data store property sheet.

All the objects (processes or data stores) you will create afterwards will be automatically numbered in ascending order. Number ID of the already created objects will not be impacted unless you use the Renumber Process IDs command or the Renumber Data Store IDs command by right-clicking the diagram background.

This feature reorders objects number IDs and the object number ID you have changed become the last in the ascending order.

Numbered processes

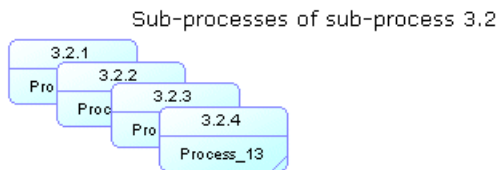
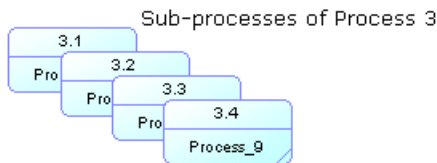
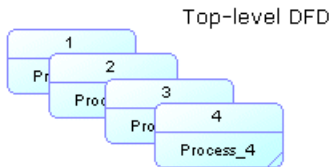


Numbered data stores



In addition, numbers are used in leveled data flow diagrams. You can decompose a process into lower level processes, which more closely analyze the various transformations carried out by the parent process. The numbering feature allows child processes to inherit the number ID of their parent process.

So for example, a top level DFD would have processes 1 2 3 4, the sub-process of process 3 would have processes 3.1, 3.2, 3.3, and 3.4, and the sub-process of the sub-process 3.2 would have components 3.2.1, 3.2.2, 3.2.3, and 3.2.4:



For more information about decomposed processes, see *Decomposing Processes* on page 44.

Data store numbering is only available with Gane & Sarson methodology.

Data Flow Diagram Balancing

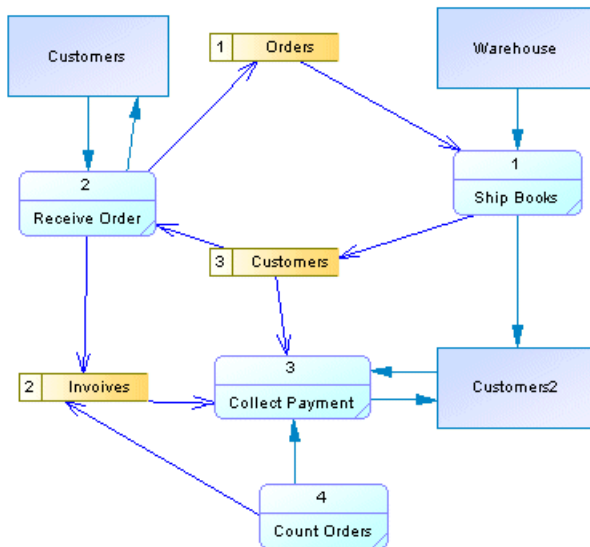
The concept of balancing states that all the incoming flows to a process and all the outgoing flows from a process in the parent diagram should be preserved at the next level of decomposition.

Process decomposition lets you organize your overall DFD in a series of levels so that each level provides successively more detail about a portion of the level above it.

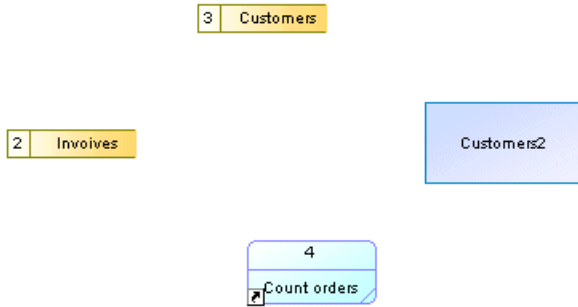
The goal of the balancing feature is to check your system internal consistency, which is particularly useful as different levels of expertise are generally involved in a project.

When you decompose a process, PowerDesigner helps you initialize, in the sub diagram, the objects from the upper-level to link to the sub-process. PowerDesigner automatically retrieves global objects, such as external entities or data stores and creates object shortcuts, if need be.

The following example shows a top level DFD, in which we are going to decompose the Collect Payment process:



The following example shows the default sub-process diagram of the Collect Payment decomposed process containing its related objects coming from the upper-level. Then, you have to link the objects. Each object you link afterwards to the parent process in the upper-level is automatically displayed in the sub-process diagram:



In addition, PowerDesigner ensures, using balancing checks, that each data on flows or in data stores in the upper-level exist in the sub-diagram.

Service Oriented Architecture (SOA)

SOA is a logical orchestration process language that allows you to orchestrate your processes without being linked to any platform or language. It belongs to the Service Orchestration family.

The SOA process language is very close to BPEL4WS, except that:

- It has no generation and reverse functions
- Any type of operation can be attached to a process (BPEL4WS only supports One-Way and Request-Response operations on processes)
- No correlation key can be defined for sent messages

An SOA BPM is a model that allows you to assemble software components that are designed by a WSDL. Therefore, you can import WSDL files in an SOA BPM.

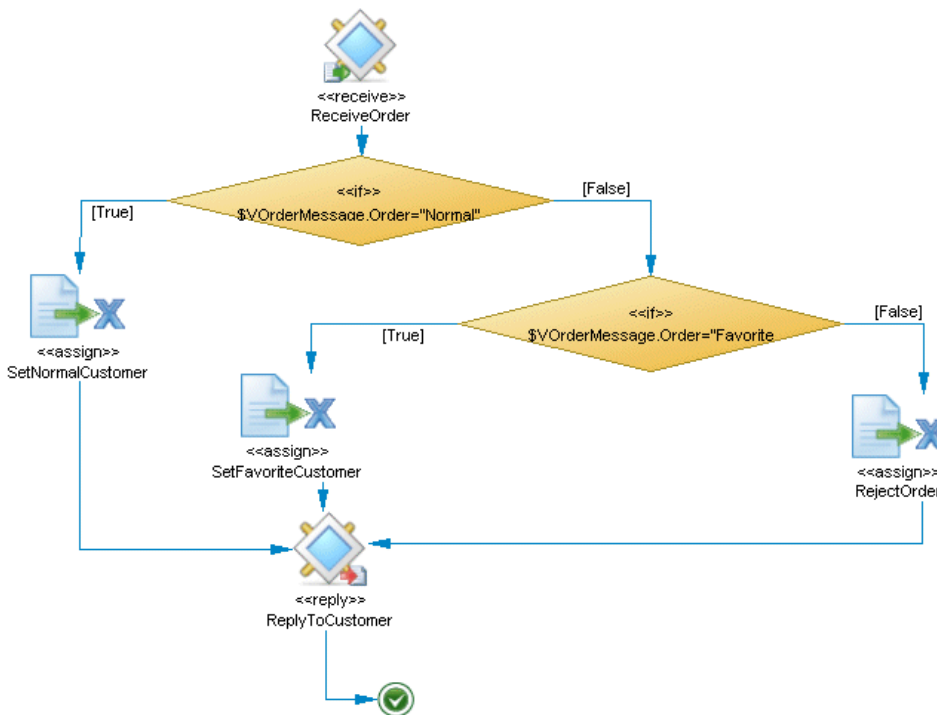
When you decide to use the Service Oriented Architecture process language, you generally do not know yet which platform you are going to use to execute your processes. However, SOA allows you to design the Web services orchestration by providing access to service providers, service interfaces and operations.

You can afterwards use the **Change Target** command (see *Changing the Process Language* on page 12) or Model-to-Model Generation (*Chapter 8, Generating Other Models from a BPM* on page 173) to change to the appropriate platform or language.

CHAPTER 13 BPEL4WS and WS-BPEL

BPEL4WS 1.1 (Business Process Execution Language for Web Services) and its successor WS-BPEL 2.0 (Web Services for Business Process Execution Language) are business process orchestration standards which let you describe your business processes under the form of Web services, and specify how they are connected to each other to accomplish specific tasks. PowerDesigner supports modeling for BPEL4WS 1.1 and WS-BPEL 2.0 and higher, including round-trip engineering.

The following example shows a business process model diagram modeled with WS-BPEL 2.0 in which the control flow takes a different path whether the condition of the first decision is true or false, then on the false path, the control flow takes a different path again whether the condition of the second decision is true or false. All paths go to the same end:



Modeling for BPEL Languages

PowerDesigner supports the modeling of all the components required to deploy a BPEL solution in your environment.

1. Create a BPM with the process language set to BPEL4WS 1.1 or WS-BPEL 2.0.

Note: You can generate a BPEL BPM from an analysis BPM (see *Chapter 8, Generating Other Models from a BPM* on page 173) or reverse engineer BPEL files into a BPM (see *Reverse Engineering Source Files into a BPM* on page 171).

A valid BPEL model must consist of a top-level diagram with one or more top-level processes.

2. For each of your top-level processes, specify its partners and their interactions using organisation units (see *Organization Units (BPM)* on page 52) and role associations (see *Role Associations (BPM)* on page 73) respectively.
3. Import a WSDL file you own or one you have found published in a UDDI server (see *Importing a Service Provider from a WSDL File* on page 103) to retrieve service description objects (service providers, service interfaces, and operations). You can also create these objects from scratch (see *Service Providers (BPM)* on page 98, *Service Interfaces (BPM)* on page 110, and *Operations (BPM)* on page 111).
4. Drill down in the choreography diagram into which each of your top-level processes is decomposed.
5. For each process within each top-level process, assign a partner using an organisation unit (see *Attaching Processes to Organization Units* on page 55), and specify its implementation (see *Process Properties* on page 34).
6. Complete your process choreography by creating any appropriate additional processes (for example to catch a fault or compensate an error), and specify how you want to manage data in the exchanged messages using variables, data transformations and correlation keys.
7. [optional] Decompose one or more processes you want to analyze in more detail (see *Decomposing Processes* on page 44).
8. [optional] Generate BPEL code from your BPM objects to be interpreted by orchestration engine (see *Generating for BPEL Languages* on page 223).

Building BPEL Top-Level Diagrams

The PowerDesigner BPM provides support for the following elements in a top-level diagram, when modeling a BPEL environment:

- *Empty activities* - used to specify *top-level Processes* which are global services, that interact with partners (see *Stereotype Activities in Building BPEL Choreography Diagrams* on page 211).

BPEL4WS 1.1 models top-level processes (see *Processes (BPM)* on page 32) as standard processes in top-level diagrams with additional properties (see *BPEL Top-Level Process Properties* on page 217).

- *Partners* - represent consumers of a service provided by the business process (initiating role) or providers of a service to the business process (responding role), and are connected to a top-level process by partner link types.

PowerDesigner models partners as organization units represented as actors (see *Organization Units (BPM)* on page 52).

- *Partner Link Types* - represent interactions between top-level processes and business partners.

PowerDesigner models partner link types as role associations (see *Role Associations (BPM)* on page 73) with additional properties (see *WS-BPEL 2.0 Object Properties* on page 218 and *BPEL4WS 1.1 Object Properties* on page 221).

Building BPEL Choreography Diagrams

The PowerDesigner BPM provides support for the following elements in a choreography diagram, when modeling a BPEL environment.

Note: You create most BPEL activities by modifying the properties (see *Specifying Implementation Types* on page 37) of processes (see *Processes (BPM)* on page 32), and specifying additional BPEL properties (see *WS-BPEL 2.0 Object Properties* on page 218 and *BPEL4WS 1.1 Object Properties* on page 221). WS-BPEL 2.0 provides a Toolbox to guide you through the activities creation.

- *WSDL files* - describe services provided by business partners and the way to access them. WSDL files contain port types and operations.

PowerDesigner models WSDL files as standard service providers (see *Service Providers (BPM)* on page 98), port types as standard service interfaces (see *Service Interfaces (BPM)* on page 110), and operations as standard operations (see *Operations (BPM)* on page 111).

- *Start and ends in scopes* - starts and terminate respectively the flow represented in the diagram.







PowerDesigner models BPEL starts and ends as standard starts (see *Starts (BPM)* on page 62) and ends (see *Ends (BPM)* on page 63). If you need to immediately stop a business process instance use the exit activity.

- *Partners in scopes* - specify the people, groups or organizations which are responsible for a process.

PowerDesigner models partners as standard organization units represented as swimlanes (see *Organization Units (BPM)* on page 52).







Composite Activities

Composite activities specify activities modeled as composite processes in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>Scope activities</i> - provide the context which influences the execution behavior of their nested activities, and allow the definition of variables, partner, message exchanges, correlation sets, event handlers, fault handlers, a compensation handler, and a termination handler.</p> <p>BPEL4WS 1.1 models scopes as standard composite processes.</p>
		<p><i>Sequence activities</i> - specify collections of activities that must be executed sequentially in lexical order. Use this tool if you need a "box" to gather your sequence activities.</p> <p>BPEL4WS 1.1 can gather your sequence activities in a "box" modeled as a standard composite process with <<sequence>> stereotype.</p>
		<p><i>Flow activities</i> - specify a set of concurrent activities that must be executed concurrently. A link modeled as a flow with a link stereotype can express synchronization dependencies between activities. Use this tool if you need a "box" to gather your flow activities.</p> <p>BPEL4WS 1.1 models flow activities as standard synchronizations with <<split>> or <<join>> stereotype. If you need to gather your flow activities in a "box", use a composite process with <<flow>> stereotype and create a nested flow activity.</p>











Loop Activities

Loop activities specify activities modeled as composite processes with Loop implementation type in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>While activities</i> - specify that their nested activities must be repeated until their specified <condition> becomes true.</p> <p>BPEL4WS 1.1 models while activities as standard composite processes with Loop implementation type and While loop type.</p>
		<p><i>ForEach activities</i> [WS-BPEL 2.0] - execute their scope for a specified count. The execution iterations can occur in parallel or in sequence.</p>
		<p><i>RepeatUntil activities</i> [WS-BPEL 2.0] - execute their contained activity at least once until their specified <condition> becomes true.</p>









Generate Event Activities

Generate event activities specify activities modeled as processes with Generate event implementation type in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>Wait activities</i> - specify a delay for a certain period of time or until a certain deadline is reached.</p> <p>BPEL4WS 1.1 models wait activities as standard processes with Generate event implementation type and Timer type event .</p>
		<p><i>Compensate activities</i> - cause all immediately enclosed scopes to be compensated in default order.</p> <p>BPEL4WS 1.1 models compensate activities as standard processes with Generate event implementation type and Compensation type event.</p>
		<p><i>Compensate scope activities</i> [WS-BPEL 2.0] - cause one specified child scope to be compensated.</p> <hr/> <p>Note: To compensate a scope, create a flow from the scope to the compensate activity, and select the generated compensation event in the Events tab of the flow property sheet to use it. When the compensation event is not used by any flow, the scope generating the event is a compensate activity.</p> <hr/>
		<p><i>Throw activities</i> - specify a business process which needs to signal an internal fault explicitly.</p> <p>BPEL4WS 1.1 models throw activities as standard processes with Generate event implementation type and Fault type event.</p>
		<p><i>Rethrow activities</i> [WS-BPEL 2.0] - rethrow the fault that was originally caught by the immediately enclosing <catch> and <catchAll> elements within a <faultHandlers> element.</p>








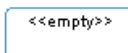
Execute Operation Activities

Execute operation activities specify activities modeled as processes with Execute operation implementation type in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>Receive activities</i> - specify how the business process waits for a matching message to arrive.</p> <p>BPEL4WS 1.1 models receive activities as standard processes with Execute operation implementation type and Receive request action type.</p>
		<p><i>Reply activities</i> - send responses to requests previously accepted through receive activities.</p> <p>BPEL4WS 1.1 models reply activities as standard processes with Execute operation implementation type and Reply action type.</p>
		<p><i>Reply fault activities</i> - send faulted responses to requests previously accepted through receive activities.</p> <p>BPEL4WS 1.1 models reply fault activities as standard processes with Execute operation implementation type and Reply fault action type.</p>
		<p><i>Invoke activities</i> - call Web Services offered by service providers.</p> <p>BPEL4WS 1.1 models invoke activities as standard processes with Execute operation implementation type and Invoke operation action type.</p> <hr/> <p>Note: WS-BPEL receive, reply (fault), and invoke activities let you directly specify assignments on their Assignments tab to copy the value of exchanged message variables into another or to calculate the value of an expression, and store it in a variable (see <i>Process Properties</i> on page 34).</p> <hr/>



Stereotype Activities





Stereotype activities specify activities modeled as processes with stereotype with no implementation type in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>Assign activities</i> - update the values of variables with new data. The assign activity should be composed of at least one atomic assign task (see <i>Process Properties</i> on page 34).</p> <p>BPEL4WS 1.1 models assign activities as standard processes with <code><<assign>></code> stereotype.</p>
		<p><i>Validate activities</i> [WS-BPEL 2.0] - validate the values of variables against their associated XML and WSDL data definition. Specify variables (see <i>Variables (BPM)</i> on page 119) in the ValidatedVariables tab.</p>
		<p><i>Extension activities</i> [WS-BPEL 2.0] - create activities, which are not defined by the specification.</p>
		<p><i>Empty activities</i> - specify activities that do nothing (for example a fault which needs to be caught and suppressed) or can provide a synchronization point in a flow activity. It is also used to model top-level processes in top-level diagrams (see <i>Building BPEL Top-Level Diagrams</i> on page 210)</p> <p>BPEL4WS 1.1 models empty activities as standard processes with no implementation type nor stereotype.</p>

Other Activities

Other activities specify activities modeled as other objects than processes in WS-BPEL 2.0:

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>If activities</i> - select exactly one activity to execute from a set of activities.</p> <p>You can also use composite processes with <code><<if>></code> stereotype for modeling purposes.</p> <p>BPEL4WS 1.1 models switch activities as standard decisions (see <i>Decisions (BPM)</i> on page 65) with <code><<switch>></code> stereotype.</p>

WS-BPEL 2.0 Tool	WS-BPEL 2.0 Symbol	Description
		<p><i>Pick activities</i> - wait for the occurrence of exactly one event from a set of events, then execute the activity associated with that event.</p> <p>BPEL4WS 1.1 models pick activities as standard decisions (see <i>Decisions (BPM)</i> on page 65) with <<pick>> stereotype.</p>
		<p><i>Exit activities</i> - immediately end the business process instance.</p> <p>BPEL4WS 1.1 models terminate activities as standard ends (see <i>Ends (BPM)</i> on page 63) with <<terminate>> stereotype.</p>

Building BPEL Messages

The PowerDesigner BPM provides support for the following elements to build messages in a choreography diagram, when modeling a BPEL environment.

The messages exchanged between activities are handled in the Implementation tab of the processes property sheet:

- *Messages* - identify exchanged data between activities. PowerDesigner models messages as standard message formats (see *Message Formats (BPM)* on page 78) with additional properties (see *WS-BPEL 2.0 Object Properties* on page 218 and *BPEL4WS 1.1 Object Properties* on page 221)
- *Parameters* - identify subdivisions of messages. PowerDesigner models messages as standard message parts (see *Message Parts (BPM)* on page 81) .
- *Variables* - provide the means for holding messages that constitute a part of the state of a business process. PowerDesigner models variables as standard variables (see *Variables (BPM)* on page 119).
- *Properties* - refer to any parts of a variable. PowerDesigner models properties as standard variables (see *Variables (BPM)* on page 119).
- *Property aliases* - provide the means to map a property to a field in a specific message part or variable value. PowerDesigner models property aliases as standard data transformations (see *Data Transformations* on page 123).
- *Correlations*- specify groups of properties that, taken together, serve to identify a message. PowerDesigner models correlations as standard correlation keys (see *Correlation Keys (BPM)* on page 121).
- *XSD schema files* - specify the data schemas handled by Web services, and act as definitions of grammars which take precedence when a disagreement occurs. PowerDesigner models XSD schema files as standard XSD documents (see *XSD Documents* on page 117). You can create an XSD document from the service provider property sheet or import or reverse engineer a WSDL to obtain one.

BPEL Top-Level Process Properties

BPEL top-level process property sheets contain all the standard process tabs, along with the BPEL tab, the properties of which are listed below:

Name	Description
Abstract process	Specifies whether the process being defined is abstract. Default value: No Scripting name: abstractProcess
Definition namespace prefix	Specifies the prefix of the namespace that defines the BPEL definition file. The DefinitionTargetNamespace extended attribute on the Model defines the BPEL definition namespace. Default value: %bpDefPrefix% Scripting name: definitionNamespace
[BPEL4WS] Enable instance compensation	Specifies whether the process instance as a whole can be compensated by platform-specific means. Default value: No Scripting name: enableInstanceCompensation
[WS-BPEL] Exit on standard fault	When set to "yes" on a scope, the process must exit immediately. Default value: No Scripting name: ExitOnStandardFault
Expression language	Specifies the expression language used in the process. Default value: [BPEL4WS] http://www.w3.org/TR/1999/REC-xpath-19991116 and [WS-BPEL] urn:oasis:names:tc:wsbpel:2.0:sub-lang:xpath1.0 Scripting name: expressionLanguage
Query language	Specifies the XML query language used for selection of nodes in assignment, property definition, and other uses. Default value: http://www.w3.org/TR/1999/REC-xpath-19991116 Scripting name: queryLanguage
Target namespace	Specifies the target namespace of the process which is necessary in the generated file. Default value: %urnName% Scripting name: targetNamespace

Name	Description
[BPEL4WS] Variable access serializable [WS-BPEL] Isolated	When set to "yes", the scope provides concurrency control in governing access to shared variables. Such a scope is called a serializable scope. Serializable scopes must not be nested. A scope marked with variableAccessSerializable (or isolated)="yes" must be a leaf scope. Default value: No Scripting name: [BPEL4WS] variableAccessSerializable, [WS-BPEL] Isolated

WS-BPEL 2.0 Object Properties

WS-BPEL 2.0 object property sheets contain additional properties on the WS-BPEL tab.

Name	Description
Counter name	[forEach] Specifies the variable used by the <<forEach>> activity to store the counter of the loop. During each repetition, the "xsd:unsignedInt" variable is implicitly declared in the activity's child <<scope>>. The name of the implicit variable is specified in the CounterName attribute. Scripting name: CounterName
Create instance	[receive and pick] Specifies the instantiation of the process. Default value: No Scripting name: CreateInstance
Data schema target namespace	[WSDL file] Specifies the target namespace of the data schema. Scripting name: schemaNameSpace
Definition namespace	[message format] Specifies the namespace URI message that can only be used by BPEL variables. Default variable: %ownerServiceNmspc% Scripting name: DefinitionNamespace
Definition target namespace	[model and package] Specifies the target namespace. Default value: %urnName% Scripting name: DefinitionTargetNamespace
Exit on standard fault	[scope, sequence, flow and if] When set to "yes" on a scope, the process must exit immediately. Default value: No Scripting name: ExitOnStandardFault

Name	Description
Expression language	<p>Specifies the expression language used in expressions.</p> <p>Default value: urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0</p> <p>Scripting name: expressionLanguage</p>
Final counter expression	<p>[forEach] Computes the final value of the counter variable used by the <<forEach>> activity. It is evaluated when the activity starts.</p> <p>Scripting name: FinalCounterExpression</p>
First correlation pattern	<p>[invoke] When the first correlation is used by the invoke activity, you can choose between one of the following values:</p> <ul style="list-style-type: none"> • request • request-response • response <p>Scripting name: InCorrelationPattern</p>
Imported WSDL content	<p>[WSDL file] Specifies the content of the original reversed WSDL file.</p> <p>Scripting name: ImportedWsdContent</p>
Isolated	<p>[scope, sequence, flow and if] When set to "yes", the scope provides concurrency control in governing access to shared variables. Such a scope is called a serializable scope. Serializable scopes must not be nested. A scope marked with variableAccessSerializable (or isolated)="yes" must be a leaf scope.</p> <p>Default value: No</p> <p>Scripting name: Isolated</p>
Initiate correlation	<p>[receive, reply and reply fault] Specifies the initiation of the correlation used by the receive activity. You can choose between one of the following values to specify the initiate attribute:</p> <ul style="list-style-type: none"> • join, no, yes <p>Scripting name: InitiateCorrelation</p>
Initiate input correlation	<p>[invoke] When the first correlation is used by the invoke activity, you can choose between one of the following values for the initiate attribute:</p> <ul style="list-style-type: none"> • yes • no • join <p>Scripting name: InCorrelationInitiate</p>

Name	Description
Initiate output correlation	<p>[invoke] When the second correlation is used by the invoke activity, you can choose between one of the following values for the initiate attribute:</p> <ul style="list-style-type: none"> • join • no • yes <p>Scripting name: OutCorrelationInitiate</p>
Join condition	<p>Each activity has optional standard attributes: a name, a join condition, and an indicator whether a join fault should be suppressed if it occurs. A join condition is used to specify requirements about concurrent paths reaching at an activity. The default value of the join condition (for the default expression language XPath) is the logical OR of the link status of all incoming links of this activity.</p> <p>Scripting name: joinCondition</p>
Multiple correlation	<p>Specifies a BPEL Invoke, Receive or Reply using multiple correlation keys.</p> <p>Scripting name: MultipleCorrelation</p>
Name	<p>[partner link type] Specifies the name of the partner link type.</p> <p>Scripting name: PartnerLinkTypeName</p>
Namespace prefix to use	<p>[XSD document and WSDL file] Specifies the XML prefix used by the process to reference the schema definition or service provider.</p> <p>Default variable: %wsdlUsedPref%</p> <p>Scripting name: PrefixForUse</p>
Organization unit role	<p>[partner link type] Specifies the role played by a partner in the collaboration. When undefined, the generated role is the name of the organization unit.</p> <p>Scripting name: OrganizationUnitRole</p>
Process role	<p>[partner link type] Specifies the role played by the process in the collaboration. When undefined, the generated role is the name of the process.</p> <p>Scripting name: ProcessRole</p>
Parallel	<p>[forEach] Specifies whether the activity is serial or parallel.</p> <p>Default value: No</p> <p>Scripting name: Parallel</p>

Name	Description
Second correlation pattern	<p>[invoke] When the second correlation is used by the invoke activity, you can choose between one of the following values:</p> <ul style="list-style-type: none"> • request • request-response • response <p>Scripting name: OutCorrelationPattern</p>
Start counter expression	<p>[forEach] Computes the initial value of the counter variable used by the <<forEach>> activity. It is evaluated when the activity starts.</p> <p>Scripting name: StartCounterExpression</p>
Successful branches only	<p>[forEach] Specifies whether to count all scopes or only those which have completed successfully. The <branches> element of the forEach activity represents an unsigned-integer expression used to specify a completion condition.</p> <p>Default value: No</p> <p>Scripting name: SuccessfulBranchesOnly</p>
Suppress join failure	<p>Specifies whether the joinFailure fault will be suppressed for all activities in the process. The effect of the attribute at the process level can be overridden by an activity using a different value for the attribute.</p> <p>Default value: No</p> <p>Scripting name: suppressJoinFailure</p>
Validate	<p>[Assign] Specifies whether the assign activity validates all the variables being modified by the activity.</p> <p>Default value: No</p> <p>Scripting name: Validate</p>

BPEL4WS 1.1 Object Properties

BPEL4WS 1.1 object property sheets contain additional properties on the BPEL4WS tab.

Name	Description
Create instance	<p>[receive] Specifies the instantiation of the process.</p> <p>Default value: No</p> <p>Scripting name: CreateInstance</p>
Data schema target namespace	<p>[WSDL file] Specifies the target namespace of the data schema.</p> <p>Scripting name: schemaNameSpace</p>

Name	Description
Definition namespace	<p>[message format] Specifies the namespace URI message that can only be used by BPEL variables.</p> <p>Default variable: %ownerServiceNmosp%</p> <p>Scripting name: DefinitionNamespace</p>
Definition target namespace	<p>[model and package] Specifies the target namespace.</p> <p>Default value: %urnName%</p> <p>Scripting name: DefinitionTargetNamespace</p>
First correlation pattern	<p>[invoke] When the first correlation is used by the invoke activity, you can choose between one of the following values:</p> <ul style="list-style-type: none"> • in • in-out • out <p>Scripting name: InCorrelationPattern</p>
Imported WSDL content	<p>[WSDL file] Specifies the content of the original reversed WSDL file.</p> <p>Scripting name: ImportedWsdIContent</p>
Initiate correlation	<p>[receive, reply and reply fault] Specifies the initiation of the correlation used by the receive activity. You can choose between one of the following values to specify the initiate attribute:</p> <ul style="list-style-type: none"> • true, false <p>Scripting name: InitiateCorrelation</p>
Join condition	<p>Each activity has optional standard attributes: a name, a join condition, and an indicator whether a join fault should be suppressed if it occurs. A join condition is used to specify requirements about concurrent paths reaching at an activity. The default value of the join condition (for the default expression language XPath) is the logical OR of the link status of all incoming links of this activity.</p> <p>Scripting name: joinCondition</p>
Multiple correlation	<p>Specifies a BPEL Invoke, Receive or Reply using multiple correlation keys.</p> <p>Scripting name: MultipleCorrelation</p>
Name	<p>[partner link type] Specifies the name of the partner link type.</p> <p>Scripting name: PartnerLinkTypeName</p>

Name	Description
Namespace prefix to use	<p>[WSDL file] Specifies the XML prefix used by the process to reference the service provider.</p> <p>Default value: %wsdlUsedPref%</p> <p>Scripting name: prefixForUse</p>
Organization unit role	<p>[partner link type] Specifies the role played by a partner in the collaboration. When undefined, the generated role is the name of the organization unit.</p> <p>Scripting name: OrganizationUnitRole</p>
Process role	<p>[partner link type] Specifies the role played by the process in the collaboration. When undefined, the generated role is the name of the process.</p> <p>Scripting name: ProcessRole</p>
Second correlation pattern	<p>[invoke] When the second correlation is used by the invoke activity, you can choose between one of the following values:</p> <ul style="list-style-type: none"> • in • out-in • out <p>Scripting name: OutCorrelationPattern</p>
Suppress join failure	<p>Specifies whether the joinFailure fault will be suppressed for all activities in the process. The effect of the attribute at the process level can be overridden by an activity using a different value for the attribute.</p> <p>Default value: No</p> <p>Scripting name: suppressJoinFailure</p>
Variable access serializable	<p>[scope, sequence and flow] When set to "yes", the scope provides concurrency control in governing access to shared variables. Such a scope is called a serializable scope. Serializable scopes must not be nested. A scope marked with variableAccessSerializable (or Isolated)="yes" must be a leaf scope.</p> <p>Default value: No</p> <p>Scripting name: variableAccessSerializable</p>

Generating for BPEL Languages

You can generate BPEL code from BPM objects that can be interpreted by any orchestration engine. A separate .BPEL file per each top-level process is generated, and contains the process

descriptions. A .WSDL file (process language definition file) for the entire model is also generated.

1. Select **Language > Generate BPEL4WS 1.1 [or WS-BPEL 2.0] code** to open the Generation dialog.
2. Specify a directory in which to generate the code.
3. [optional] Select the **Check Model** check box, if you want to verify the validity of your model before generation.
4. On the **Selection** tab, select the objects that you want to generate. Use the sub-tabs to navigate between separate lists of object types. The selections you make here will affect the files that are available to select on the **Generated Files** tab.
5. [optional] On the **Options** tab, set the generation option as appropriate:

Option	Description
Generate WSDL files	You can choose between one of the following values: <ul style="list-style-type: none">• Local - Enforces the generation of the .WSDL file into a separate local file which is referenced into the .BPEL definition file via the [Import] clause.• Import - Generates an [Import] clause into the .BPEL definition file.• Embedded - Generates a .WSDL file into the .BPEL definition file, and a .WSDL file is generated for each service provider.

6. Click **OK** to generate the files in the specified directory.

When the generation is complete, the Generated Files dialog opens listing the files, each of which you can open, and review by selecting it, and clicking **Edit**.

Reverse Engineering BPEL Languages

You can reverse engineer files that contain BPEL objects into a BPM.

You reverse engineer the following types of BPEL files into a BPM:

- .BPEL files
- .WSDL files
- .XML files containing a BPEL definition

The WSDL definitions contained in the .BPEL files are reversed into service providers.

Note: We recommend that you begin with importing your .WSDL files before reverse engineering the .BPEL files, as PowerDesigner does not support the [import] clause, which allows you to reverse the WSDL definitions contained in .BPEL files.

1. Select **Language > Reverse Engineer BPEL4WS [or WS-BPEL] File** to display the Reverse dialog.

2. Select to reverse engineer files or directories from the Reverse Engineer list.
3. On the **Selection** tab, click the **Add** button to open a standard **Open** dialog.
4. Select the files or directory you want to reverse, and click Open to display the selected files in the **Reverse** dialog.

You can multi-select files to reverse engineer using the **Ctrl** or **Shift** keys. All files will be reversed in the same BPM.

5. [optional] On the **Options** tab, select the **Create XML Model** check box, whether you want to automatically create an XML model for each schema of the WSDL file.
6. Click **OK** to close the **Reverse** dialog.

The reverse engineering begins, and the Merge Models dialog opens to let you control the differences between your BPM and the reversed engineered files.

For more information about merging models, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

7. Click **OK** to close the dialog.

The objects are added to your model.

Note: You can also reverse engineer BPEL files in a new BPM. For more information, see *Reverse Engineering Source Files into a BPM* on page 171.

You define a BPM with the Sybase WorkSpace Business Process language when you want to design business processes and implement them using Business Process Service in Sybase WorkSpace.

The key concepts of Sybase WorkSpace Business Process are the following:

Key concepts	In PowerDesigner
Service	Interface to an external application or business process. Sybase WorkSpace supports many different kinds of services like SOAP, database, EJB, Java, and transformation service.
Business process	Business process model.
Service invocation	Activity that invokes a service operation.
One-way operation	This kind of operation is invoked when a business process sends a message to a service.
Request/Reply operation	This kind of operation is invoked when a business process sends a message to a service and waits for a response from the service.
Receive activity	Activity that receives a request from an external user or application.
Send activity	Activity that sends back a response to a request.
Assign activity	Each assign activity is a sequence of atomic assign tasks. An assign task is an XPath expression that copies value from a variable to another variable.
Split-Join activity	A split is a point in the business process where a single activity splits into two or more parallel activities. A join is a point in the business process where two or more parallel activities converge into a single common activity.
Complex activity	Activity made of atomic activities that imply drill-down capabilities.
Loop activity	Iteration in a cycle involving the repetitive execution of one or more activities until a condition is met. The loop is a sort of complex activity.
Exception handling activity	Fault, timeout or compensation handling.
Delay activity	Defines a pause on the runtime processing and the delay for continuing the runtime processing.

Key concepts	In PowerDesigner
Terminal activity	Defines different kinds of end activities in order to end the runtime processing of a business process or a complex activity. The end object only terminates a branch of the process.
Sequence flow	Control flow between activities.

Note: To check your Sybase WorkSpace BPM, select **Tools > Check Model**. The model check may perform automatic corrections to your model or output errors and warnings to manually correct. Custom checks appear under the different metaclasses in the Profile category of the Sybase WorkSpace Business process language. Right-click these checks and select **Help** to obtain information about the check.

Top-Down Design

A business analyst designs a model using the Analysis process language because he wants to have a global perspective on the processes of an enterprise, and a developer recovers this model to automate some of its processes.

1. Generate a BPM for Sybase WorkSpace Business Process - Select **Tools > Generate Business Process Model**. The original Analysis model is preserved for future regeneration. If the Analysis model contains several top-level processes, you need to generate one model for each top-level process by deselecting the other processes in the **Selection** tab of the Generation dialog. During generation, some semantic transformations are performed on the model to make it compliant with Sybase WorkSpace Business Process.
2. Identify <<Send>> and <<Receive>> Activities - Change the <<Undefined>> processes corresponding to the reception of a service request to receive activities and those corresponding to answers to these requests, to send activities and select their <<Receive>> process.
3. Implement Other Services by Web service operations - If the operations are already implemented in the WorkSpace environment, you can import them as service provider operations and associate the service operations to <<Undefined>> processes (see *Importing WorkSpace Services* on page 247). If the operations are not yet implemented, right-click the <<Undefined>> process and select **Invoke New ... Service** to create an empty operation that can be further defined using WorkSpace service editors. The supported WorkSpace service types are: Java, Transformation, Database and Message (see *Invoking WorkSpace Services* on page 248).
4. Decision Stereotype - A <<SingleRule>> decision is taken during the evaluation of a boolean expression defined in the WorkSpace decision editor. Only two flows must come from the <<SingleRule>> decision and one of these flows must have a **False** condition. A <<Choice>> decision is triggered by a receive activity. All the flows

coming from the <<Choice>> decision must target receive activities, and there can be more than two flows.

5. Check the Model - To verify that it conforms to the Sybase WorkSpace Business Process language standard.
6. Generate Sybase WorkSpace Business Process files - Select **Language > Generate Sybase WorkSpace Business Process 1.0 code**. See *Generating for Sybase WorkSpace Business Process* on page 248.
7. Use the Business Process Service Editor to continue process implementation. You will have to define the type of received and sent messages, define <<SingleRule>> condition expressions, and so on.

Importing Existing Services

A developer recovers a model (either from an Analysis model or from BPEL files) where he wants to automate processes. To do so, he needs services and he knows existing services are available in the WorkSpace environment.

1. Import Services - Select **Tools > Import WorkSpace Services**. Services are imported as service providers and service operations in the current BPM. The import process retrieves all the services of the selected project and the new service providers appear in the Model Explorer when import is over. See *Importing WorkSpace Services* on page 247.
2. Invoke Services - Associate service operations to <<Undefined>> processes that represent interactions with internal and external applications. Right-click an <<Undefined>> process and select **Invoke Existing Service**. You can also select a service operation from the Implemented by list in the **Implementation** tab of the <<Undefined>> process property sheet. See *Service Invocation* on page 234.
3. Organize Service Orchestration - Define the different orchestration items of the model. Create <<Send>> and <<Receive>> activities, implement other processes, and define decisions.
4. Check the Model - To verify that it conforms to the Sybase WorkSpace Business Process language standard.
5. Generate Sybase WorkSpace Business Process files - Select **Language > Generate Sybase WorkSpace Business Process 1.0 code**. Previously imported services are preserved in the WorkSpace environment. See *Generating for Sybase WorkSpace Business Process* on page 248.
6. Use the Business Process Service Editor to continue process implementation.

Importing EJB or Java Web Services

A developer designs and generates EJB or Java Web services in an OOM. He then wants to import them as service providers into a BPM in order to reuse them for automating processes in his orchestration BPM, he does not need any other type of services.

1. Design and Generate EJB or Java Web Services in an OOM component diagram - Select **Language > Generate Java Code** to generate the Web services files, which appear in the project in the WorkSpace navigator (see *Object-Oriented Modeling > Building OOMs > Working with Web Services > Generating Web Services for Sybase WorkSpace*).
2. Import the EJB or Java Web Services into a BPM as service providers - Select **Tools > Service Provider Import**. The new service providers appear in the Model Explorer (see *Importing a Service Provider from an OOM or a PDM* on page 107).
3. Invoke Services - Associate service operations to <<Undefined>> processes that represent interactions with internal and external applications. Right-click an <<Undefined>> process and select **Invoke Existing Service**. You can also select a service operation from the Implemented by list in the **Implementation** tab of the <<Undefined>> process property sheet. See *Service Invocation* on page 234.
4. Organize Service Orchestration - Define the different orchestration items of the model. Create <<Send>> and <<Receive>> activities, implement other processes, and define decisions.
5. Check the Model - To verify that it conforms to the Sybase WorkSpace Business Process language standard.
6. Generate Sybase WorkSpace Business Process files - Select **Language > Generate Sybase WorkSpace Business Process 1.0 code**. Previously imported services are preserved in the WorkSpace environment. See *Generating for Sybase WorkSpace Business Process* on page 248.
7. Use the Business Process Service Editor to continue process implementation.

Importing BPEL Files

A developer wants to implement in WorkSpace an already defined orchestration process. He can use PowerDesigner as a bridge for importing BPEL files into the Sybase WorkSpace Business Process.

1. Reverse engineer the BPEL files - Select **File > Reverse Engineer > Process Language**, specify a project and folder, select the BPEL4WS language in the Process Language list, and the different BPEL files to reverse engineer.
2. Change the process language - Select **Language > Change Current Process Language** and choose Sybase WorkSpace Business Process 1.0.

3. Check the Model - To verify that it conforms to the Sybase WorkSpace Business Process language standard.
4. Generate Sybase WorkSpace Business Process files - Select **Language > Generate Sybase WorkSpace Business Process 1.0 code**. See *Generating for Sybase WorkSpace Business Process* on page 248.
5. Use the Business Process Service Editor to continue process implementation.

Services

In Sybase WorkSpace, a *service* is an interface to an external application or a business process. BSM (Sybase Base Service Model) is the native language used to describe a service. The BSM file contains the description of the entire Web service expressed by several port types. Each port type defines different operations. Sybase WorkSpace Business Process supports the following types of Web service operations:

- *OneWay receive* operations with only input parameters and no return value
- *RequestReply* operations with input and output parameters

You design a Sybase WorkSpace Business Process service, port type and operation in the following way:

Sybase WorkSpace Business Process concept	PowerDesigner concept
Service	Service provider
Port type	Interface
Operation	Operation

Furthermore, PowerDesigner allows you to perform the following:

- Import WSDL files for importing existing SOAP Web services.
- Generate SOAP files in the WorkSpace environment from the imported WSDL files.
- Create new empty services to be implemented in WorkSpace with specialized editors.

Business Processes

A *business process* is a particular service implemented by the orchestration of other services including several activities linked by a graph. This graph defines the choreography of the Process. The business process may have fault, compensation, and timeout handlers that are not linked to the main choreography.

You design each business process with a business process model in PowerDesigner.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Business process	Business Process Model
Business process diagram	BPM default diagram

To design a business process with event handlers you have to:

- Create a composite process
- Attach event handlers to the composite process
- Define the main choreography in the sub-diagram of the composite process

The following extended attributes are used to generate properties of business process.

Name	Internal code	Description
Ad hoc process	AdHoc	AdHoc property of the business process.
Language	Language	Language of the process.
[none]	GUID	Global Unique Identifier of the model.
[none]	ModelGUID	Identifier of the model.
[none]	ModelName	Name of the model.
[none]	ProcessGUID	Defines the ID of the business process.

Variables

Variables are named data type instances that are accessible by all activities within a business process definition. Variables can have simple data types (string, integer, float, or boolean) or XSD complex data types.

Context variables are modeled as follows:

- Variable with message type - Variable with a message format data type.
- Message parameter - Message part of the message format. The message part can be of simple type or XSD type.
- Context variable with simple type - Variable with simple type.
- Context variable with complex type - Message format.

XSD schema files contain the definition of the complex data types used within the business process, and are used to define context variables that are accessible by the different activities. Each XSD file is a set of elements and complex types that can be used as a data type of a variable. You can associate XML elements to BPM variables or message parts.

To use XSD types in a BPM you have to import WSDL files as service providers into your model. The schema section of the WSDL file is imported as an XSD document (linked to an

XML model) associated with a service provider. The input/output messages of the different operations are imported as message formats containing parts of XSD type, which are used as data types of context variables for receive or send messages. For context variables, the XSD definition file is extracted and generated from the service provider DataSchema attribute. When the DataSchema is empty the following extended attribute is available from the message format with message parts property sheet:

Name	Internal code	Description
XSD file location	XSDLocation	Location of the XSD file that contains the definition of the types used by the parts of the message.

Designing XSD Data Types

XSD schema files contain the definition of the complex data types used within the business process. These data types are used to define context variables that are accessible by the different activities.

Each XSD file is a set of elements and complex types that can be used as a data type of a variable. In PowerDesigner, you can associate XML elements to BPM variables or message parts.

To use XSD types in a BPM you have to import a WSDL file into your model.

PowerDesigner imports WSDL files as service providers, the schema section of the WSDL file is imported as an XML model associated with a service provider as an XSD document. The input/output messages of the different operations are imported as message formats containing parts of XSD type. These message formats are used as data types of context variables, which are used to receive or send messages. For context variables, the XSD definition file is extracted and generated from the service provider DataSchema attribute. When the DataSchema is empty the following extended attribute is available from the message format with message parts property sheet:

Name	Internal code	Description
XSD file location	XSDLocation	Location of the XSD file that contains the definition of the types used by the parts of the message.

Partner Links

You design partner links using organization units in the business process model.

Each organization unit will define a partner link. To associate a partner link with a receive, send or invoke activity the user should use the Organization unit property of the process.

For more information on organization units, see *Organization Units (BPM)* on page 52.

Service Invocation

Services represent the interaction with internal and external applications, such as a database, Java applications or an ERP system. A service may contain several interfaces and each interface may contain multiple operations. Each operation has a schema in the service that defines the operation inputs and outputs. A service interaction represents an activity that invokes a service operation of the following type:

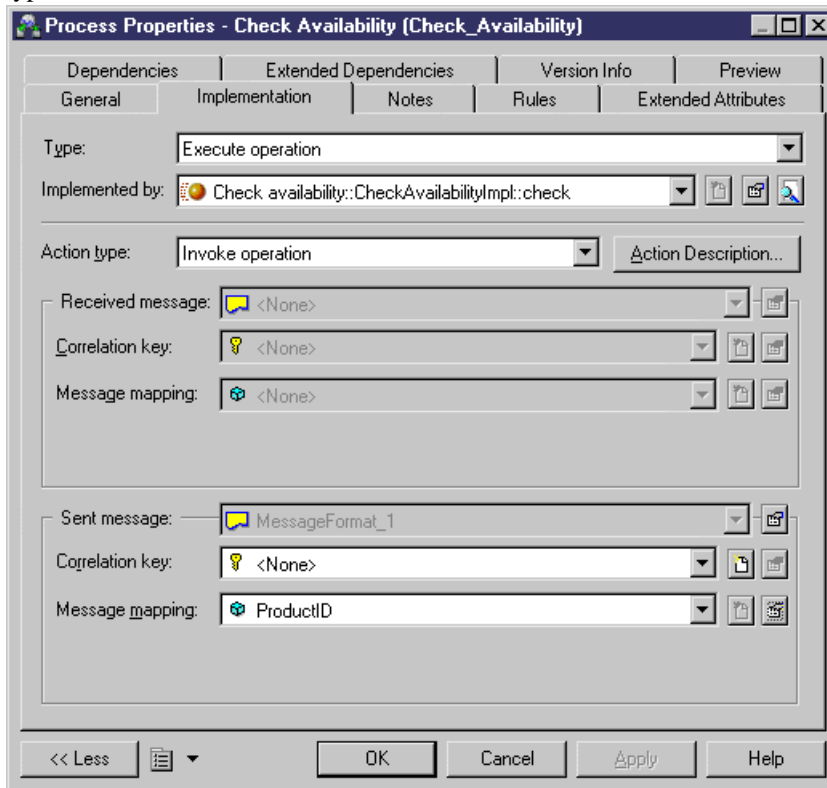
- *One-Way*: when a business process sends a message to a service
- *Request/Reply*: when a business process sends a message to a service and waits for a response from the service

You design an activity that invokes a service operation using a business process with the <<Invoke>> stereotype. This implies that you already have imported services from the WorkSpace environment.

You also need to define the Execute operation implementation type and make sure it is implemented by an operation in a service provider defined in the same model.

Designing a One-Way Service Invocation

You design an activity that invokes a One-Way service operation using an <<Invoke>> business process implemented by a One Way operation and with an Invoke Operation action type.



The Message mapping property of the Invoke operation action corresponds to the invoke variable defined in the Sybase WorkSpace Business Process service interaction. If you click the New button beside the Message mapping box, you create a variable with the same name as the received message.

You can associate one correlation key to the <<Invoke>> activity.

Designing a Request/Reply Service Invocation

You design an activity that invokes a Request/Reply service operation using an <<Invoke>> business process implemented by a Request-Response operation and with an Invoke operation action type.

The Message mapping property of the Invoke operation action corresponds to the invoke variable defined in the Sybase WorkSpace Business Process service interaction. If you click

the New button beside the Message mapping box, you create a variable with the same name as the received message.

Interface Activities

Receive and Send activities are interface activities. The first time you create a receive or a send activity, a new service provider called `ThisService` is created, containing a `Default` service interface that contains all the operations created automatically for `<<Send>>` and `<<Receive>>` business processes.

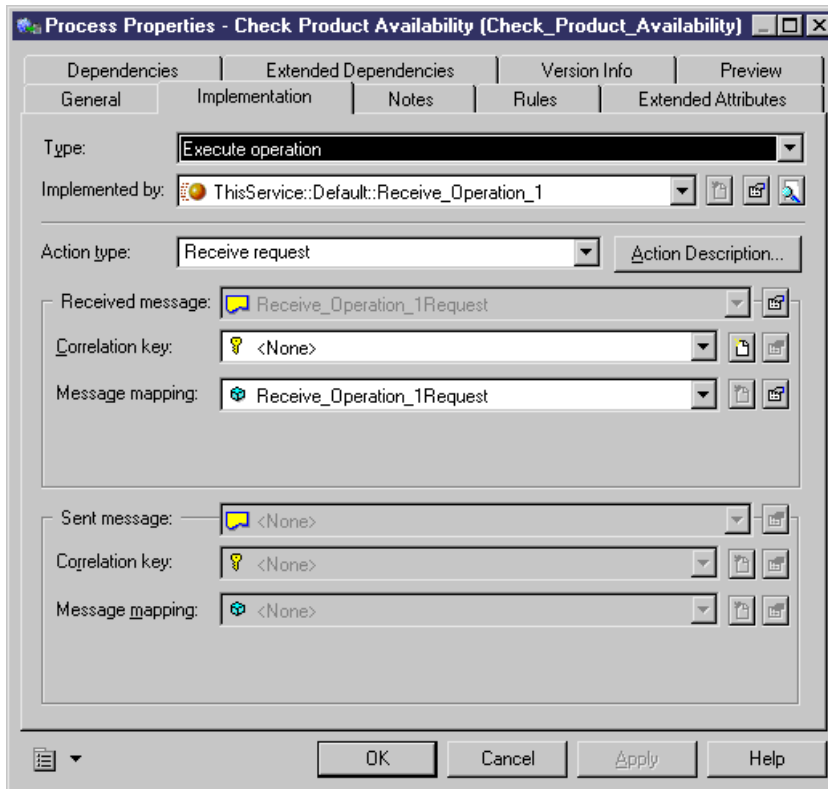
Designing a Receive Activity

A business process must have at least one receive activity connected to start the process.

You design a Receive activity using a `<<Receive>>` business process. To create a `<<Receive>>` business process you can right-click an `<<Undefined>>` process and select `Change to Receive` in the contextual menu. You can also select the `<<Receive>>` stereotype in the process property sheet.

The `<<Receive>>` stereotype is associated with an event handler that automatically performs the following actions in the model:

- Sets the implementation type of the process to `Execute` operation
- Creates a new `One Way` operation together with an input message. The word "Receive" is displayed before the name of this operation
- Sets the Action type to `Receive request`
- Creates a new variable in the Message mapping box of the received message with the same name as the received message format



Note: You can right-click an <<Undefined>> process and select the Change to Receive command.

Designing a Send Activity

You design a Send activity using a <<Send>> business process. To create a <<Send>> business process you can right-click an <<Undefined>> process and select Change to Send in the contextual menu and select the corresponding receive activity. You can also select the <<Send>> stereotype in the process property sheet.

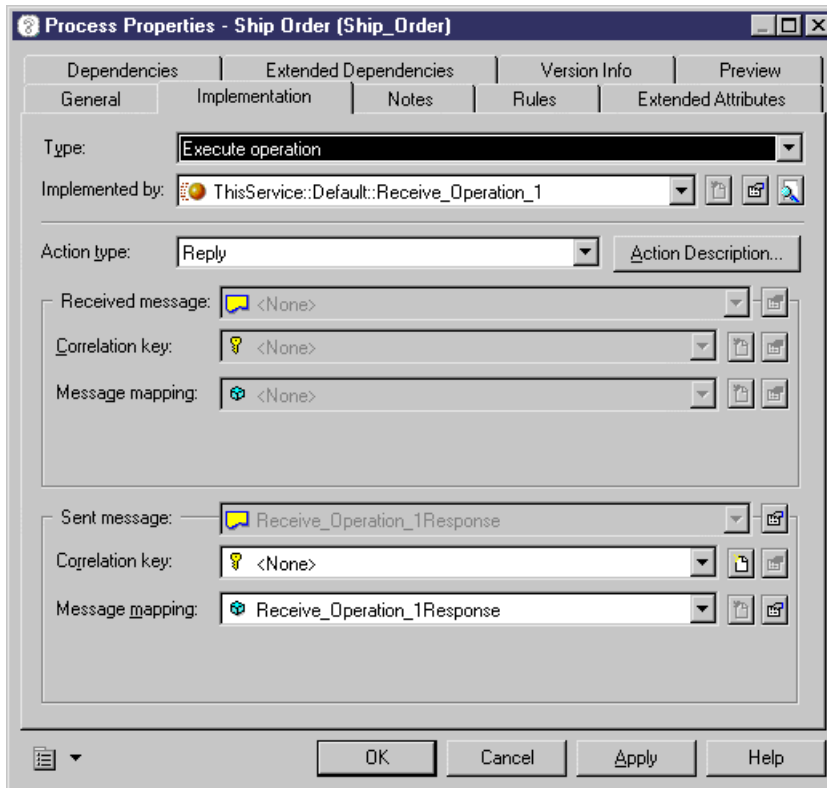
The <<Send>> stereotype is associated with an event handler that automatically performs the following actions in the model:

- Sets the implementation type of the process to Execute operation
- Depending on the selected receive activity, sets the Implemented by property to the same operation that implements the receive activity and changes the type of the operation to Request-Response
- Automatically creates the output message format corresponding to the Request-Response operation and displays it in the Message mapping box in the Sent message group box
- Sets the Action type property to Reply

- Creates a new variable with the same name as the sent message

If the activity sends a fault, you can right-click an <<Undefined>> process and select Change to Send Fault in the contextual menu.

When you assign the <<Send>> stereotype to a business process, you are prompted to select a corresponding Receive activity, because the Send is a reply to the Receive activity. In Sybase WorkSpace Business Process, you cannot associate more than one Send activity to a Receive activity.



Note: You can right-click an <<Undefined>> process and select the Change to Send command.

Assign Activities

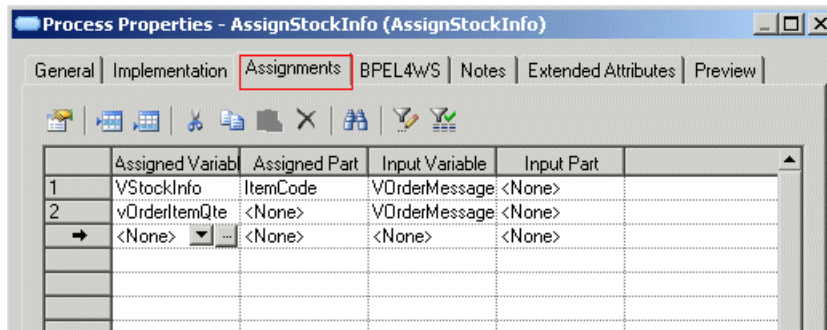
An assign activity is a sequence of atomic assign tasks. An assign task is an XPath expression that copies value from a variable to another variable. The assign task can be: a regular assign for setting the value of a context variable, a Get environment variable or a Get process Id that copies its value to a context variable, a Graft that adds nodes to an XML tree or a Prune that removes nodes from an XML tree.

The type of the assign task depends on the source of the task whereas the target of the assign task is always a context variable or an XPath expression.

To design an assign activity, it is strongly recommended to use the *Insert Assign* command in the flow contextual menu. This command automatically creates an assign activity with the correct input and assigned variables. You just need to define the source property (data transformation input variable) and the target property (data transformation assigned variable) part in the Assignments tab of the process property sheet.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Assign activity	Process with "Assign" implementation type.
Assign task	Data transformation defined in the Assignments tab of the process with "Assign" implementation type.
Sequence of assign tasks	Several data transformations defined in the Assignments tab of the process with "Assign" implementation type.
Source property for the assign task	Input variable attribute of the data transformation.
Target property for the assign task	Assigned variable attribute of the data transformation.
Part	Assigned part attribute of the data transformation.

When you create an assign activity with the "Assign" implementation type, this triggers the display of the Assignments tab:



In the Assignments tab you then have to:

- Click a blank line to create a data transformation object.
- Select an Assigned Variable to define the target property of the assign task.
- Select an Input Variable to define the source property of the elementary assign task.

Note: You can right-click an <<Undefined>> process and select the Change to Assign command.

Split-Join Activities

A split is a point in the business process where a single activity splits into two or more parallel activities. A join is a point in the business process where two or more parallel activities converge into a single common activity.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Join activity	Synchronization with the <<Join>> stereotype. A <<Join>> synchronization must have only one outgoing link.
Single rule activity (DataXORSplit)	Decision with the <<SingleRule>> stereotype. The decision must have two outgoing flows. The default output of the decision is defined by setting the "false" value in the condition alias of the flow outgoing from the decision. The rule expression cannot be defined in the BPM except when a business process is generated from a BPEL process; in this case the BPEL expression is reported in the condition expression of the decision.
Split activity (ANDSplit)	Synchronization with the <<Split>> stereotype.
Choice activity (EventXOR)	Decision with the <<Choice>> stereotype. When the Instantiate extended attribute is set to True, it indicates that the event creates an instance of the process. Outgoing flows must have the <<Exception>> stereotype.

Complex Activities

A complex activity is made of atomic activities. Complex activities can end normally, after a timeout event, or after a fault event. The complex activity catches exception events (exception, compensation, or timeout) by using exception handlers, compensation handler and timeout handler.

In Sybase WorkSpace Business Process, a complex activity defines a sub-process element that includes context variables: one default exception handler and optionally one handler per fault, one compensation handler, and several timeout handlers. These handlers are complex activities executed when the corresponding event happens.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Complex activity	Composite process. If you want to visualize the atomic activities of the complex activity you have to open the composite process sub-diagram.
Timer	Event with <<Timer>> stereotype. To associate a timer event to a composite process you have to attach events to flows coming from the composite process.

Complex activity exits are designed as flows outgoing from the composite process:

- A flow with a <<Fault>> event targets a <<FaultHandler>> activity
- A flow with a <<Timer>> event targets a <<TimeoutHandler>> activity
- A flow with a <<Compensation>> event targets a <<CompensationHandler>> activity

If you right-click the composite process, you can see different menu commands for creating these exits: Add Compensation Handler, Add Exception Handler, Add Timeout Handler, and Add Default Exception Handler.

Loop Activities

A loop activity is a complex activity with iterations.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Loop activity	Composite process with a Loop implementation type.
Loop condition	Use the Loop Expression box in the Implementation tab of the loop process.

The sub-diagram of a loop activity can contain a Break activity. You design this break activity as an end object with the <<Break>> stereotype.

Event Handling Activities

Events happen only within top-level process and complex activities. The different types of event handling activities are: exception, timeout, and compensation handling.

- Exceptions are caused by <<ThrowException>> activities or correspond to faults returned by operations invoked by service invocation activities. Exception handling is possible when the complex activity contains an exception activity or service invocation. These activities are modeled as follows:

- Exception handler - Composite process with the <<ExceptionHandler>> stereotype. The exception handler must be at the same level as the complex activity and connected with a flow to this activity.
- Connecting flow - Flow with a <<Fault>> event.
- Default exception handler - Flow with <<DefaultExceptionHandler>> stereotype.
- Timeout handling is possible only when the complex activity contains timers. These activities are modeled as follows:
 - Timeout handler - Composite process with the <<TimeoutHandler>> stereotype. The timeout handler must be at the same level as the complex activity and connected with a flow to this activity.
 - Connecting flow - Flow with a <<Timer>> event.
- Compensation handling is executed when a compensate activity is executed in the outer scope of the complex activity to compensate. A compensate activity must be used within an exception handler or another compensation handler; it cannot be linked to more than one complex activity. These activities are modeled as follows:
 - Compensation handler - Composite process with the <<CompensationHandler>> stereotype. The compensation handler must be at the same level as the complex activity and connected with a flow to this activity.
 - Connecting flow - Flow with a <<Compensation>> event.

Delay and Terminate Activities

A delay activity allows the user to define a pause on the runtime processing and the delay for continuing the runtime processing.

You design a delay activity using a process with the <<Delay>> stereotype. You can also use the corresponding tool in the Sybase WorkSpace Business Process toolbox.

The <<Delay>> process generates a <<Timer>> event.

A terminate activity allows the user to define different kinds of end activities in order to stop the runtime processing of a process or complex activity.

Sybase WorkSpace Business Process supports several terminate activities: Throw Fault, Compensate, Terminate, and Break.

Sybase WorkSpace Business Process concept	PowerDesigner concept
ThrowException activity	Process with <<ThrowException>> stereotype with Generate event implementation type and <<Fault>> event.
Compensate activity	Process with <<Compensate>> stereotype with Generate event implementation type and Compensate event.

Sybase WorkSpace Business Process concept	PowerDesigner concept
Terminate activity	End object with the <<Terminate>> stereotype.
Break activity	End object with the <<Break>> stereotype.

You can create these activities using the corresponding tool in the Sybase WorkSpace Business Process toolbox.

Using a process that generates a <<Fault>> event is useful if the fault name is explicitly caught by the exception handler of the parent complex activity. The name of the fault event will be the name of the fault caught by the exception handler.

Sequence Flow from Activities

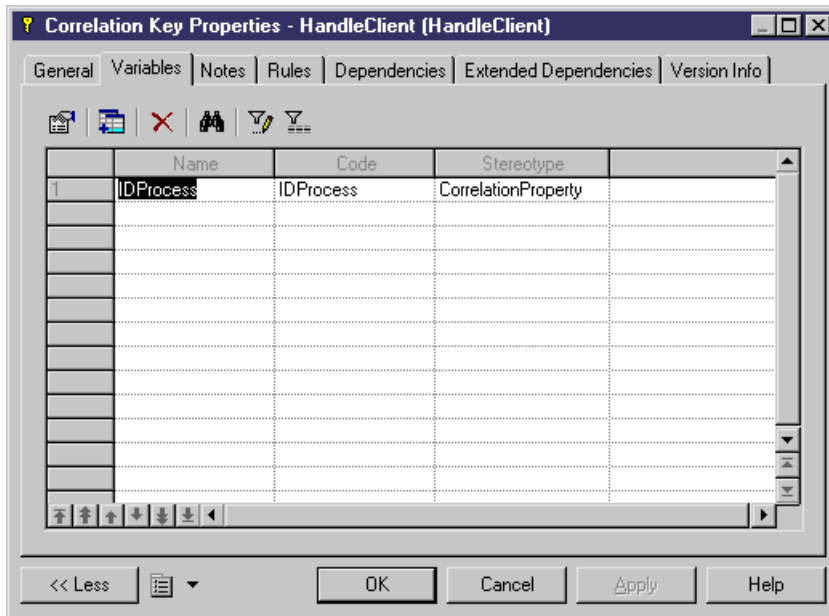
A sequence flow is a control flow between activities. The link goes from a fixed out anchor in the source object to a fixed in anchor in the target object. Some objects like Start, Choice, Split, and complex activity may have more than one output coming from the same anchor. SingleRule may have more than one output going out, each one from a specific anchor. The remaining activities may have only one sequence flow going out from one anchor except terminal activities.

You design a sequence flow using flows in the BPM. The constraints existing on outgoing flows are verified when you check your model.

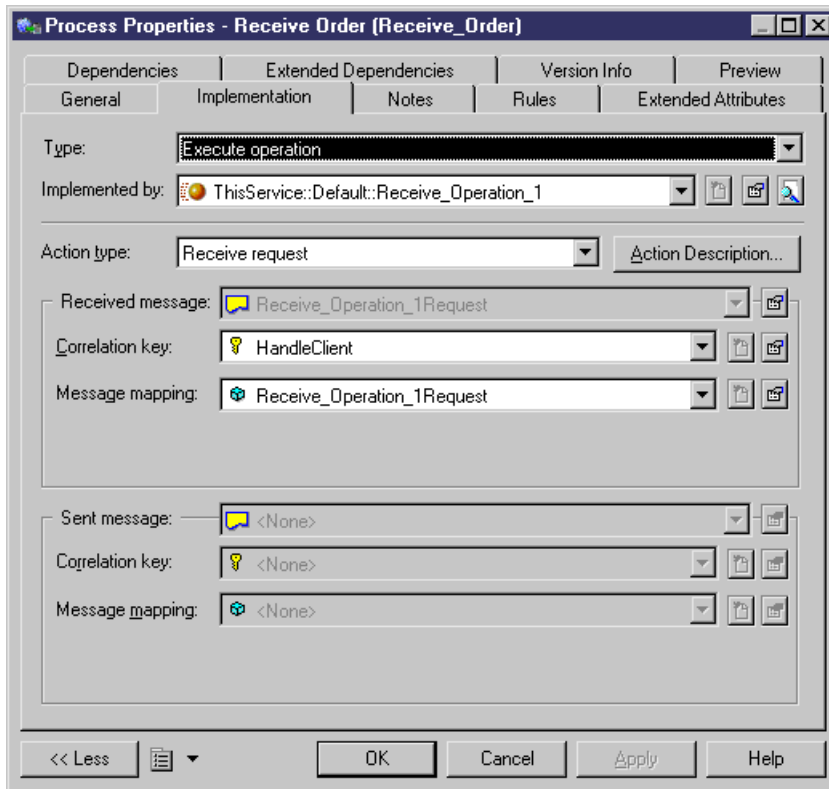
Correlations

A correlation is a set of identifying properties used by Sybase WorkSpace engine to identify the instance of a business process concerned by a received or a sent message.

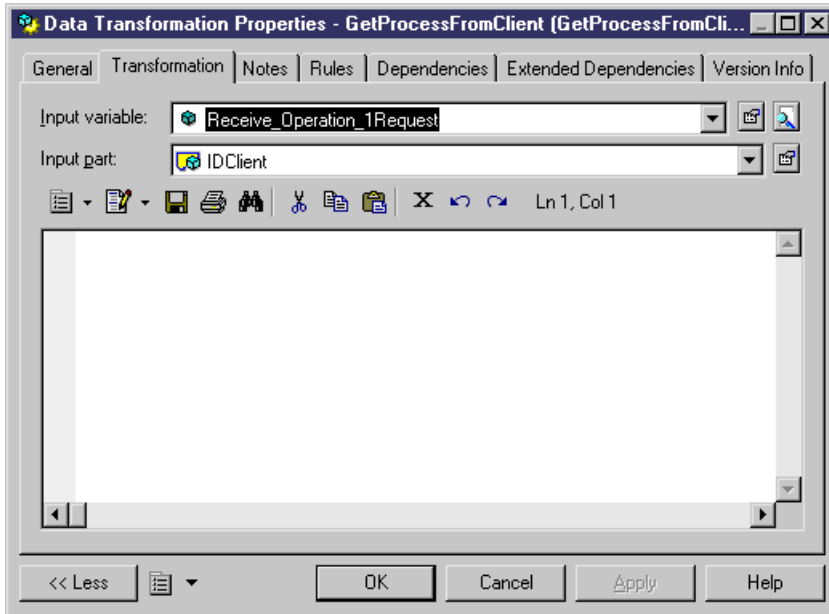
You design a correlation using a correlation key. You create a correlation key from the List of Correlation Keys or using the New command in the model contextual menu, and you add variables to the correlation key from the Variables tab of the correlation key property sheet. Correlation variables must have the <<CorrelationProperty>> stereotype.



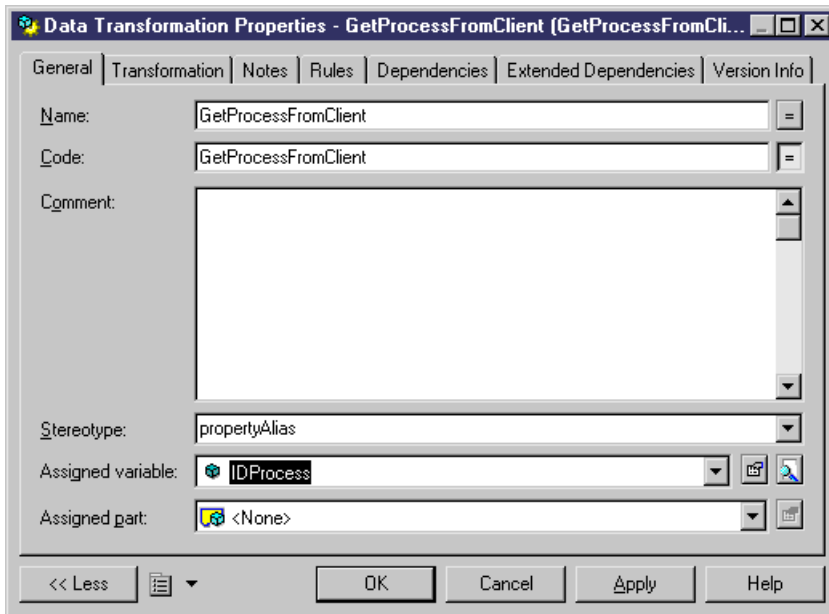
Receive, send and invoke activities share the correlation keys designed under the model or the parent complex activity. You can select a reception correlation for a receive activity, an emission correlation for a send activity, and the reception and emission correlation for an invoke activity.



Before associating a correlation key to an activity, you must design a data transformation that expresses how to obtain the properties of the correlation from the received or the sent message. You create one or several data transformations with the `<<propertyAlias>>` stereotype for each activity that uses a correlation key; each data transformation must have the reception/emission message mapping variable as the input variable:



and an assigned variable with the <<CorrelationProperty>> stereotype.



Switching to Sybase WorkSpace Business Process Language

You can change the target language of an imported BPEL file or an Analysis model in order to design for Sybase WorkSpace Business Process. To do so, you can generate a model with the Sybase WorkSpace Business Process language or you can change the target language of the model.

If You Use the Generation Feature

Make sure you select a single top level process in the source model. One model corresponds to one process in Sybase WorkSpace Business Process. The top-level process is removed and the flow chart can be defined directly on the top-level diagram

If You Use the Change Target Language Feature

You cannot change the target if the source model contains several top level processes.

When you switch to Sybase WorkSpace Business Process, some transformations are performed on the model to make it compliant with the Sybase WorkSpace Business Process language. These transformations are logged into the Output window.

Importing WorkSpace Services

The Import WorkSpace Services feature allows you to import any type of service as a service provider from a selected WorkSpace project to the current BPM. You can import business process services, database services, EJB services, Java services, message services, SOAP services, and transformation services.

This feature is only available when you are using the PowerDesigner plugin in the WorkSpace environment.

1. Right-click the BPM in the Model Explorer and select **Import WorkSpace Services**, or select **Tools > Import WorkSpace Services**.
2. Select the project containing the services you want to import in the Import Sybase WorkSpace Services dialog box and click Finish.

The services are imported as service providers in the model.

Note: If you re-import services from the same project into the same model, a merge dialog box is displayed to let you select the services you want to update into your model. In the left pane you can see the services available in WorkSpace, in the right pane you can see services in your model. Use the right pane to carefully define the result you want to obtain into your model.

For more information about the merge feature, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

Invoking WorkSpace Services

When you change the target language of your BPM from Analysis to Sybase WorkSpace Business Process you can rapidly implement <<Undefined>> business processes by invoking different types of services supported in Sybase WorkSpace.

The Invoke commands available from the contextual menu of an <<Undefined>> process allow you to define the type of implementation of a process by creating new services or reusing services of the following types: Transformation, Java, EJB, Message and Database.

When you invoke a new service, a new service provider is automatically created with the corresponding stereotype. This service provider contains a new operation that implements the <<Undefined>> process. The new operation is empty and needs to be later defined in the WorkSpace specialized editors.

When you invoke an existing service, an operation selection dialog box allows you to select the existing operation for implementing the current <<Undefined>> process.

Right-click an <<Undefined>> process and select an Invoke ... command.

or

Open the property sheet of an <<Undefined>> process and select the <<Invoke>> stereotype and select a service provider operation.

or

Select the service provider operation to invoke in the Model Explorer and drag it to an <<Undefined>> process, the <<Invoke>> stereotype is not assigned yet, you need to check the model to have automatic correction set the <<Invoke>> stereotype.

Generating for Sybase WorkSpace Business Process

This section explains some of the features of the Sybase WorkSpace Business Process language in order to generate code from objects in the BPM.

Defining Sybase WorkSpace Business Process Generation Parameters

You can set the following option, available from the Options tab of the Generation dialog box in Sybase WorkSpace Business Process:

Option	Description
Overwrite resource files (services, xsd, wsdl, java, map and ruleml)	These files are called resources in Sybase WorkSpace and are generated by PowerDesigner. You can choose to generate them once and not to overwrite them during next generations.

If the generated files have already been used in Sybase WorkSpace Business Process for implementation purpose, it is strongly recommended to avoid overwriting them the next time you generate from PowerDesigner as they may contain implementation details that may be lost during overwriting. Make sure you re-generate into a separate file or select the task "Increment version number of the generated files" in the Tasks tab.

Generation Task

The task "Increment version number of the generated files" can be used to add a version number to all the files generated from the current model. This is to avoid overwriting files that you have already started implementing in Sybase WorkSpace Business Process.

When you select this generation task, the GenerationVersionNumber extended attribute of the model and the service providers is automatically incremented each time you generate the model.

Generation Path

You must generate in an existing project folder, or in a folder at the Eclipse workspace root directory, in this case, this folder becomes a project folder. The following folder hierarchy is not allowed as folder1 is not a project folder:

```
d:\Sybase\WorkSpace\Eclipse\workspace\folder1\folder2
```

Generating Sybase WorkSpace Business Process Files

When you generate for Sybase WorkSpace Business Process, the following types of files are generated:

- The *model.bpmn* file is an XML file that contains the bpmn description of the business process model. Each activity is defined as a NotationBag element and each flow is defined as a SequenceFlow bag.
- The *svc_bpmn* file is an XML file generated from the service provider called ThisService. All the operations defined in the Default service interface are generated in the business process service file.

- The *bpmn.gem* file is an XML file is generated from the default business process diagram of the model.
- Service files (*svc_xyz*) for new services.
- An *XSD* file is generated for each BPM message format having message parts, this file is used by a service provider.

Note: It is recommended to use the Check Model feature before generating in order to check the validity of your model.

1. Select **Language > Generate Sybase WorkSpace Business Process 1.0 Code** to display the Generation dialog box.
2. Type a destination directory for the generated file in the Directory box.

or

Click the Select a Path button to the right of the Directory box and browse to select a directory path.

3. Select the Check Model check box if you want to verify the validity of your model before generation.
4. Click the Selection tab and select the objects to include in the generation from the sub-tabs at the bottom of the page.

Note: All processes of the model, are selected and displayed by default. You use the Select tools to the right of the Folder Selection list to modify the selection.

5. Click the Options tab and select a value for each required option.
6. Click OK to generate.

A Progress box is displayed. The Result list displays the files that you can edit. The result is also displayed in the Generation tab of the Output window, located at the bottom of the main window.

All Sybase WorkSpace Business Process files are generated in the destination directory.

Electronic Business XML (EbXML)

A collaborative BPM is used to design Business-to Business exchange. It displays the exchange of messages between partners in *top-level processes*, each of which can be decomposed into a choreography that represents the sequence of message exchanges. PowerDesigner supports ebXML BPSS v1.01 and v1.04.

ebXML BPSS is a global electronic business standard that allows enterprises of any size, and in any location to safely and securely transact business through the exchange of XML-based messages. The Specification Schema deals with the business process. It identifies such things as the overall business process, the roles, transactions, identification of the business documents used (the DTDs or schemas), document flows, legal aspects, security aspects, business level acknowledgements, and status. A Specification Schema can be used by a software application to configure the business details of conducting business electronically with another organization.

The ebXML BPSS supports the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations.

A specification created using the ebXML BPSS is referred to as an ebXML Business Process Specification. This specification is a declaration of the partners, roles, collaborations, choreography and business document exchanges that make up a business process.

You build a collaborative BPM to describe the collaborations between partners that are all considered at the same level.

In a collaborative BPM, atomic processes correspond to a predefined message exchange between two partners. This exchange only contains a request potentially followed by a response.

Atomic processes (activities) must be defined under a top-level process that represents the sequence of exchanges between partners.


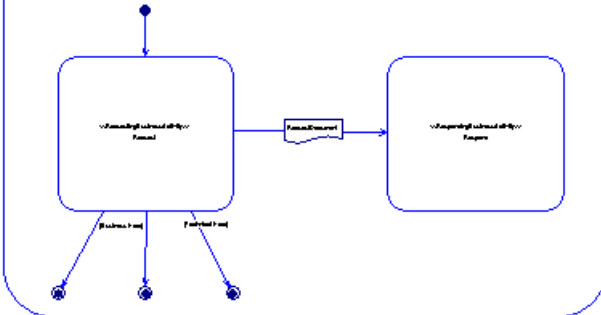
The collaborative BPM describes all the binary collaborations that the process plays with all partners.


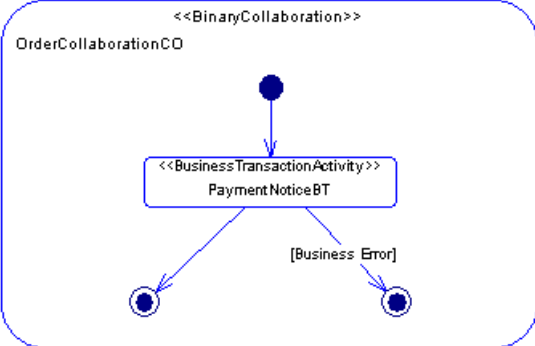

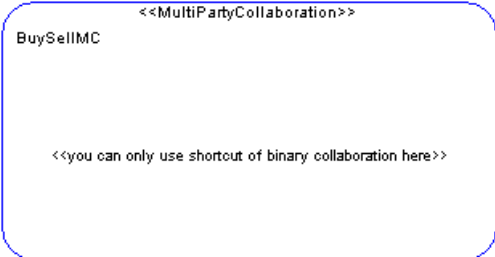
Objects you use to build the collaborative BPM are mainly the same as those you use to build an analysis BPM. However, some constraints are added to meet ebXML specific requirements. They are defined in the following sections.

EbXML Business Process Specification Schema (BPSS)

The main concepts of ebXML BPSS are business collaboration (binary and multiparty), business transactions, business document flows, and choreography.

You use the ebXML BPSS specific toolbox to create Business Transactions and Business Collaborations in a BPM:

Tool	Description
 ...	<p>Business Transaction - Creates a default composite process bearing a Business Transaction stereotype, with a sub-process in composite view mode. Each of the various BT tools creates a Business Transaction process with distinct flows between the requesting and the responding activities. In the ebXML toolbox, reading from left to right, tools are available to create:</p> <ul style="list-style-type: none"> • A simple requesting flow • A requesting flow with a responding flow • A requesting flow, responding flow, and a receipt acknowledgment for the requesting flow • A requesting flow, responding flow, and receipt and acceptance acknowledgments for the requesting flow • A requesting flow, responding flow, receipt and acceptance acknowledgments for the requesting flow, and a receipt acknowledgment for the responding flow <div data-bbox="373 927 982 1333" style="border: 1px solid blue; padding: 10px; margin: 10px 0;"> <p style="text-align: center;"><<BusinessTransaction>></p> <p>PaymentNotice BT</p>  </div> <p>For information about composite views, see <i>Working with Composite Views</i> on page 49.</p>

Tool	Description
	<p>Binary Collaboration - Creates a default composite process bearing a Binary Collaboration stereotype with a sub-process in composite view mode, attaches two organization units as initiating and responding roles, and creates an initial choreography inside the Binary Collaboration process. You must complete the choreography by specifying a Business Transaction process for implementation.</p> 
	<p>MultiParty Collaboration - Creates a default composite process with the following message <<you can only use shortcut of binary collaboration here>>:</p> 

Business Transaction

Each *Business Transaction* consists in one or two predefined Business document flows. A Business Transaction may be additionally supported by one or more Business Signals.

A Business Transaction is the atomic interaction (in a trading arrangement) between two business partners. One party plays the Requesting role, the other plays the Responding role. This interaction always results in one Business Document Flow from the requesting role to the responding role and possibly one or more Business Document Flow in the inverse sense. A Business Transaction may be additionally supported by one or more Business Signals.

Like the Binary Collaboration, a Business Transaction is a reusable protocol between two roles. The protocol is reused by referencing it from a Binary Collaboration, through the use of a Business Transaction Activity. In a Business Transaction Activity the roles of the Binary Collaboration are assigned to the execution of the Business Transaction.

A Business Transaction can either succeed or fail:

- If it succeeds, it may be designated as legally binding between the two partners, otherwise it governs their collaborative activity.
- If it fails, it is null and void, and each partner must relinquish any mutual claim established by the transaction.

Business Collaboration

A *Business Collaboration* consists in a set of roles that interact through *Business Transactions* by exchanging *Business Documents*.

Business Collaboration can either be a:

- *Binary Collaboration* between two roles only
- *MultiParty Collaboration* between more than two roles, but such MultiParty Collaborations are always a synthesis from two or more Binary Collaborations

Binary Collaborations are expressed as a set of Business Activities, which can consist in conducting a Business Transaction (Business Transaction Activity) or another complete Binary Collaboration (Collaboration Activity). The business activities are sequenced in a choreography. Placing a purchase order or requesting a catalog can be an example of a Business Transaction Activity; negotiating a contract can be an example of a Collaboration Activity.

The ability of a Binary Collaboration to have activities that execute other Binary Collaborations is the key to recursive compositions of Binary Collaboration, and to the re-use of Binary Collaborations.

A Binary Collaboration must have exactly two associated roles (Initiating and Responding). To do so, you must define role associations in the top-level diagram.

Only one start is allowed in a Binary Collaboration. Its sub-processes must always be implemented by a Business Transaction process or a Binary Collaboration process. Decision objects are not allowed and the activities must be created using **Alt**+drag and drop.

MultiParty collaboration is a set of Binary Collaboration between business partners. Each partner plays one or more roles in the collaboration.

A MultiParty collaboration can only contain organization units with the icon representation and shortcuts of Binary Collaborations linked to each other using extended dependencies.

Business Document Flows

A business transaction is realized using *Business Document flows* between the requesting and responding roles. There is always a requesting Business Document, and optionally a responding Business Document, depending on the desired transaction semantics, e.g. one-way notification vs. two-way conversation. To do so, you must define a message format on the flow inside the Business Transaction. It is the only way to exchange data. You cannot define a message format on the flow inside a Binary Collaboration.

The actual document definition is achieved using the ebXML core component specifications, or by some methodology external to ebXML but resulting in a DTD or Schema that an ebXML Business Process Specification can point to. This schema is referenced with a message format object.

Choreography

The *Business Transaction Choreography* describes the ordering and transitions between business transactions or sub-collaborations within a binary collaboration. The choreography is described in the ebXML Business Process Specification Schema using activity diagram concepts such as start state, completion state, activities, synchronizations, transitions between activities, and guards on the transitions.

ebXML Top-Level Diagrams and Processes

When you create a collaborative BPM, you must always define a *top-level diagram* under a model or a package to represent the subject of the process model and set the model scope and orientation.

The top-level diagram must contain only:

- Top-level processes - global services that do not belong to a graph but describe their behavior in sub-graphs (a stand alone composite process defined at the package level) using atomic activities and orchestration elements. For ebXML, top-level process can design:
 - An atomic transaction between two partners (Business Transaction)
 - A Binary Collaboration between two partners that involves the invocation of one or a sequence of Business Transactions (Binary Collaboration)
 - A Collaboration between several partners that involves the invocation of several Binary Collaborations (Multiparty Collaboration)

For more information, see *EbXML Business Process Specification Schema (BPSS)* on page 252.

- Organization units (see *Organization Units (BPM)* on page 52)

Note: In a collaborative BPM, *organization units* are always displayed using the icon representation to designate a user of a process that is engaged in a collaboration with a partner. The role association object is used to define the link with the process. The swimlane representation can only be used in diagrams contained in composite processes to ease the definition of invoking or invoked partners for processes implemented by operations (see *Example: Using the Execute Operation Implementation Type* on page 40).

If you used traceability links to a link an organization unit to a process to express a binary collaboration, they are transformed into role association objects.

-
- Role associations (see *Role Associations (BPM)* on page 73)

Designing a Business Transaction

You design a Business Transaction using a process with a <<BusinessTransaction>> stereotype.

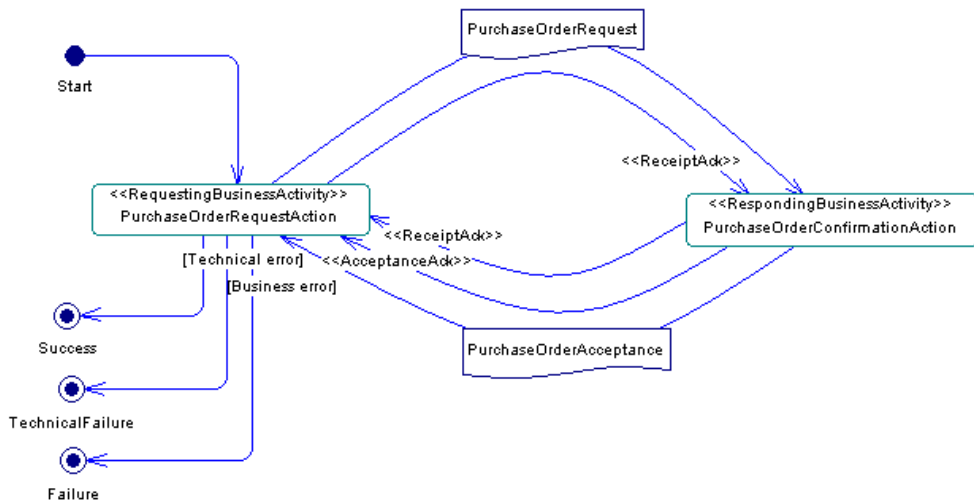
The following extended attributes (accessible from the Extended Attributes tab of the process property sheet) apply to the Business Transaction process. The table shows what attributes are available for ebXML BPSS 1.01 or for ebXML BPSS 1.04 process language:

Name	Internal code	Description	BPSS 1.01	BPSS 1.04
Requires guaranteed delivery	IsGuaranteedDeliveryRequired (true false) "false"	Both partners must agree to use a transport that guarantees delivery	Yes	Yes
Pre conditions	preConditions	A description of a state external to this transaction that is required before this transaction can start	Yes	No
Post conditions	postConditions	A description of a state that does not exist before the execution of this transaction but will exist as a result of the execution of this transaction	Yes	No
Begins when	beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to start	Yes	No
Ends when	endsWhen	A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to end	Yes	No
Requires secure transport	IsSecureTransportRequired	Both partners must agree to exchange business information using a secure Delivery Channel. The following security controls ensure that business document content is protected against unauthorized disclosure or modification and that business services are protected against unauthorized access. This is a point-to-point security requirement. Note that this requirement does not protect business information once it is off the network and inside an enterprise	No	Yes

Name	Internal code	Description	BPSS 1.01	BPSS 1.04
Pattern	Pattern	The optional name of the pattern that this business transaction is based on	No	Yes

Requesting and Responding Business Activities

The Business Transaction process is a composite process with a sub-diagram that designs the Business Signal exchanges between partners. This sub-diagram contains one *Responding Business Activity* (a process with <<RespondingBusinessActivity>> stereotype) and one *Requesting Business Activity* (a process with <<RequestingBusinessActivity>> stereotype):



The *Signals* are designed with flows between Requesting and Responding activities. The <<ReceiptAck>> and <<AcceptanceAck>> flow stereotypes allow the design of a Receipt Acknowledgment Signal and Acceptance Acknowledgment Signal. The Request Document and Response Document are also designed with flows with message formats that design the document format.

The following extended attributes (accessible from the Extended Attributes tab of the process property sheet) apply to both the Requesting and Responding Business Activity processes:

Name	Internal code	Description
Requires authorization	IsAuthorizationRequired (true false) "false"	If a partner role needs authorization to request a business action or to respond to a business action, then the sending partner role must sign the business document exchanged and the receiving partner role must validate this business control and approve the authorizer. A responding partner must signal an authorization exception if the sending partner role is not authorized to perform the business activity. A sending partner must send notification of failed authorization if a responding partner is not authorized to perform the responding business activity
Requires intelligible check	IsIntelligibleCheckRequired (true false) "false"	Receiving partner role must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt
Requires non repudiation (true false) "false"	IsNonRepudiationRequired (true false) "false"	If non-repudiation of origin and content is required, then the business activity must store the business document in its original form for the duration mutually agreed to in a trading partner agreement. A responding partner must signal a business control exception if the sending partner role has not properly delivered their business document. A requesting partner must send notification of failed business control if a responding partner has not properly delivered their business document

Name	Internal code	Description
Requires non repudiation receipt	isNonRepudiationReceiptRequired (true false) "false"	Both partners agree to mutually verify the receipt of a requesting business document and that the receipt must be non-reputable. A receiving partner must send notification of failed business control (possibly revoking a contractual offer) if a responding partner has not properly delivered its business document. Non-repudiation of receipt provides the following audit controls: Verify responding role identity (authenticate) – Verify the identity of the responding role (individual or organization) that received the requesting business document Verify content integrity – Verify the integrity of the original content of the business document request
Time to acknowledge acceptance	TimeToAcknowledgeAcceptance (specific to the Requesting Activity)	The time a receiving role has to acknowledge business acceptance of a business document
Time to acknowledge receipt	TimeToAcknowledgeReceipt	The time a receiving role has to acknowledge receipt of a business document
Number of retries	retryCount	Business level retries. For requesting business activity and available only for BPSS 1.04

A specific ebXML toolbox allows you to directly create a Business Transaction composite process in the current diagram and initialize the composite process with the Requesting and Responding Activities. The following table summarizes the distinct flows that exist between the Requesting and the Responding Activity:

Flow	Request Document	Response Document	Receipt on Response	Acceptance on Response	Receipt on Request
Without response	Yes	No	No	No	No
Default	Yes	Yes	No	No	No
With receipt on response	Yes	Yes	Yes	No	No

Flow	Request Document	Re- response Docu- ment	Receipt on Response	Acceptance on Re- sponse	Receipt on Request
With receipt and acceptance on response	Yes	Yes	Yes	Yes	No
Full	Yes	Yes	Yes	Yes	Yes

Document Envelope

A *Document Envelope* conveys business information between the two roles in a Business Transaction. One Document Envelope conveys the request from the requesting role to the responding role, and another Document Envelope conveys the response (if any) from the responding role back to the requesting role. This concept is designed with a flow and a message format in the Business Process Model.

The following extended attributes (accessible from the Extended Attributes tab of the message format property sheet) apply to the Document Envelope:

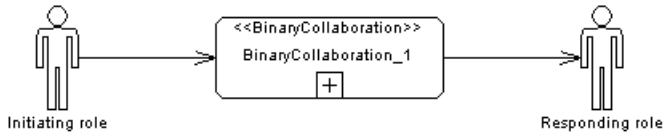
Name	Internal code	Description
DocTypeDocu- mentation	DocTypeDocumentation	Documentation of DocumentType
Is positive response	IsPositiveResponse	If TRUE the Document Envelope is intended as a positive response to the request. This parameter is only relevant on the response envelope
Is authenticated	isAuthenticated (true false) "false"	There is a digital certificate associated with the document entity. This provides proof of the signatory identity
Is confidential	isConfidential (true false) "false"	The information entity is encrypted so that unauthorized parties cannot view the information
Is tamper proof	isTamperProof (true false) "false"	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity

Designing a Binary Collaboration

You design a Binary Collaboration using a process with a `<<BinaryCollaboration>>` stereotype.

Organization units design the initiating and responding roles of the Binary Collaboration. They are linked to the process using roles associations.

Example:

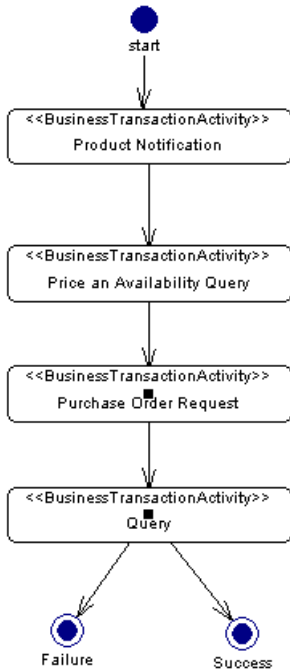


The following extended attributes (accessible from the Extended Attributes tab of the process property sheet) apply to the Binary Collaboration process:

Name	Internal code	Description
Pattern	Pattern	Optional name of the pattern on which this binary collaboration is based
Pre conditions	preConditions	Description of a state external to this collaboration/transaction that is required before this collaboration/transaction ends
Post conditions	postConditions	Description of a state that does not exist before the execution of this collaboration/transaction but will exist as a result of the execution of this collaboration/transaction
Begins when	beginsWhen	Description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to start
Ends when	endsWhen	Description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to end

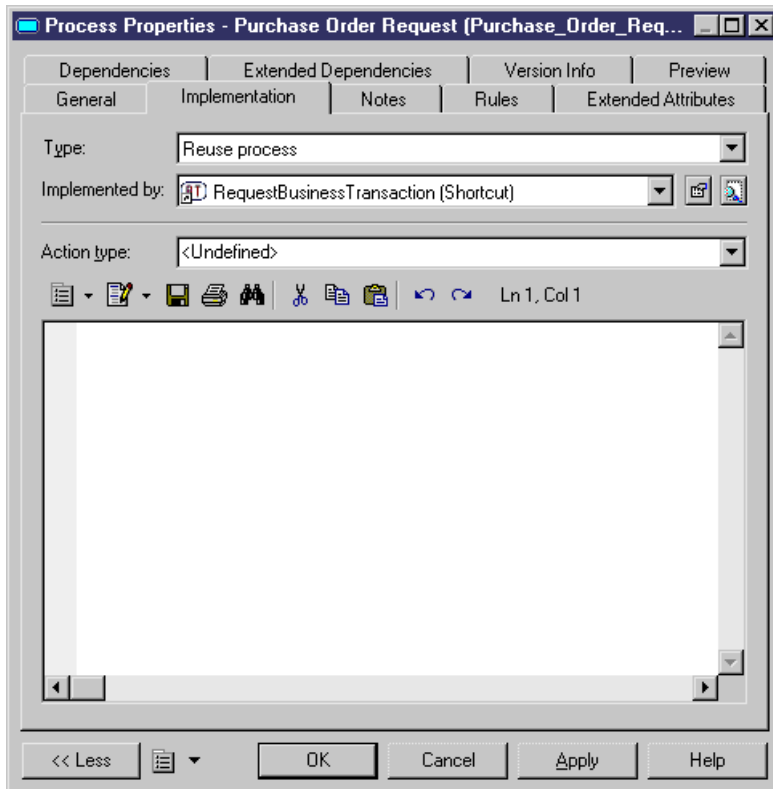
Choreography

The Binary Collaboration is a composite process with a sub-diagram designing the *choreography*.

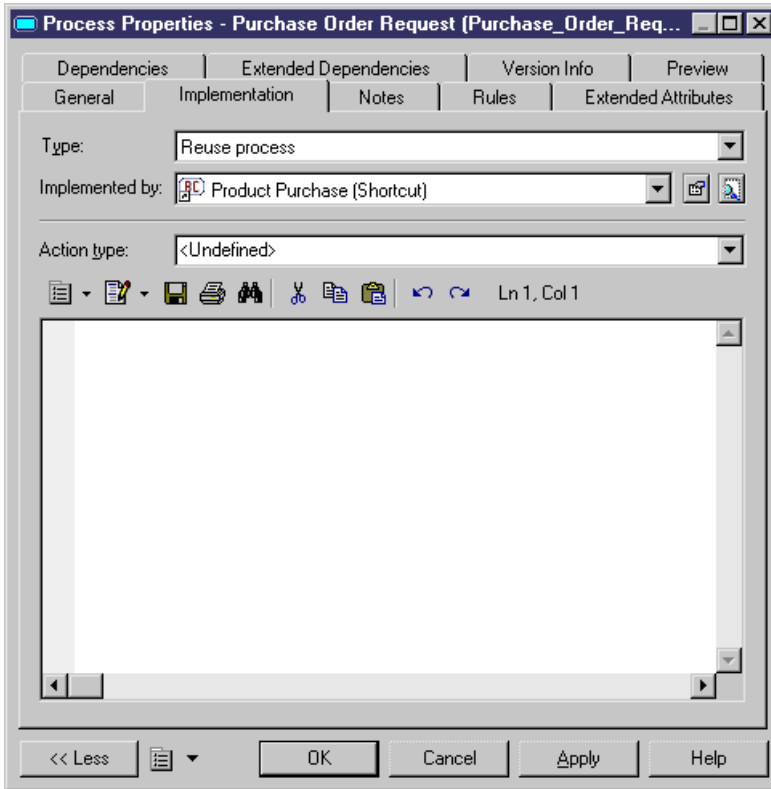


The choreography is a sequence of activities, which can be:

- A Business Transaction Activity. It specifies the use of a shared Business Transaction in the context of a Binary Collaboration and is designed with a process with the <<BusinessTransactionActivity>> stereotype. This Business Transaction Activity process references the Business Transaction process using the Implemented By attribute that you access by selecting Reuse Process in the Type list of the Implementation tab in the Business Transaction Activity process property sheet:



- A Collaboration Activity. It specifies the use of a shared Binary Collaboration in the context of another Binary Collaboration. It is designed with a process with the <<CollaborationActivity>> stereotype. This Collaboration Activity process references the Binary Collaboration process using the Implemented By attribute that you access by selecting Reuse Process in the Type list of the Implementation tab in the Business Transaction Activity process property sheet:



The following extended attributes (accessible from the Extended Attributes tab of the process property sheet) apply to the Business Transaction Activity process. The table shows what attributes are available for ebXML BPSS 1.01 or for ebXML BPSS 1.04 process language:

Name	Internal code	Description	BPSS 1.01	BPSS 1.04
Is concurrent	isConcurrent (true false) "false"	If the business transaction activity is concurrent, then more than one business transaction can be opened at a time. If the business transaction activity is not concurrent, then only one business transaction activity can be opened at a time	Yes	Yes

Name	Internal code	Description	BPSS 1.01	BPSS 1.04
Is legally binding	isLegallyBinding (true false) "true"	The Business Transaction performed by this activity is intended by the trading parties to be binding. Default value is True	Yes	Yes
isSynchronous	isSynchronous (true false) "false"	The Business Transaction is intended to be performed by this activity in a synchronous way	Yes	Yes
Pre conditions	preConditions	A description of a state external to this transaction that is required before this transaction can start	No	Yes
Post conditions	postConditions	A description of a state that does not exist before the execution of this transaction but will exist as a result of the execution of this transaction	No	Yes
Begins when	beginsWhen	A description of an event external to the collaboration/transaction that normally causes this collaboration/transaction to start	No	Yes
Ends when	endsWhen	A description of an event external to this collaboration/transaction that normally causes this collaboration/transaction to end	No	Yes

TimeToPerform

The *timeToPerform* is the duration from the time a Business Transaction Activity initiates the first Business Transaction until there is a transition back to the initiating Business Transaction Activity. To define *timeToPerform* for a Business Transaction Activity you can use the Duration attribute in the property sheet of the process.

Flow

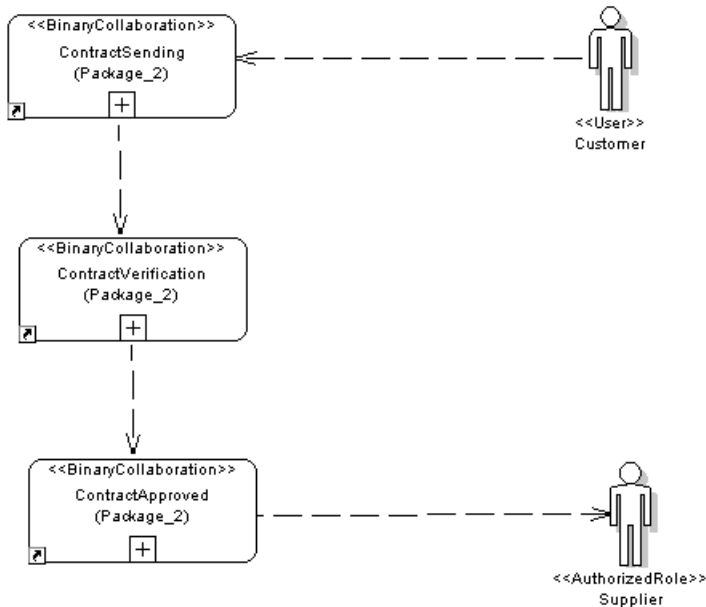
The flows in the choreography specify transitions between activities. The following extended attributes (accessible from the Extended Attributes tab of the flow property sheet) apply to the flow:

Name	Internal code	Description
On initiation	onInitiation (true false) "false"	Used to specify a nested BusinessTransactionActivity and a second transaction is performed before returning to this transaction to send the response back to the original requestor
Expression language	expressionLanguage	Specifies the language in which the condition expression has been written
Condition guard	conditionGuard	The condition that guards the transition. This attribute is available only for model with BPSS 1.04; it can take one of the following values: ProtocolSuccess, AnyProtocolFailure, RequestReceiptFailure, RequestAcceptanceFailure, ResponseReceiptFailure, ResponseAcceptanceFailure, SignalTimeout, ResponseTimeout, BusinessSuccess, BusinessFailure, Success, Failure. (For BPSS 1.01, the type of the flow indicates the condition)

Designing a MultiParty Collaboration

You design a MultiParty Collaboration using a process with a <<MultiPartyCollaboration>> stereotype.

The MultiParty Collaboration process is a composite process with a sub-diagram describing the synthesis of Binary Collaborations and Business Partner Roles:



This sub-diagram uses organization units to design the Business Partner Roles and role associations to design the Performs link between organization units and processes. The processes used in this MultiParty Collaboration diagram are Binary Collaboration processes. We use shortcuts of processes to reuse Binary Collaborations defined under another package or model.

Generating EbXML BPSS Files

The generated files have the .XML extension.

1. Select **Language > Generate ebXML Code** to display the ebXML generation dialog box.
2. Type a destination directory for the generated file in the Directory box.

or

Click the Select a Path tool to the right of the Directory box and browse to select a directory path.

3. <optional> Select the Check Model check box if you want to verify the validity of your model before generation.
4. Click the Selection tab and select the objects to include in the generation from the different sub-tabs.

Note: All objects of the model, including those grouped into packages, are selected and displayed by default. You use the selection tools to the right of the Select Location list to

modify the selection. The Include Sub-Packages tool allows you to include all objects located within packages.

5. Click the Options tab and select a value for each required option.
6. Click OK to generate.

A Progress box is displayed. The Result list displays the files that you can edit. The result is also displayed in the Generation tab of the Output window, located in the bottom part of the main window.

All ebXML files are generated in the destination directory.

Selecting EbXML Generation Options

You can set the following options, available from the Options tab of the Generation dialog box in ebXML:

Option	Description
Generate description	Allows the generation of Process Description attribute as an ebXML documentation tag
Default file name	Defines the default name of the file to generate
Use XPathS	Allows the generation of XPathS when a BPSS document references Business Transactions (BT) or Binary Collaborations (BC). When this option is set to False, the BT or BC is referenced by its name only
Use relative paths	Allows the generation of file name without the full path. This is used for the generation of specification location of Business Document in ebXML. This option has no effect when the location is an http URL. It is only considered when the location is an external file. The specification location will be generated as a complete URL (full path name) when the option is set to False and as a file name only when the value is True
Generate SchemaRef	Enables the generation of the XML schema namespace in BPSS document
Generate NameIDs	Enables the generation of the nameID for each element of the BPSS document
Generate CPA Template	Enables the generation of a second file that contains a template for CPA associated with ebXML BPSS

Reverse Engineering EbXML BPSS

You can reverse engineer from an XML file an ebXML Business Process Specification Schema (BPSS) and build a Business Process Model from it. The supported versions of the Business Process Specification Schema are 1.01 and 1.04.

1. Select **Language > Reverse Engineer ebXML BPSS File** to display the Reverse ebXML BPSS File dialog box.
2. Click the Add button in the Selection tab to display a standard Open dialog box.
3. Select the .xml files you want to reverse and click Open.

Note: You select several files simultaneously using the **Ctrl** or **Shift** keys.

The Reverse ebXML BPSS File dialog box displays the files you selected.

4. Click OK.

A progress box is displayed. If the model in which you are reverse engineering already contains data, the Merge Models dialog box is displayed.

For more information on merging models, see *Core Features Guide > The PowerDesigner Interface > Comparing and Merging Models*.

The reversed objects are added to your model. They are visible in the diagram and in the Browser. They are also listed in the Reverse tab of the Output window, located at the bottom part of the main window.

Note: You can also reverse engineer ebXML BPSS files from the File menu and create a new BPM. For more information, see *Chapter 7, Generating and Reverse Engineering Process Languages* on page 169.

EbXML Collaboration Protocol Agreement (CPA)

ebXML CPA is an XML format for describing agreement information for partners that agree to collaborate. This agreement is based on the ebXML architecture. In addition to support for ebXML BPSS, PowerDesigner can optionally generate an ebXML Collaboration Protocol Agreement (CPA) template for v1.04.

An ebXML CPA document defines a sequence of actions the trading partners take to execute a business process, messages sent from one partner to another, typically with attached business documents. Actions are CanSend/CanReceive Action pairs between the partners, so if the CPA defines a CanSend element for Partner A, then it must also define a corresponding CanReceive element for Partner B. The CPA template generation is, therefore, mainly based on the choreography of Binary Collaboration, Business Documents and signals exchanged during each Business Transaction.

Each CanSend/CanReceive action constitutes an elementary agreement between the two partners. It defines agreed technical parameters related to message exchange such as delivery

channel protocol and security, or message packaging. PowerDesigner provides CPA specific extended attributes to define these technical parameters.

An ebXML Collaboration Protocol Profile defines a business partner's technical capabilities to engage in electronic business collaborations with other partners by exchanging electronic messages. An ebXML CPA documents the technical agreement between two (or more) partners to engage in electronic business collaboration. The CPA can be seen as an intersection of the collaborative partners CPP (Collaborative Partner Profile).

The Collaborative Partner Agreement captures critical information for communications between applications and business processes and also records specific technical parameters for conducting electronic business.

Information agreed by the two parties includes BPSS documents, choreography (Requests, Responses, ordering), and parameters for exchanging messages: transport (protocol, security, addresses), document exchange (protocol, security), and message packaging.

A CPA document can be produced in the following ways:

- Based on the same BPSS document instance, two partners negotiate technical and/or functional details of their collaboration and draw up the result in the form of a CPA
- Based on the CPP of each partner, two partners try to match their technical capabilities at the various levels of the collaboration protocol and put the matching results in the CPA document
- One trading partner registers a CPA template document based on the BPSS document and the technical parameters he can support. In this "almost complete" CPA, some items must be completed or negotiated. Other items (such as endpoint address) should be provided by another party

PowerDesigner supports the generation of both BPSS documents and a CPA template. The generated CPA document cannot be directly registered; you must open it in a text editor and modify manually the parts that are in commentary.

Since a CPA document is based on a BPSS document, the two documents share some parameters such as roles, business transaction activities, or business transactions characteristics (business actions, business documents and messages ordering, times to acknowledge receipt, times to acknowledge acceptance). However, CPA values often override BPSS values. PowerDesigner generates both BPSS and CPA documents from the same BPM, so the shared items have the same value. In order to save BPSS and CPA values separately, you should:

- Design a BPM with the ebXML BPSS 1.04 process language to generate a BPSS document
- Generate another BPM from the first one by using the Save As function or the model-to-model generation function, and use this new model to define CPA-specific values and generate the CPA document template.

Designing a Partner's Identification

In ebXML CPA, a trading partner is designated as a party to the agreement, and identified with a party Id. The generated CPA template always includes two parties. Each party can play more than one role defined by the BPSS document on the Binary Collaborations. To define the characteristics of the two parties and some shared parameters (CPA Id, Start and End time, Status, conversation constraints), you can use the following extended attributes available from the model property sheet:

Name	Internal code	Description
Identifier	PartyInfo1ID, PartyInfo2ID	Identifier of the party
Name	PartyInfo1Name, PartyInfo2Name	Name of the party
Reference to more details	PartyRef1, PartyRef2	Reference to more detailed information on the party
Default delivery channel	PartyInfo1defaultMshChannelId, PartyInfo2defaultMshChannelId	Default Delivery channel used by the party to exchange the messages
Default message packaging	PartyInfo1defaultMshPackageId, PartyInfo2defaultMshPackageId	Default message Packaging used by the party
CPA ID	cpaid	An ID of the CPA document
BPSS document reference	hrefBPSS	Reference to the BPSS document
Start time, End time	Start, End	The Start and End time define how long the agreement is valid
Status	Status	Defines the status of agreement. By default it is proposed. It can be agreed or signed
Invocation limit, Concurrent conversations	ConversationConstraints_invocationLimit, ConversationConstraints_concurrentConversations	Place limits on the number of conversations under the CPA

Designing CanSend/CanReceive Actions

Flow objects define Business Documents and signals exchanged during the Business Transaction. Consequently each flow can be generated as CanSend or CanReceive element depending on whether the party is the Requester or the Receiver of the message.

The following extended attributes (accessible from the Extended Attributes tab of the flow property sheet) apply to the flow. They represent mainly three technical parameters: actions,

delivery channels and the message packaging used by the sender and the receiver. Since these parameters can be different in the sender and the receiver side, PowerDesigner associates two extended attributes to each parameter.

Extended attribute	Description
ReceiverActionID, SenderActionID	ID of the Receive/Send action
ReceiverActionName, SenderActionName	Name of the Receive/Send action
ReceiverChannelID, SenderChannelID	Delivery channel used by the Receiver/Sender to receive/send the message. The value references the name of a resource object having <<Delivery-Channel>> stereotype
ReceiverPackageID, SenderPackageID	Id of the Message packaging used by the receiver/sender. This packaging is not defined by a resource object but will be generated as Packaging template. Receiver and Sender often use the same Packaging
isAuthenticated, isConfidential, isTamperProof	Attributes defined only on the flows representing an acknowledgement

The Action identification parameters Sender/ Receiver Action ID and Name are not required. Indeed PowerDesigner generates a default ID by combining the name of the Requesting or Responding business activity and a default string.

This default string can be:

- Send or Receive for the business document (Purchase Order Request Action_Send_ID)
- ReceiptAck or AcceptanceAck for business signals (Purchase Order Request Action_Receive_ReceiptAck_ID)

The Sender/Receiver message packaging ID is the string that represents the reference to one packaging element in the CPA. PowerDesigner uses this ID to generate a default packaging template.

The Sender/Receiver channel ID references a detailed delivery channel that should have been defined by a resource object having <<Delivery Channel>> as stereotype.

Designing a Delivery Channel

You design a Delivery Channel using a resource with a <<DeliveryChannel>> stereotype.

The name of this object constitutes the Id of the Delivery Channel element. If at least, one CanSend/CanReceive action references the Id of the Delivery Channel, then the generation function produces a corresponding Delivery Channel in the CPA template.

The following extended attributes (accessible from the Extended Attributes tab of the resource property sheet) apply to the Delivery Channel:

Name	Internal code	Description
Document exchange	DocExchange	References a DocExchange element
Transport	Transport	References a Transport element that should be defined by using a Resource object having <<Transport>> stereotype
Synchronization reply mode	MsgCharacteristicsSyncReplyMode	Possible values are: mshSignalsOnly, responseOnly, signalsAndResponse, signalsOnly, or none. With a value different from "None", it indicates what the sending application expects in a synchronous response. This parameter impacts the generation of the sequence of actions in the CPA
Requests acknowledgement	MsgCharacteristicsAckRequested	Possible values are: always, never, or perMessage
Eliminates duplicate	MsgCharacteristicsDuplicateElimination	Possible values are: always, never, or perMessage
Requests acknowledgement signature	MsgCharacteristicsAckSignatureRequested	Possible values are: always, never, or perMessage
Actor	actor	Possible values are: urn:oasis:names:tc:ebxml-msg:actor:nextMSH or urn:oasis:names:tc:ebxml-msg:actor:toPartyMSH

The extended attribute `MsgCharacteristicsSyncReplyMode` on the resource object with <<DeliveryChannel>> stereotype impacts the sequence of actions in the CPA.

With a value different from "None", PowerDesigner generates:

- A `CanReceive` element (Receive action) for the party receiving a business document
- A nested `CanSend` elements (Send action) to express that the requests and responses are synchronized

The nested `CanSend` elements can represent:

- Send signal actions (`signalsOnly`)
- Send response document actions (`responseOnly`)
- Send signal actions and Send response document actions (`signalsAndResponse`)

If the `MsgCharacteristicsSyncReplyMode` attribute is `None`, the `CanSend` actions using the Delivery Channel are not nested in the `CanReceive` element.

With a value different from `None`, the Sender and the Receiver must use the same channel Id (`SenderChannelId` and `ReceiverChannelID` attributes on the flow must reference the same resource with a <<DeliveryChannel>> stereotype).

Designing a Transport Element

You design a Transport element using a resource with a <<Transport>> stereotype.

The name of this object is the Id of the Transport element. If the name of the resource object with a <<Transport>> stereotype is referenced by a Delivery Channel object, then the generation function produces a corresponding Transport element in the CPA template.

The following extended attributes (accessible from the Extended Attributes tab of the resource property sheet) apply to the Transport element. These attributes allow you to define technical parameters related to the Transport element. PowerDesigner generates CPA Transport element when a <<DeliveryChannel>> resource object references the corresponding resource objects:

Name	Internal code	Description
Access authentication	SenderTransport_AccessAuthentication ReceiverTransport_AccessAuthentication	Values: basic (default) or digest. Sender Transport Access Authentication Receiver Transport Access Authentication
Endpoint	ReceiverTransport_Endpoint	ReceiverTransport Endpoint (URI address)
Endpoint type	ReceiverTransport_Endpoint_type	Values: allPurpose (default), login, request, response, or error. Type of the ReceiverTransport Endpoint
Protocol	SenderTransport_Protocol ReceiverTransport_Protocol	Value: HTTP. SenderTransport_Protocol Receiver Transport Protocol
Protocol version	SenderTransport_Protocol_Version ReceiverTransport_Protocol_Version	Value: 1.1. SenderTransport_Protocol_Version Version of the ReceiverTransport Protocol
Security protocol	SenderTransport_TransportClientSecurity_TransportSecurityProtocol ReceiverTransport_TransportServerSecurity_TransportSecurityProtocol	Value: SSL. SenderTransport_TransportClientSecurity_TransportSecurityProtocol ReceiverTransport_TransportServerSecurity_TransportSecurityProtocol

Name	Internal code	Description
Security protocol version	SenderTransport_TransportClientSecurity_TransportSecurityProtocol_Version ReceiverTransport_TransportServerSecurity_TransportSecurityProtocol_Version	Value: 3.0. SenderTransport_TransportClientSecurity_TransportSecurityProtocol_Version ReceiverTransport_TransportServerSecurity_TransportSecurityProtocol_Version
Client certificate reference	SenderTransport_TransportClientSecurity_ClientCertificateRef	SenderTransport_TransportClientSecurity_ClientCertificateRef
Server certificate reference	ReceiverTransport_TransportServerSecurity_ServerCertificateRef	ReceiverTransport_TransportServerSecurity_ServerCertificateRef
Server security details reference	SenderTransport_TransportClientSecurity_ServerSecurityDetailsRef	SenderTransport_TransportClientSecurity_ServerSecurityDetailsRef
Client security details reference	ReceiverTransport_TransportServerSecurity_ClientSecurityDetailsRef	Receiver Transport TransportServerSecurity ClientSecurityDetailsRef

Unsupported Concepts

Some of the concepts of the ebXML collaboration protocol agreement are not supported. However you can generate a CPA template that includes for each CPA element, a default XML code that you can modify to meet your needs.

The non-supported elements are:

- DocExchange
- SecurityDetails
- Packagings
- Certificates
- SimpleParts
- Signature

All DocExchange and Packaging elements referenced by CanSend/CanReceive elements and deliveryChannel are generated. The generated XML code constitutes a sample code that you should modify.

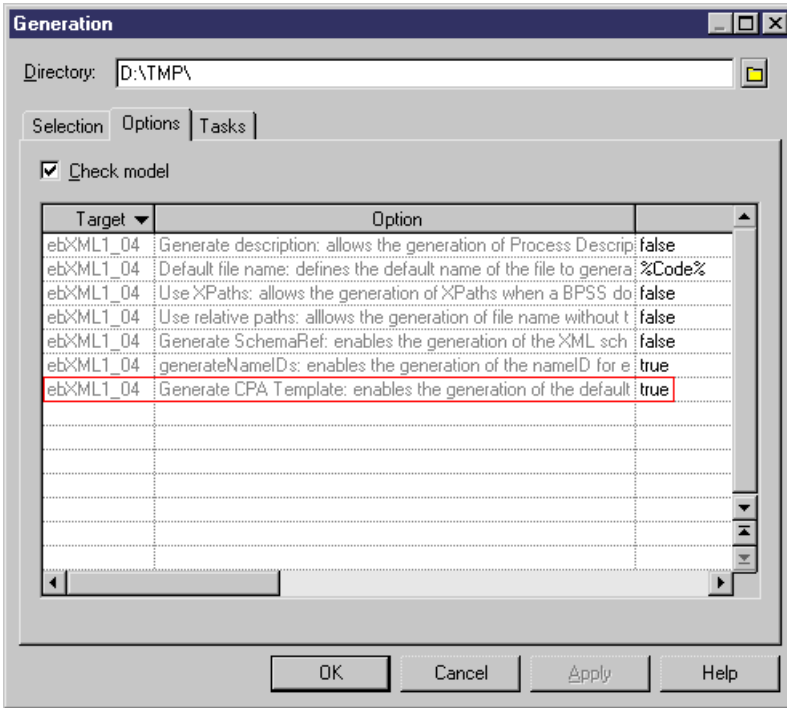
For the remaining elements, PowerDesigner generates a sample code brought from the ebXML CPA-CPP specification.

Generating for EbXML CPA

Generating a CPA template document is an option in ebXML BPSS generation.

You generate a CPA template file the same way as you generate an ebXML BPSS file.

Before you start the generation process, you must set the Generate CPA Template option to true in the Options tab of the Generation dialog box.



For more information on generation options, see *Selecting ebXML generation options* on page 268.

Index

A

- activity 195
- Add Compensation Handler 240
- Add Default Exception Handler 240
- Add Exception Handler 240
- Add Timeout Handler 240
- Analysis (process language) 169
- arrange process in process hierarchy diagram 18
- Assign activity 238
- atomic process (decomposed process) 45

B

- Binary Collaboration
 - ebXML language 261
 - generate orchestration BPM 175
- Binary Collaboration (ebXML BPSS) 252
- BP4WS
 - build message 216
 - create environment 210
 - generate 223
 - generation 169
 - introduction 209
 - object modeling 211
 - object properties 221
 - process language 169
 - reverse engineering 224
 - top-level process 217
 - workflow 210
- BPM
 - analysis 3
 - BPMN 2.0 process language 183
 - business process diagram 21
 - changing 12
 - check model 149
 - collaborative 3
 - create 5
 - data flow diagram 199
 - edit definition file 11
 - orchestration 3
 - process hierarchy diagram 15
 - process service diagram 127
 - simulate 134
- BPM objects
 - convert to SIMUL8 objects 131

- BPMN 2.0 183
 - activity 195
 - choreography diagram 184, 186
 - choreography task 186, 190
 - collaboration diagram 186
 - conversation diagram 183
 - conversation link 197
 - conversation node 186, 189
 - correlation key 197
 - correlation property 197
 - data 196
 - data association 197
 - event 191
 - gateway 194
 - message 197
 - message flow 197
 - participant 189
 - sequence flow 197
 - task 195
- BPMN process language 169
- browse for WSDL on UDDI 104
- Business Collaboration (ebXML BPSS) 252
- Business Document (ebXML BPSS) 252
- business process
 - designed for Sybase WorkSpace Business Process 231
 - variables 232
- business process diagram
 - choreography diagram 23
 - data 83
 - data flow diagram 32
 - message format 78
 - organization unit 52
 - process 32
 - resource 93
 - resource flow 95
 - service interface 110
 - start 62, 63, 65
 - top-level diagram 22
- business process model 3
- Business Transaction
 - ebXML language 256
 - generate orchestration BPM 175
- Business Transaction (ebXML BPSS) 252

Index

C

change target (Analysis to DFD) 176

check model 149

 choreography task 165

 communication link 167

 conversation node 166

 correlation key 163

 data 158

 data transformation 162

 decision 151

 end 157

 event 164

 flow 153

 message format 157

 operation 160

 organization unit 155

 package 149

 process 150

 resource 154

 resource flow 154

 service interface 160

 service provider 159

 start 156

 synchronization 152

 variable 161

choreography 24

choreography (ebXML language) 261

choreography diagram 23, 184

 link to conversation node 186

choreography task 190

 check model 165

 link to conversation node 186

code

 preview 8

collaboration diagram 186

collaborative BPM

 define 251

 ebXML BPSS 251

collapse process hierarchy 18

committee process 44, 55

communication link

 check model 167

Compensation (event) 75

compensation handling 241

complete process hierarchy 18

complex activity 240

component

 export in BPM 106

 import in BPM 106

 imported as service provider 107

composite process (collaborative BPM) 255

composite view 49

conditional branch 65

context variable 232

control flow 19

conversation diagram 183

conversation link 197

conversation node 189

 check model 166

 link to choreography diagram 186

 link to choreography task 186

convert to decomposed 47

correlation 243

correlation key 197

 check model 163

 create 122

 define 121

 properties 122

 variable 121

correlation property 197

Create New Operation Wizard 112

CRUD matrix 50

D

data 30, 83, 196

 analyze 25

 check model 158

 create from the list 84

 CRUD matrix 50

 data CRUD 25

 data flow 25

 export to other models 85, 88

 flow 25

 generate orchestration BPM 174

 import from other models 85, 89

 link to other model objects 85, 86

 message format 25

 migrate to process 92

 properties 84

 select for a flow 91

 select for a message format 91

 select for a resource flow 91

 sub-data 84

data association 197

Data CRUD matrix 50

data flow 199

data flow diagram

 balancing 205

- check model 199
 - concepts 199
 - create 201
 - data store 202
 - design 201
 - external entity 202
 - flow 201
 - process 201
 - split/merge 202
 - data store
 - data flow diagram 202
 - number ID 203
 - data transformation
 - assigned variable 123
 - check model 162
 - create 123
 - define 123
 - input variable 123
 - properties 123
 - Transformation tab 125
 - database web service
 - imported as service provider 107
 - decision 30, 65
 - check model 151
 - conditional branch 65
 - create 67
 - merge 65
 - properties 67
 - decompose
 - atomic process 45
 - decomposed process 44, 47
 - build default flows 19
 - close 49
 - committee process 44, 55
 - composite view 49
 - create from selection of symbols 46
 - expanded view 44
 - go up one level 49
 - open 49
 - open sub-diagram 49
 - decomposed view (process) 49
 - decomposition link 16
 - delay activity 242
 - diagram
 - business process 21
 - choreography diagram 21
 - data flow 199
 - data flow diagram 21
 - process hierarchy 15
 - process service 127
 - simulation properties 144
 - top-level (collaborative BPM) 255
 - top-level diagram 21
 - disable swimlane mode 52
 - display preferences 11
 - Document Envelope (ebXML language) 260
 - drag and drop process in hierarchy diagram 18
- ## E
- ebXML
 - generation 169
 - generation options 268
 - ebXML BPSS
 - Binary Collaboration 252
 - Business Collaboration 252
 - Business Document 252
 - Business Transaction 252
 - Choreography 252
 - collaborative BPM 251
 - generate 267
 - key concepts 252
 - MultiParty Collaboration 252
 - reverse engineering 269
 - ebXML CPA 269
 - design CanReceive action 271
 - design CanSend action 271
 - design Delivery Channel 272
 - design partner's identification 271
 - design Transport 274
 - generate 275
 - unsupported concepts 275
 - EJB Web services
 - imported as service provider 230
 - enable swimlane mode 52
 - end 30
 - check model 157
 - create 64
 - define 63
 - properties 64
 - simulation properties 142
 - event 191
 - check model 164
 - Compensation 75
 - create 76
 - define 75
 - event handler 77
 - Fault 75
 - properties 76

Index

- Timer 75
- event activity 241
- event activity (exception handling) 241
- event handler 77
- exception handling 241
- expand process hierarchy 18
- export
 - component in BPM 106
 - data to other models 85, 88
 - service provider from BPM 108
 - SIMUL8 model 134
- extension 12
- extension file 12
- external entity
 - data flow diagram 202

F

- Fault event 75
- flow 30, 70
 - check model 153
 - create 71
 - data flow diagram 201
 - ebXML language 265
 - generate orchestration BPM 174
 - properties 71
 - select data 91
 - simulation properties 143
- fork 68

G

- Gane & Sarson symbols 199
- gateway 194
- generate
 - BPEL4WS 169
 - BPM to BPM 173
 - ebXML 169
 - ebXML BPSS 267
 - ebXML CPA 275
 - ebXML options 268
 - options for Sybase WorkSpace Business Process 249
 - orchestration BPM from analysis BPM 173
 - orchestration BPM from collaborative BPM 173
 - orchestration BPM from orchestration BPM 173
 - path for Sybase WorkSpace Business Process 249

- process language 169
- SIMUL8 model 134
- Sybase WorkSpace Business Process 169, 248
 - tasks for Sybase WorkSpace Business Process 249
- WS-BPEL 169
- guard condition for decision 65

H

- hierarchy
 - collapse 18
 - create with Process tool 17
 - root process 17
 - sibling process 17
 - sub-process 17

I

- implementation
 - Execute operation 40
- implementation type 37
- import
 - component in BPM 106, 107
 - data from other models 85, 89
 - database web service in BPM 106, 107
 - EJB or Java Web services 230
 - existing services use case 229
 - SIMUL8 model 137
 - Visio 179
 - web service 106
 - WorkSpace services 247
 - WSDL 103
- increment version number of the generated files 249
- initiating role 73
- input flow
 - fork 68
 - join 68
- invoke
 - database 248
 - Java 248
 - message 248
 - service 248
 - transformation 248

J

Java Web services
 imported as service provider 230
 join 68

L

link
 data to other model objects 85, 86
 loop
 activity 241
 condition 241

M

merge 65
 services 247
 message 197
 message flow 197
 message format 30
 check model 157
 create 79
 create message part 82
 define 78
 generate orchestration BPM 174
 properties 79
 select data 91
 message part 30
 create 82
 define 81
 properties 82
 migrate data to process 92
 model
 create 5
 model options 10
 new 5
 preview code 8
 properties 7
 model options 10
 modeling environment
 customize 10
 move process in process hierarchy diagram 18
 MultiParty Collaboration
 ebXML language 266
 generate orchestration BPM 175
 MultiParty Collaboration (ebXML BPSS) 252

N

new
 BPM 5

model 5

O

One-way service invocation 235
 operation
 check model 160
 copy 111
 create 112
 create from wizard 112
 define 111
 designed for Sybase WorkSpace Business
 Process 231
 move 111
 properties 115
 orchestration BPM
 generate from analysis BPM 174
 generate from collaborative BPM 175
 generate from orchestration BPM 175
 organization unit 23, 30, 52–55
 attached to process 55
 check model 155
 committee process 55
 creating 53
 generate orchestration BPM 175
 icon in a collaborative BPM 255
 parent organization 54
 properties 54
 swimlane 52
 switching to actor or swimlane 61
 See also swimlane
 Organization Unit Swimlane tool 54
 output flow
 fork 68
 join 68

P

package
 check model 149
 participant 189
 partner link 233
 port type
 designed for Sybase WorkSpace Business
 Process 231
 preview
 WSDL 103
 preview code 8

Index

- process 23, 30
 - atomic 32
 - attached to organization unit 55
 - BPM 32
 - build default flow between processes 19
 - business process diagram 32
 - check model 150
 - choreography 24
 - committee process 55
 - create 34
 - data flow diagram 32, 201
 - decomposed 32, 44
 - decomposed view 44, 49
 - decomposition link 16
 - drag and drop into decomposed process 44
 - generate orchestration BPM 174
 - implementation 29
 - implementation type 37, 40
 - move with drag and drop 18
 - number ID 203
 - process hierarchy diagram 16, 32
 - properties 34, 44
 - simulation properties 138
 - tool 17
 - top-level (collaborative BPM) 255
- process diagram
 - convert to decomposed process 47
- process hierarchy
 - complete 18
 - expand 18
 - horizontal display 16
 - root process 16
 - sibling process 16
 - sub-process 16
 - vertical display 16
- process hierarchy diagram 15
 - arrange process 18
 - move process 18
 - reuse process 19
- process language
 - BPEL4WS 169
 - BPMN 169
 - BPMN 2.0 183
 - Data Flow Diagram 199
 - generate 169
 - Service Oriented Architecture 169, 207
 - WS-BPEL 169
- process service diagram 127
 - service interface 110

R

- receive activity (Sybase WorkSpace Business Process) 236
- Remove Composite Level 48
- Request/Reply service invocation 235
- Requesting Business Activity (ebXML language) 257
- resource 30
 - check model 154
 - create 94
 - CRUD matrix 50
 - define 93
 - properties 94
 - simulation properties 140
- Resource CRUD matrix 50
- resource flow 30
 - check model 154
 - create 96
 - define 95
 - properties 97
 - select data 91
 - simulation properties 140
- Responding Business Activity (ebXML language) 257
- responding role 73
- reuse process in process hierarchy diagram 19
- reverse engineering 171
 - BPEL4WS 224
 - into new BPM 171
 - WS-BPEL 224
- reverse engineering into existing BPM 172
- role association 23, 30
 - create 74
 - define 73
 - initiating role 73
 - properties 74
 - responding role 73
 - unavailable in Swimlane Mode 73
- role in swimlanes 24

S

- scope (decomposed process) 46
- Send activity 237
- Send Fault activity 237
- sequence flow 197, 243
- service
 - designed for Sybase WorkSpace Business Process 231

- invoke 248
 - invoke WorkSpace service 248
- service interaction 234
- service interface
 - check model 160
 - create 110
 - define 110
 - properties 111
- Service Oriented Architecture process language
 - 169, 207
- service process diagram
 - operation 111
- service provider
 - check model 159
 - create 100
 - create components 108
 - created from components 107
 - created from database web service 107
 - define (business process diagram) 98
 - define (process service diagram) 98
 - exported as component 108
 - import EJB or Java Web services 230
 - process service diagram 127
 - properties 100
- Service Provider Export Wizard 108
- Service Provider Import Wizard 107
- services
 - import 247
 - merge 247
- SIMUL8
 - generate model 134
 - import file into a new BPM 137
 - import model into an existing BPM 137
 - support 131
- SIMUL8 objects
 - convert to BPM objects 131
- simulation
 - analyze results 135
 - default properties 134
 - define 129
 - define properties 138
 - fine-tune 135
 - prepare your BPM 131
- Split-Join activity 240
- split/merge
 - data flow diagram 202
- start 30, 62
 - check model 156
 - collaborative BPM 255
 - create 62
 - generate orchestration BPM 174
 - properties 63
 - simulation properties 141
- sub-data 84, 85, 89
- swimlane 53
 - changing format 61
 - changing orientation 60
 - choreography diagram 24
 - copying and pasting 57
 - creating 54
 - creating links between pools 60
 - grouping 58
 - moving 57
 - not allowed in top level diagram 255
 - organization unit 52
 - resizing 61
 - responsibility 24
 - role 24
 - selecting symbol 56
 - ungrouping 58
 - See also organization unit
 - See also organization unit
- Sybase WorkSpace Business Process
 - Assign activity 238
 - Assignments tab 238
 - bpmn.gem 249
 - check model 227
 - compensation handling 241
 - complex activity 240
 - concepts 227
 - context variable 232
 - correlations 243
 - delay activity 242
 - design business process 231
 - design service 231
 - event activity 241
 - exception handling 241
 - generate 248
 - generated files 249
 - generation 169
 - generation options 249
 - generation path 249
 - generation task 249
 - import BPEL use case 230
 - import EJB or Java Web services 230
 - import existing services use case 229
 - import WorkSpace services 247
 - invoke 234

Index

- invoke Workspace service 248
- loop activity 241
- model.bpmn 249
- One-way service invocation 235
- partner link 233
- receive activity 236
- Request/Reply service invocation 235
- Send activity 237
- sequence flow 243
- service interaction 234
- Split-Join activity 240
- svc_bpmn 249
- svc_soap 249
- switch to Sybase Workspace Business Process language 247
- terminate activity 242
- ThisService 236
- timeout handling 241
- top-down use case 228
- variables 232
- XSD 249
- XSD data types 232, 233
- synchronization 30, 68
 - change to horizontal 69
 - change to vertical 69
 - check model 152
 - create 69
 - properties 69

T

- task 195
- terminate activity 242
- ThisService 236
- timeout handling 241
- Timer event 75
- timeToPerform in ebXML language 265
- top-level diagram 22
 - collaborative BPM 255
 - not allowed for swimlane 255
 - objects 23
- top-level process
 - BPEL4WS 217
 - collaborative BPM 255
 - generate orchestration BPM 174
 - WS-BPEL 217
- traceability link 13

U

- UDDI
 - operator URL 104

- version 104

V

- variable
 - check model 161
 - code uniqueness 119
 - create 120
 - define 119
 - move 119
 - name uniqueness 119
 - properties 120
- Visio
 - import 179

W

- web service
 - imported in BPM 106
- Workspace services 247
- WS-BPEL
 - build message 216
 - create environment 210
 - generate 223
 - generation 169
 - introduction 209
 - object modeling 211
 - object properties 218
 - reverse engineering 224
 - top-level process 217
 - workflow 210
- WS-BPEL process language 169
- WSDL
 - import 103
 - reverse engineering 103
- WSDL URL 103

X

- xem 12
- XML model attached to XSD document 119
- XS8
 - import file 137
 - import SIMUL8 file 137
 - SIMUL8 file 134, 137
- XSD data type 232, 233
- XSD document
 - attach XML model 119
 - create 117

define 117
properties 118

Y

Yourdon symbols 199

