SAP® PowerDesigner®
Document Version: 16.5 SP05 – 2015-03-24

Business Process Modeling

Content

1 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). SAP® PowerDesigner® supports Analysis, SOA, DFD, SAP® Solution Manager, BPMN (including for SAP BPM), and BPEL process languages, and process simulation through SIMUL8.

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

- Analysis - An implementation-neutral notation to decompose and analyze 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 (see Analysis BPM [page 18]).
- Service Oriented Architecture (SOA) - An implementation-neutral notation that adds information about events and the implementation of services by processes (see Service Oriented Architecture BPM (SOA) [page 70]).
- Data Flow Diagram (DFD) - Analyzes your system with respect to the exchange of data between processes, data stores, and external entities (see Data Flow Diagram (DFD) [page 103]).
- SAP Solution Manager - Manages and monitors SAP Business Suite implementations and associated systems. PowerDesigner supports scenario, scenario flow, and business process diagrams (see SAP Solution Manager [page 107]).
- BPMN 2.0 - A standard graphical notation to represent the control flow of a business process, suitable for refining the analysis of a system with respect to standards. PowerDesigner supports conversation, choreography, collaboration, and process diagrams (see BPMN 2.0 Descriptive [page 120] and BPMN 2.0 Executable [page 134]).
- BPEL4WS 1.1 or WS-BPEL 2.0 - Defines the invocation of services by processes (see BPEL4WS 1.1 and WS-BPEL 2.0 [page 158]).

1.1 Creating a BPM

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

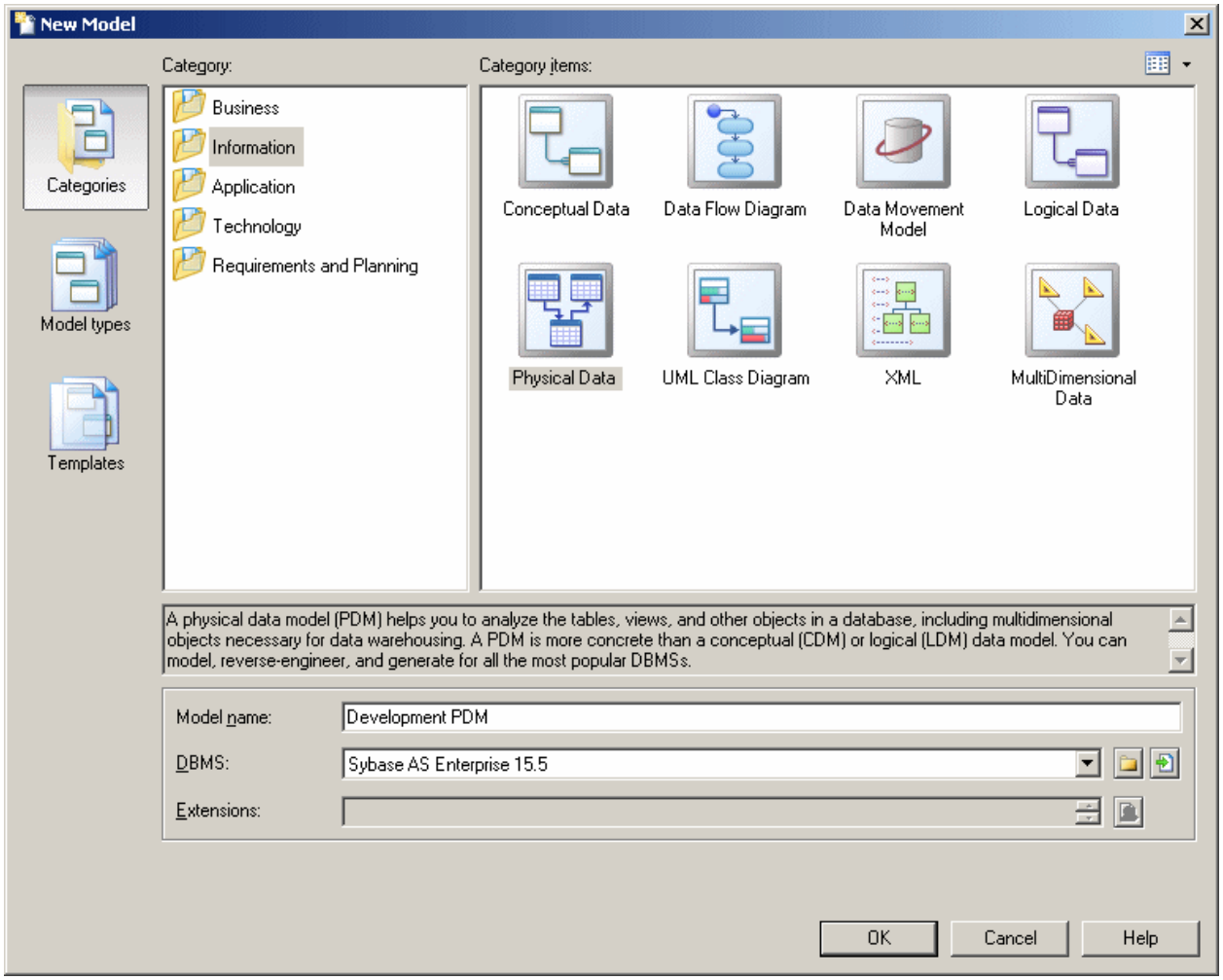
Context

> **Note**
>
> In addition to creating a BPM from scratch with the following procedure, you can also:
>
> - Reverse-engineer existing BPMN code (see Importing and Exporting BPMN 2.0 Files [page 154]) or BPEL code (see Reverse Engineering BPEL Languages [page 176]).

- Import a SIMUL 8 file (see Simulating a Business Process Model with SIMUL8 [page 178]).

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.



Procedure

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.

1.2 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.

Context

> **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:

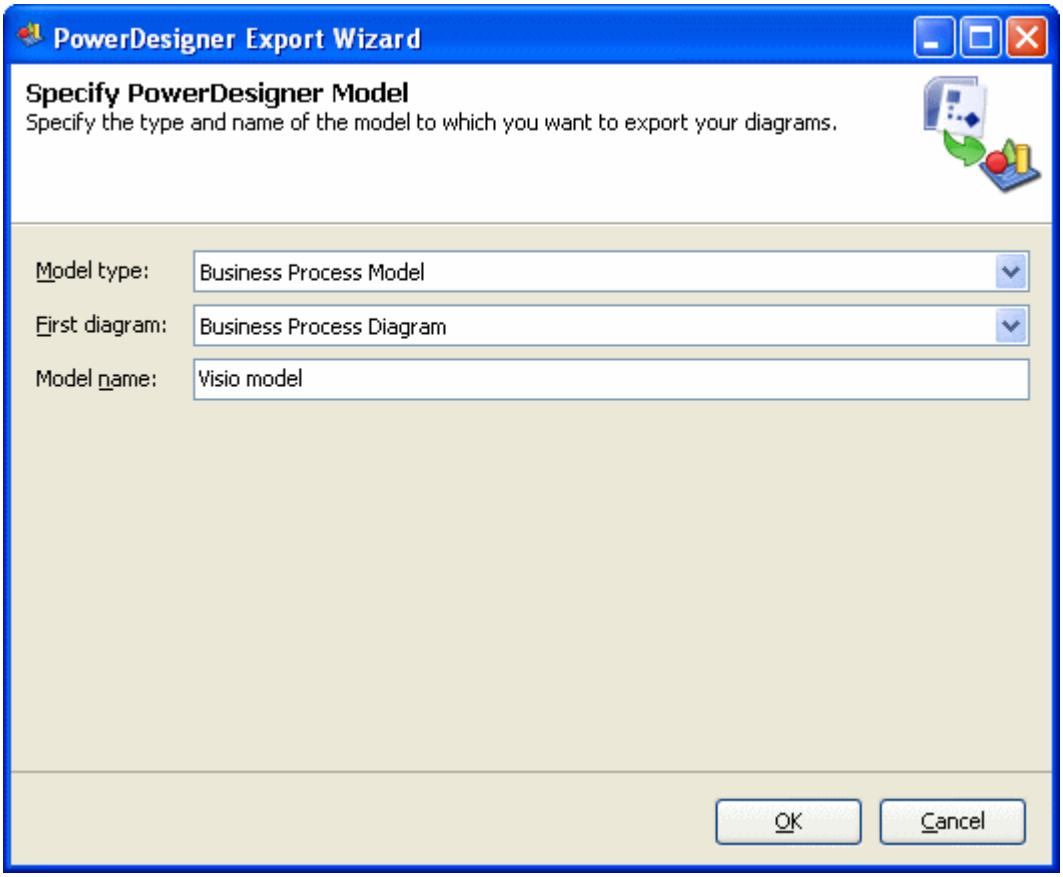
Table 1:

Visio Template	PowerDesigner Diagram
Audit Diagram	BPM Analysis/ Business Process Diagram
Basic Flowchart	BPM Analysis/ Business Process Diagram

Visio Template	PowerDesigner 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

Procedure

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.

1.3 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.

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

Table 2:

Tools	Description
	*Editor Menu [Shift+F11]* - Contains the following commands: • *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 occurence of the searched for text. • *Find Previous... [Shift+F3]* - Finds the previous occurence 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.
	*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*.
	*Save [Ctrl+S]* - Saves the content of the field to the specified file.
	*Print [Ctrl+P]* - Prints the content of the field.
	*Find [Ctrl+F]* - Opens a dialog to search for text.
	*Cut [Ctrl+X]*, *Copy [Ctrl+C]*, and *Paste [Ctrl+V]* - Perform the standard clipboard actions.
	*Undo [Ctrl+Z]* and *Redo [Ctrl+Y]* - Move backward or forward through edits.
	*Refresh [F5]* - Refreshes the Preview tab. 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.
	*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 *Customizing and Extending PowerDesigner > Extension Files > Generated Files (Profile) > Generating Your Files in a Standard or Extended Generation*.
	*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.

1.4 Customizing your Modeling Environment

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

1.4.1 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:

Table 3:

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 *Requirements Modeling*).
External Shortcut 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.
Default Message Format	Specifies the default setting for the *Message Format* property for flows and resource flows. You can choose: • 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* > *Modeling with PowerDesigner* > *Objects* > *Naming Conventions*.

1.4.2 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*.

In the *Display Preferences* dialog, select the type of object in the list in the left pane, and modify its appearance in the right pane.

You can control what properties it will display on the *Content* tab, and how it will look on the *Format* tab. If the properties that you want to display are not available for selection on the *Content* tab, click the *Advanced* button and add them using the *Customize Content* dialog.

For detailed information about controlling the appearance and content of object symbols, see *Core Features Guide* > *Modeling with PowerDesigner* > *Diagrams, Matrices, and Symbols* > *Display Preferences*.

1.4.3 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.

> **Caution**
>
> The resource files provided with PowerDesigner inside the `Program Files` folder cannot be modified directly. To create a copy for editing, use the *New* tool on the resource file list, and save it in another location. To include resource files from different locations for use in your models, use the *Path* tool on the resource file list.

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. We will perform all possible validation checks, but we do not maintain specific environments to fully certify these resource files. We will support them by accepting bug reports and providing fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. You are invited to assist us by testing fixes and reporting any continuing inconsistencies.

1.4.3.1 Changing the Process Language

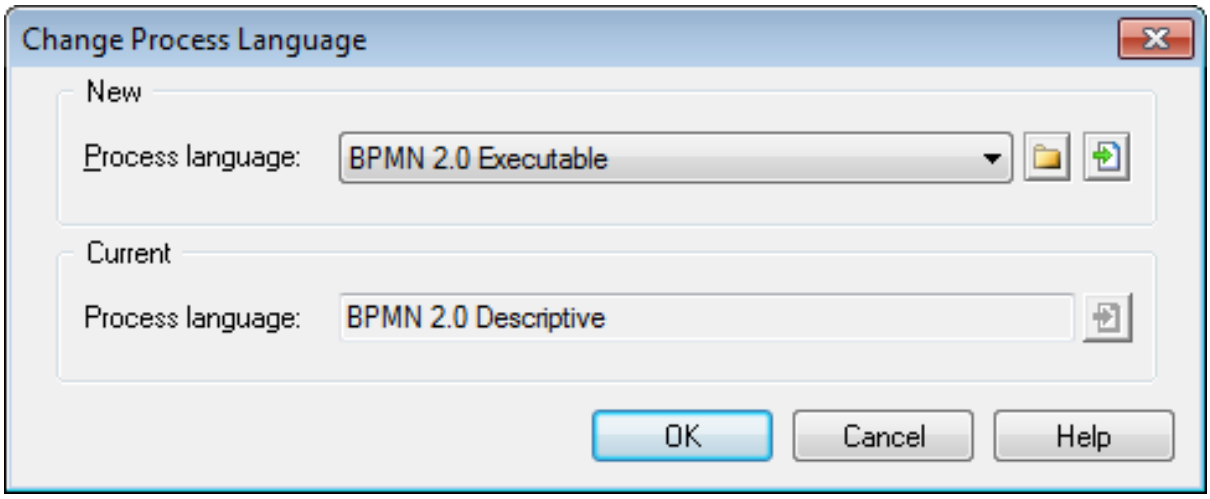
You can change the *process language* being modeled in your BPM at any time.

Context

> **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.

Procedure

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*.

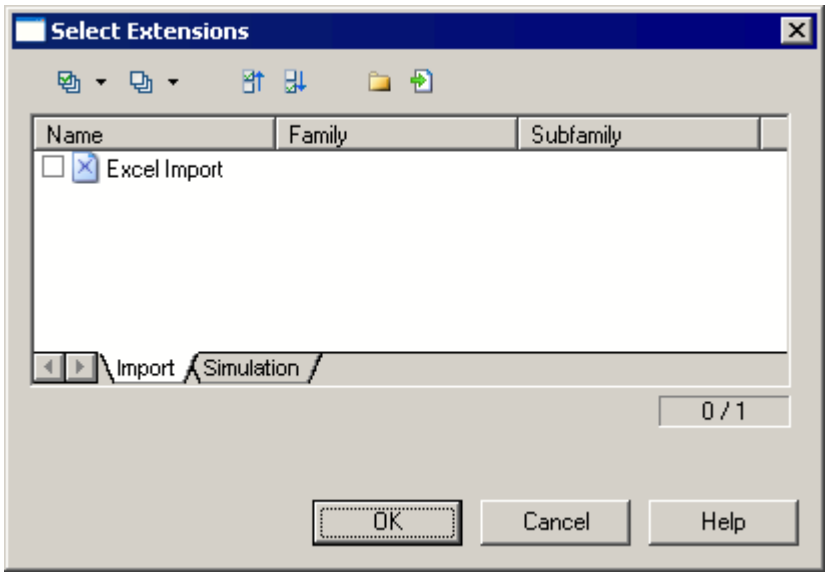
   A message box opens to tell you that the process language has been changed.
4. Click *OK* to return to the model.

1.4.4 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 extensions 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 *Attach 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 quickly add a property or collection to an object from its property sheet, click the menu button in the bottom-left corner (or press F11) and select *New Attribute* or *New List of Associated Objects*. For more information, see *Core Features Guide* > *Modeling with PowerDesigner* > *Objects* > *Extending Objects*.

To create a new extension file and define extensions in the Resource Editor, select *Model* > *Extensions*, click *Add a Row*, and then click *Properties*. For detailed information about working with extensions, see *Customizing and Extending PowerDesigner* > *Extension Files*.

1.4.5 Traceability Links

Traceability links provide a flexible means for creating a connection between any object in any type of model and any other object in the same model or any other model of any type. Traceability links have no formal semantic meaning, but can be followed when performing an impact analysis or otherwise navigating through the model structure.

To create a traceability link between objects in the same diagram, select the *Link/Traceability Link* tool in the Toolbox. Click inside the symbol of the object that is dependent and, while continuing to hold down the mouse button, drag the cursor and release it on the symbol of the object on which it depends.

In the following example, the **Work** entity is shown as being dependent on **School** through a traceability link:

To create a traceability link to any object in any model that is open in the Workspace, open the property sheet of the dependent object, click its *Traceability Links* tab, and click the *Add Objects* tool. Use the *Model* list to select a different model, select the object to point to and click *OK* to create the link and return to the dependent object's *Traceability Links* tab.

You can optionally specify a type for any traceability link in the *Link Type* column.

Click the *Types and Grouping* tool to perform various actions on this tab:

- To make a link type available for selection in the *Link Type* column, click the *Types and Grouping* tool and select *New Link Type*. Enter a *Name* for the link type and, optionally, a *Comment* to explain its purpose, and then click *OK*.

  > **Note**
  >
  > Traceability link types created in this way are stored as stereotypes in an extension file embedded in the model. To work directly with this file click the *Types and Grouping* tool and select *Manage Extensions*. For detailed information about working with these files, see *Customizing and Extending PowerDesigner > Extension Files*.

- To control the display and grouping of links, click the *Types and Grouping* tool and select:
  - *No Grouping* - to display all the links in a single list.
  - *Group by Object Type* - to display links to different types of objects on separate sub-tabs. To add a link to a new object type, click the plus sign on the leftmost sub-tab.
  - *Group by Link Type* - to display different link types on separate sub-tabs. To add a new link type, click the plus sign on the leftmost sub-tab.

> **Note**
>
> To see all of the objects that point to an object via traceability links, open its property sheet, click its *Dependencies* tab, and click the *Incoming Traceability Links* sub-tab.

2 Analysis BPM

The Analysis language is the base language of the PowerDesigner BPM, an implementation neutral notation for analyzing your systems and decomposing your processes to any level of detail.

Having developed your model, you can select *Tools* > *Generate Business Process Model* to generate a model targeting another process language.

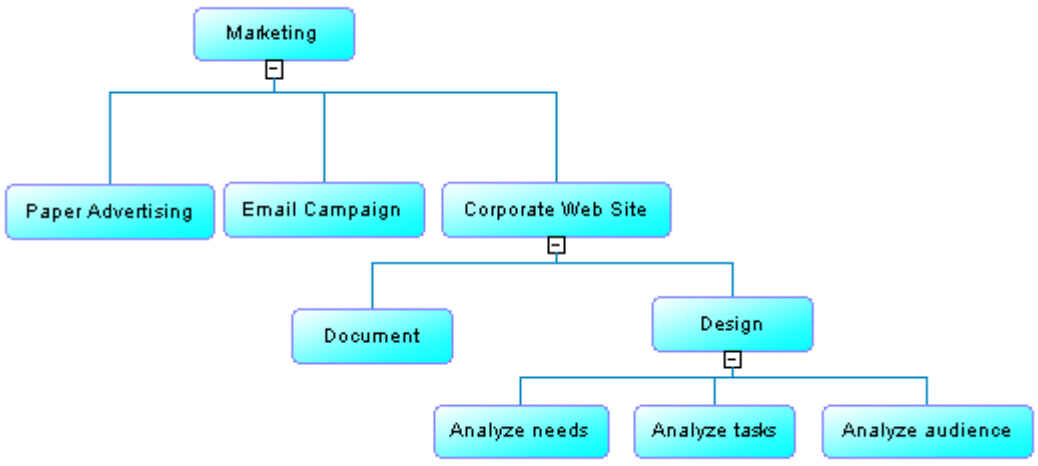
2.1 Process Hierarchy Diagrams (Analysis)

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*.

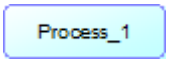
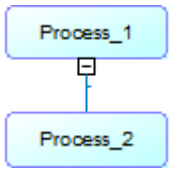
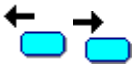
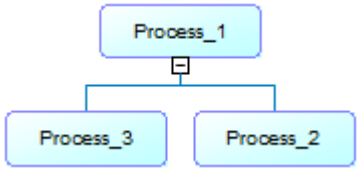
The PHD is commonly used during the analysis phase of a project to identify all the processes in a system by name, and decompose them into multiple levels of sub-processes.

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:



You can create processes with the *Process* tool from the diagram Toolbox:

Table 4:

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

You can modify your hierarchy in the following ways:

- Expand and collapse children under a process by clicking the + or - sign on the base of the symbol. Alternatively, right-click a process and select *Expand* to show one level of children, *Expand All* to show all levels, or *Collapse* to hide all children.
- Change the parent of a subprocess by dragging and dropping it from one process to another.
- Evenly distribute the sub-processes beneath a process in harmonious lines by right-clicking it and selecting *Arrange Symbols*, or selecting *Symbol* > *Auto-Layout*.
- By default, a process hierarchy displays from top-to-bottom. To display it from left-to-right, select *Tools* > *Display Preferences* > *General*, and select *Horizontal* in the *Orientation* group box.
- Hide a process and its children in the hierarchy without deleting it in the model by right-clicking it and selecting *Edit* > *Hide Symbols*. To display any hidden sub-processes underneath a process, right-click it and select *Complete* or *Complete All*.

Each of these processes can be analyzed in its own business process diagram (see Business Process Diagrams (Analysis) [page 20]). You can create a default business process diagram for any of the processes by right-clicking it and selecting *Build Default Flows between Processes*. The default flow links subprocesses in the first level beneath the process between a start and an end. You can further refine the control flow by creating other objects in the diagram.

2.1.1 Reusing Processes in a Hierarchy

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

Procedure

1. Right-click the process within which you want to reuse the process, and select *Reuse Process* to open a selection dialog, which lists all the other processes available in the model.

   > **Note**
   >
   > The *Reuse Process* command is intended to provide a quick means for creating shortcuts to processes in your Process Hierarchy Diagram, primarily when working with the Analysis language, and is not available in other BPM diagrams. For some languages it is hidden completely. If you are working with BPMN or an execution language, it may be more appropriate to set the implementation type of the process reusing another process to **`Reuse process`** or **`Execute operation`** (seeImplementing Processes [page 26]).

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.

   > **Note**
   >
   > You cannot decompose the shortcut or expand its hierarchy, even if its target object has sub-processes.

2.2 Business Process Diagrams (Analysis)

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.

The business process diagram is the core BPM diagram, which lets you:

- Trace the choreography of processes through flows from one or more starts through a sequence of sub-processes, decisions, synchronizations, and resources to one or more ends. The parent process being analyzed in the diagram must wait for the end of all its sub-processes before it terminates.
- Avoid unassigned tasks and duplicated assignments by placing processes in organization unit swimlanes (see Attaching Processes to Organization Units [page 36]).
- Analyze how data flows through a system via:
  - Message formats on flows – To define exchange formats for large amounts of data that transit between processes, usually defined by a DTD or an XSD (see Message Formats (BPM) [page 50]).
  - Data on Flows - To model data (which can be associated with objects defined in a data model or OOM) without specifying its format (see Data (BPM) [page 54]).
  - Data CRUD – To specify the actions (create, read, update and delete) that a process can perform on data (see Data and Resource CRUD Matrices [page 31]).

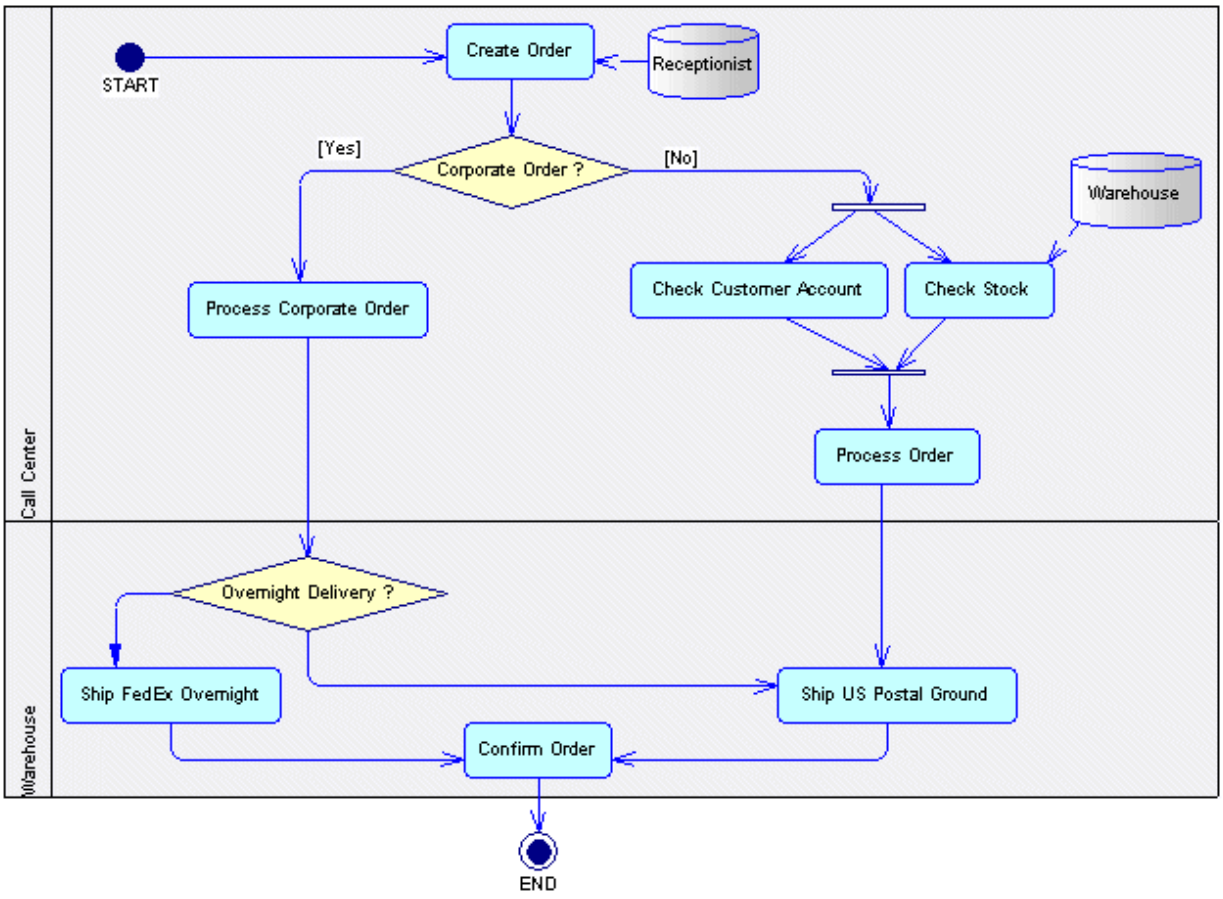
> **Note**
>
> The data flow diagram helps you analyzing data exchange between processes (see Data Flow Diagram (DFD) [page 103]).

- Model the implementation of processes (see Implementing Processes [page 26])

A business process diagram can be created directly under the model root, or inside a package or decomposed process.

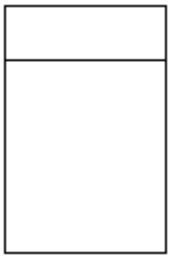
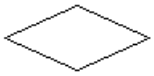
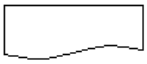
> **Note**
>
> To create a business process diagram in an existing analysis BPM, right-click the model in the Browser and select *New* > *Business Process Diagram*. To create a new model, select *File* > *New Model*, choose Business Process Model as the model type and *Business Process Diagram* as the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

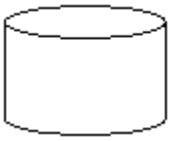
In this example, the processing of an order proceeds differently depending on whether or not it is a corporate order. Both possible paths are reunited in the Confirm Order process:



PowerDesigner supports all the objects necessary to build analysis business process diagrams:

Table 5:

Object	Tool	Symbol	Description
Process			Task to perform (see Processes (BPM) [page 23]).
Organization unit			Organization, service or person that is responsible for a process (see Organization Units (BPM) [page 33]).
Flow			Path of the control flow between processes (see Flows (BPM) [page 47]).
Decision			Decision to take when several flow paths are possible. Only one path will be triggered at execution time (see Decisions (BPM) [page 43]).
Synchronization			Enables synchronization of flows between two or more concurrent actions or allows the design of a split (see Synchronizations (BPM) [page 45]).
Start			Starting point of the processes described in the choreography diagram (see Starts and Ends (BPM) [page 41]).
End			Termination point of the processes described in the choreography diagram (see Starts and Ends (BPM) [page 41]).
Message format	None		Format definition of data exchanged between processes (see Message Formats (BPM) [page 50]).
Data	None	None	Piece of information exchanged between processes (see Data (BPM) [page 54]).

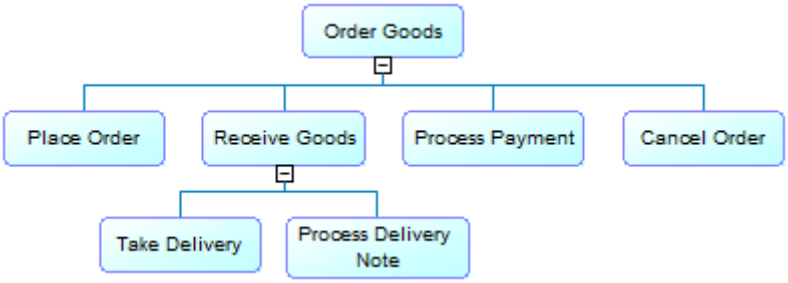
Object	Tool	Symbol	Description
Resource			Storage unit of abstract data circulating within the model, which is accessed by a process to perform actions (see Resources (BPM) [page 66]).
Resource flow			Access of a process to a resource (see Resource Flows (BPM) [page 67]).

2.3 Processes (BPM)

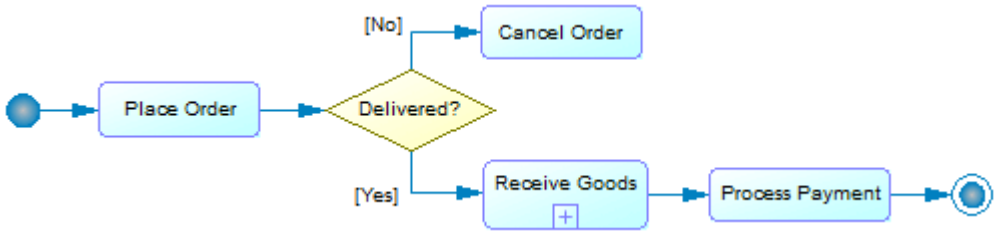
A *process* is a manual or automated action, such as "Process order", or "Send a mail". Processes are the core object in the BPM. Processes can be atomic (without subprocesses) or decomposed/composite (containing subprocesses). Each decomposed process contains its own business process diagram, which shows its subprocesses as part of its control flow.

Processes can be created in models targeting any language, and are commonly used as the basis for other objects in languages other than Analysis.

In the following example, the **`Order Goods`** process is decomposed in a process hierarchy diagram (see Process Hierarchy Diagrams (Analysis) [page 18]) into four subprocesses and the **`Receive Goods`** process is, in turn, decomposed into two subprocesses:



The **`Order Goods`** process contains a business process diagram (see Business Process Diagrams (Analysis) [page 20]), modeling its control flow, which passes from one or more starts to one or more ends (see Business Process Diagrams (Analysis) [page 20]). 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 Implementing Processes [page 26]):



The **`Receive Goods`** process is decomposed, and so its symbol displays a small plus sign overlay and it contains its own business process diagram to model the control flow of its subprocesses

2.3.1 Creating a Process

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

- Use the *Process* tool in a process hierarchy diagram (see Process Hierarchy Diagrams (Analysis) [page 18]) or business process diagram (seeBusiness Process Diagrams (Analysis) [page 20]).
- Select *Model* > *Processes* to access the List of Processes, and click the *Add a Row* tool.
- Right-click the model, a package or a decomposed process in the Browser, and select *New* > *Process*.
- [executable BPMs] Drag an operation (see Operations (BPM) [page 89]) from the Browser and drop it into a diagram to create a process that invokes the operation.

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

2.3.2 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:

Table 6:

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	Specifies the organization unit (see Organization Units (BPM) [page 33]) that performs the process. Select **<Committee Process>** to specify that it is realized by multiple organization units (see Displaying a Committee Process [page 36]). 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.
Timeout	Specifies the timeout limit, which is, by default zero. You can specify any alphanumeric value (for example, **20 seconds**) to indicate that a timeout exception occurs if the execution of the activation takes longer than that.
Duration	Specifies the estimated or statistic duration to execute the action. This property is used for documentation purposes.
Composite status	Specifies whether the process contains sub-processes. You can choose between: • Atomic Process (default) – the process does not contain any sub-processes. • Decomposed Process – the process can contain sub-processes, which are listed on a *Sub-Processes* tab and can be displayed in a business process diagram under the process (see Decomposing Processes [page 27]). If you revert the process from Decomposed to Atomic status, then any sub-processes that you have created will be deleted.
Number ID	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. Process numbering is commonly used in data flow diagrams (see Data Flow Diagram (DFD) [page 103]).
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

The following tabs are also available:

- *Implementation* - Specifies how the process is implemented (see Implementing Processes [page 26]).
- *Assignments* - [processes with the **Assign** implementation type] Lists the data transformations (see Data Transformations (BPM) [page 99]) required for the atomic assign tasks that compose the activity.
- *Sub-Processes* - [decomposed processes] Lists the sub-processes contained in the process (see Decomposing Processes [page 27]).
- *Local Variables* - [orchestration language decomposed processes] Lists the variables (see Variables (BPM) [page 96]) 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) [page 54]).

> **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 [page 65]).

2.3.3 Implementing Processes

You can add additional detail to your processes by specifying the type of implementation required for their execution. Depending on the implementation type, additional fields or tabs may be displayed, allowing you to specify a process, event, expression, operation, or data transformation upon which the implementation acts.

Context

> **Note**
>
> In BPEL (see BPEL4WS 1.1 and WS-BPEL 2.0 [page 158]), decomposed processes cannot have their implementation specified.

Procedure

1. Open the property sheet of the process and click the *Implementation* tab.

   > **Note**
   >
   > You can open the *Implementation* tab directly by right-clicking the process symbol in the diagram, and selecting *Implementation*.

2. Select an implementation type. The following list details the available implementation types, and specifies where appropriate, the required implementation object:

   Table 7:

Type	Description/Properties
None	[default] No implementation is defined, or the implementation consists of an informational description in the text box.
Loop	Transforms the process into a composite process (see Decomposing Processes [page 27]), which will iterate over the set of sub-processes that it contains. The following properties are displayed: ○ *Loop expression* - Specifies the loop condition. ○ *Loop type* - Specifies the loop type. Some languages provide predefined types.

Type	Description/Properties
Reuse process	[Analysis and BPMN] Uses another process to implement the process, which you specify in the *Implemented by* field.
Execute operation	[BPMN, SOA, and BPEL] Implements the process through a service operation to design the reception and emission of messages (see Linking an Operation to a Process [page 93]).
Generate Event	[BPMN, SOA, and BPEL] Specifies the generation of events, and can be used to raise an exception. The following properties are displayed: ○ **Implemented by** - Specifies the implementation event (see Events (BPM) [page 74]). You can specify events to model the following specific activities: ○ 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 Event Handlers [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	[SOA and BPEL] – Uses a data transformation to copy a variable value to another variable value, or to calculate the value of an expression and store it in a variable via an Xpath or XSLT expression. Enables the display of the Assignments tab (see Process Properties [page 24])

3. [None or Reuse process implementations] Specify the way the process should be executed. You can choose between:
   - Manual
   - Automated
   - User-defined
4. [optional, except Execute Operation] Specify any additional information about the process execution in the text box. You can enter any appropriate information in this box, as well as open, insert and save text files.
5. 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.

2.3.4 Decomposing Processes

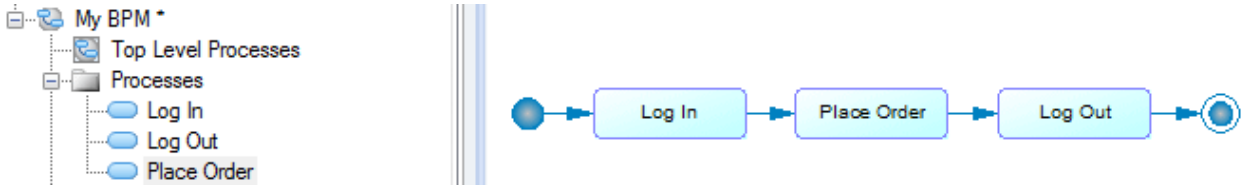
You can decompose processes into subprocesses to analyze them in more detail. The decomposed process has its own sub-diagram, which models the control flow or data flow between its sub-processes. Sub-processes can be further decomposed until you reach a sufficient level of detail or atomic tasks that cannot be further decomposed.

You can decompose a process and create an empty business process diagram under it in the following ways:

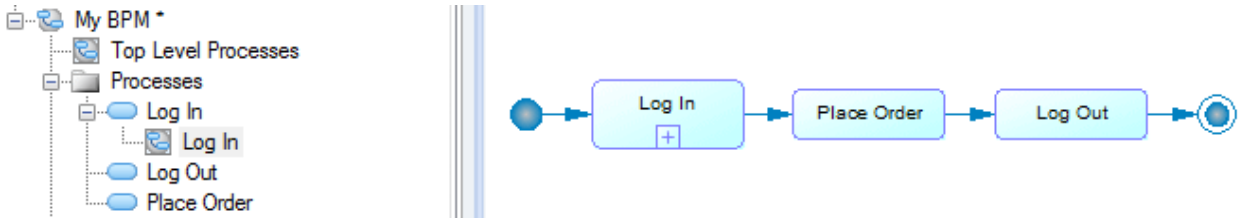
- Use the *Process* tool to create a subprocess beneath the process in a process hierarchy diagram (see Process Hierarchy Diagrams (Analysis) [page 18]).
- In a business process diagram or process hierarchy diagram, press CTRL and double-click the process symbol to decompose the process and open the new diagram.
- Right-click the process in the diagram or Browser and select *Decompose Process*.

- Open the property sheet of the process and, on the *General* tab, select the *Decomposed Process* radio button.

Any objects that you create in the sub-process diagram are listed in the Browser under the decomposed process. In the following example, we begin with three processes in the **`Top Level Processes`** diagram:



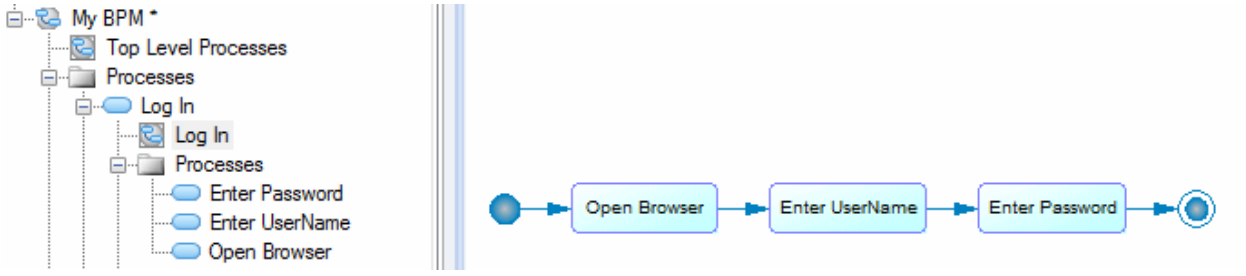
Then the **`Log In`** process is decomposed. Its symbol acquires a plus sign overview, and a new diagram is created beneath it in the Browser:



You can navigate in the hierarchy of diagrams as follows:

- To descend into the subdiagram beneath a decomposed process, press *CTRL* and double-click its symbol, or (double-click the diagram node in the Browser).
- To go up a level in the diagram hierarchy, right-click the diagram background and select *Diagram* > *Go Up One Level*.
- To go to a diagram on the same level of the hierarchy, right-click the diagram background and select *Diagram* > *Open Sibling Diagram* > **`<diagram>`**.
- To go to any diagram in the model, right-click the diagram background, select *Diagram* > *Select Diagram*, and choose the diagram from the tree.

The diagram is empty at first. We rename it and create three new processes, a start, and an end to provide a complete control flow. These objects are listed under **`Log In`** in the Browser:



> **Note**
>
> In general, we recommend that you create only one diagram under each decomposed process to capture its entire control flow, but it may in certain cases be appropriate to create addtional diagrams to model exception

cases such as for error management. You cannot create a package inside a decomposed process, but you can use shortcuts to packages.

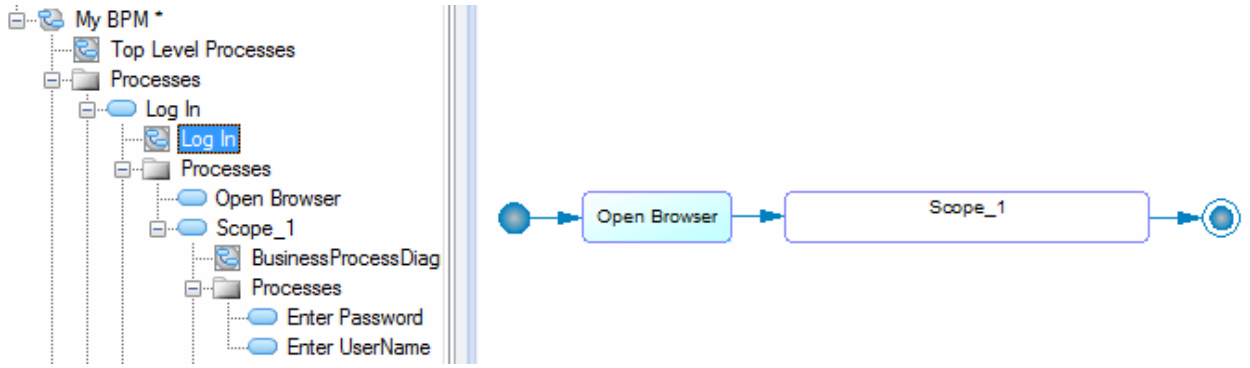
You can group existing processes and other control flow objects into a new decomposed process, which has the effect of adding a new level of decomposition above them:

- Select one or more symbols, and then select *Tools* > *Create Composite Process* to replace them with a new composite process with its symbol set to composite view to display the sub-objects.

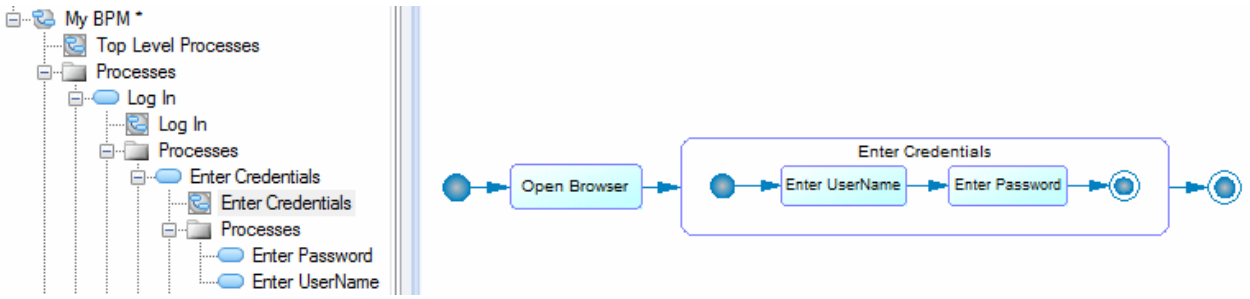
  > **Note**
  >
  > A start and end are added to the decomposed process to make a complete control flow. Any starts or ends included in your selection are not moved.

- Right-click the business process diagram background and select *Diagram* > *Convert to Composite Process* (or right-click the diagram node in the Browser and select *Convert to Composite Process*), enter a name for the new process and select the processes from the diagram that you want to move into it. Any processes that you do not select remain at their present level and are represented in the new sub-process diagram as shortcuts.

In our example, we select the two processes **`Enter UserName`** and **`Enter Password`**, and select *Tools* > *Create Composite Process*. The processes are moved to under a new process, provisionally named **`Scope_1`**, which replaces them in the **`Log In`** diagram:

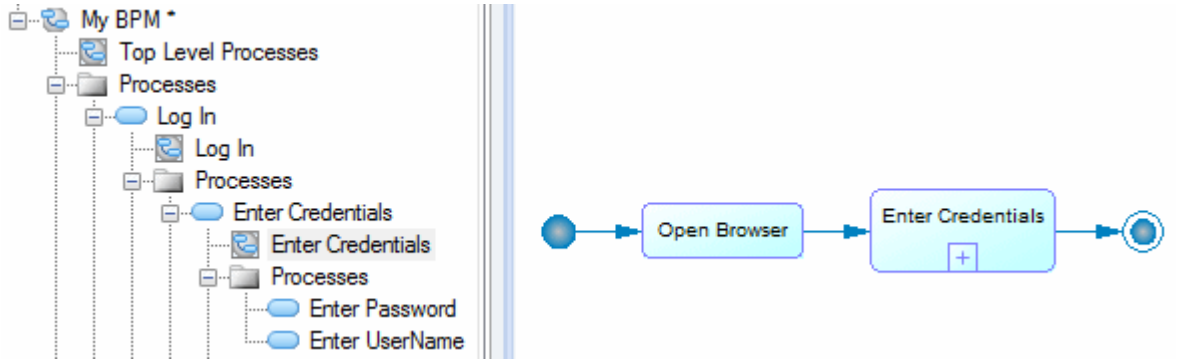


We rename the process and its diagram to **`Enter Credentials`**, and right-click the symbol and select *Composite View* > *Adjust to Read-Only View* to display the sub-processes:



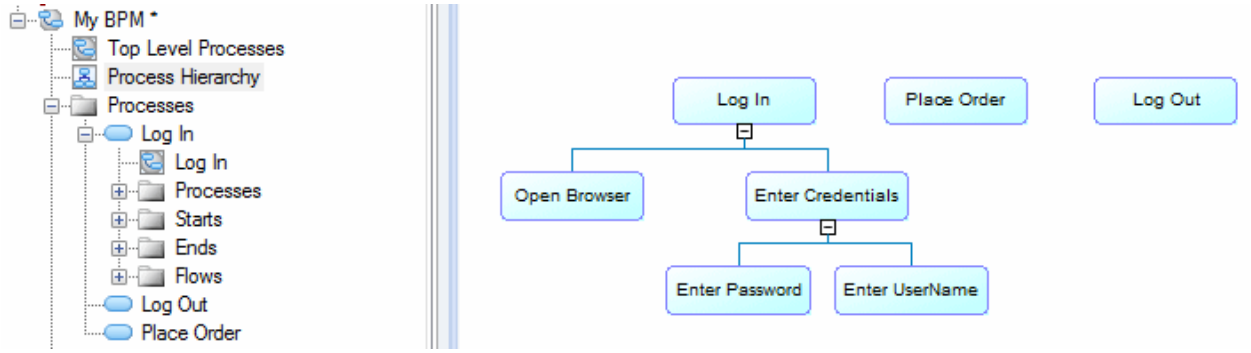
If the sub-diagram is very large, it may be shrunk to fit in the symbol. You can adjust the symbol manually to change the zoom level.

To hide the subdiagram, right-click the symbol and select *Composite View* > *None*:



To redisplay the sub-symbols, right click the symbol and select *Composite View* > *Read-only (Sub-Diagram)*.

You can view the complete structure of your processes in a process hierarchy diagram (see Process Hierarchy Diagrams (Analysis) [page 18]:). You may need to select *Symbol* > *Show Symbols* to add your decomposed processes to the diagram:

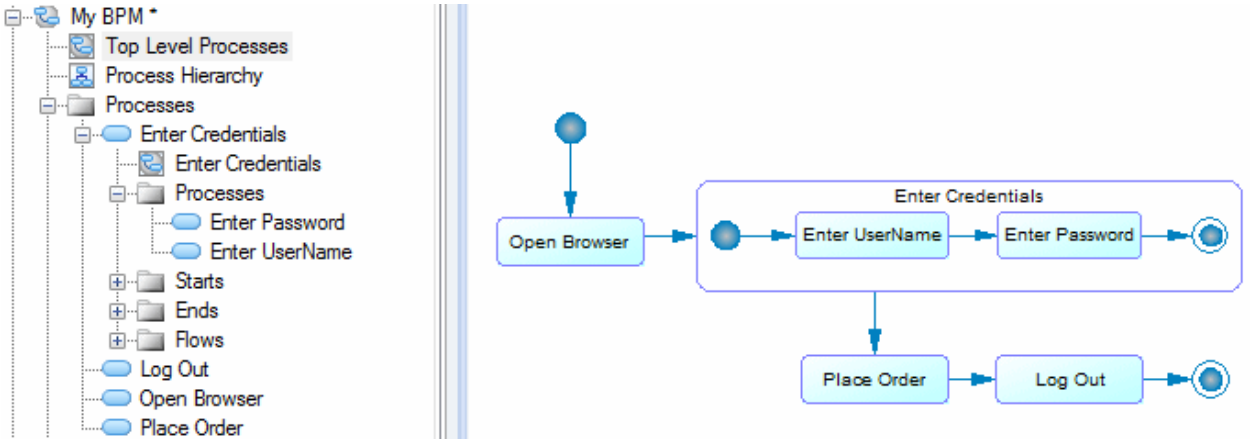


You can view your processes in list form by selecting *Model* > *Processes*. To display all processes in the List of Processes, including those belonging to decomposed processes, click the *Include Composite Processes* tool.

You can remove detail from your process hierarchy by right-clicking a decomposed process symbol and selecting:

- *Change to Atomic Task* - Deletes all objects under the decomposed process.
- *Remove Composite Process Level* - Deletes the decomposed process and replaces it in the control flow of the diagram with its child objects.

In our example, we right-click the **`Log In`** process symbol in the **`Top Level Processes`** diagram and select *Remove Composite Process Level*, and it is replaced in the control flow by the **`Open Browser`** and **`Enter Credentials`** processes:



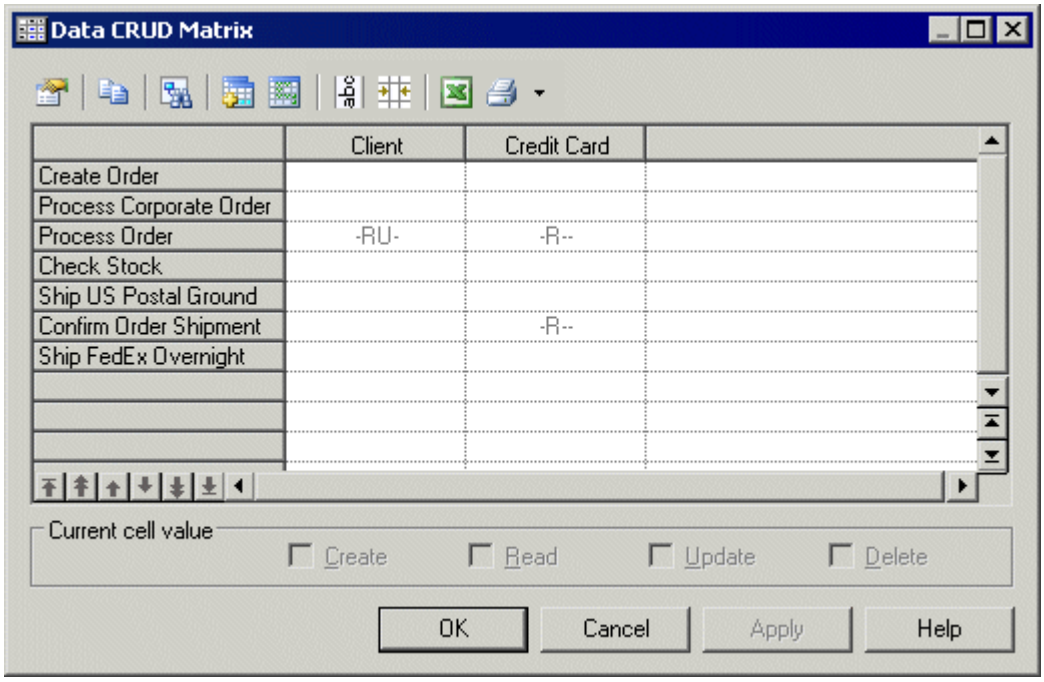
2.3.5 Data and Resource CRUD Matrices

A *CRUD matrix* is a table that allows you to observe and modify the actions (Create, Read, Update, or Delete) your processes perform on data or resources. Once you have created at least one process and one resource or data object, you can open the relevant matrix by selecting *Tools* > *Resource CRUD Matrix* or *Tools* > *Data CRUD Matrix*.

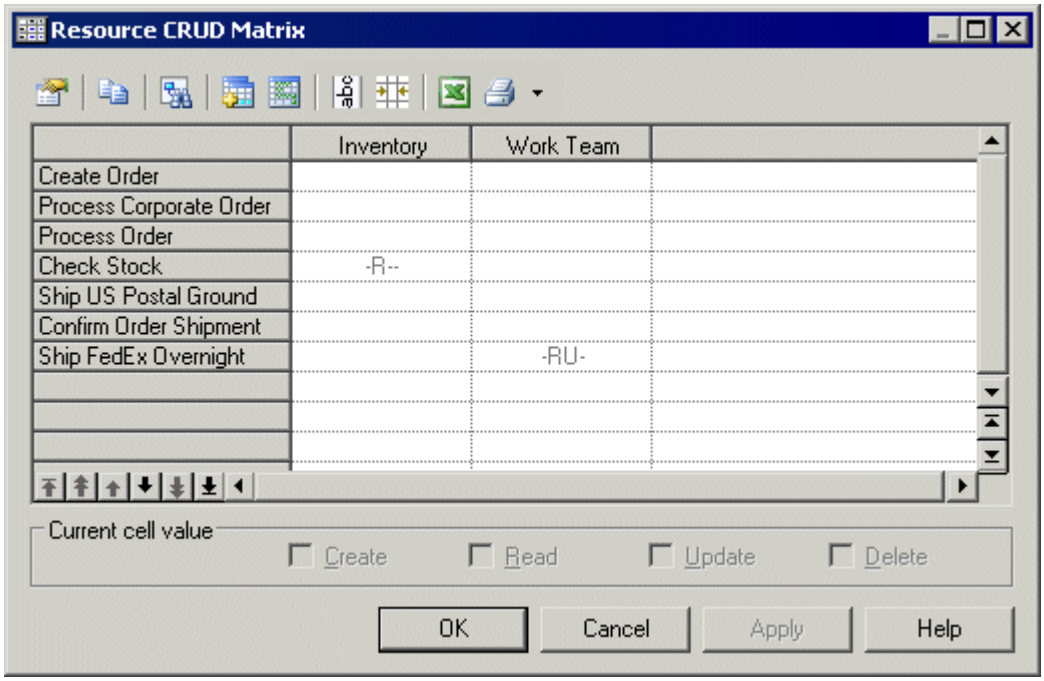
To modify the CRUD values for a process, select the appropriate cell and select or deselect the check boxes in the *Current Cell Value* groupbox.

> **Note**
>
> A process must already be associated with data or resources in order for it to be included in the matrix.

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:

Table 8:

Tool	Description
	Properties – Opens the property sheet of a: • 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. If parallel resource flows exist between a process and a resource, you are prompted to choose one.
	Copy – Copies a CRUD matrix in order to paste it into another application such as Excel (as CSV) or Word (as text).
	Find Symbol in Diagram – Finds in the diagram the symbol of a: • 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.
	Select Rows/Columns – Opens a selection box listing all the available objects, which allows you to add or remove rows and columns.
	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.

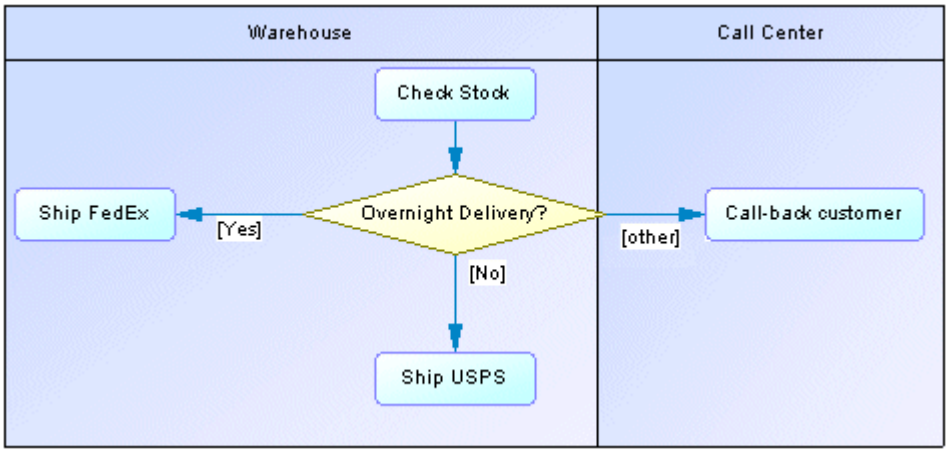
2.4 Organization Units (BPM)

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.

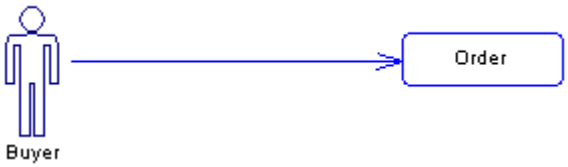
> **Note**
>
> To enable the display of organization unit swimlanes, select *Tools* > *Display Preferences*, and select the *Organization unit swimlane* checkbox on the *General* page, or right-click in the diagram background and select *Enable Swimlane Mode*.

Organization units can be created in models targeting any language, and are commonly used as the basis for other objects in languages other than Analysis.

In a business process diagram, the organization unit is displayed as a swimlane and allows you to assign responsibilities within your system. In this 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:



In a BPEL4WS or WS-BPEL top-level diagram, the organization unit is displayed as an actor and allows you to identify the external partners who interact with your system. In this example, the Buyer organization unit interacts with the Order process:



In a data flow diagram, the organization unit is displayed as a box and allows you to identify external entities that send or receive data from the system.

2.4.1 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. Click in or next to an existing swimlane or pool of swimlanes to add a swimlane to the pool. Click in space away from existing swimlanes to create a new pool.
- 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* > *Modeling with PowerDesigner* > *Objects*.

2.4.2 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:

Table 9:

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. An organization unit has the following predefined stereotypes: • 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	Specifies another organization unit as the parent to this one. 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. The relationship between parent and child organization units can be used to group swimlanes having the same parent (see Grouping Swimlanes [page 39]).
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

2.4.3 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.

2.4.4 Displaying a Committee Process

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

Procedure

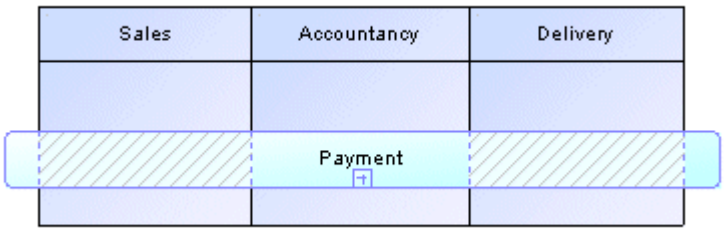
1. Open the property sheet of a decomposed process.
2. Select *Committee Process* from the Organization Unit list and click *OK*.

   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.

Results

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:

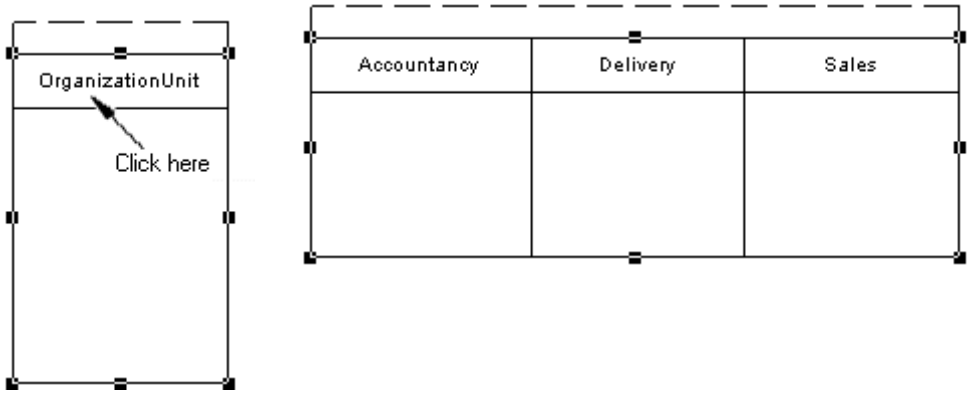
- Manage any sub-processes
- Have any symbol in the sub-process diagram

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

2.4.5 Moving, Resizing, Copying, and Pasting Swimlanes

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.

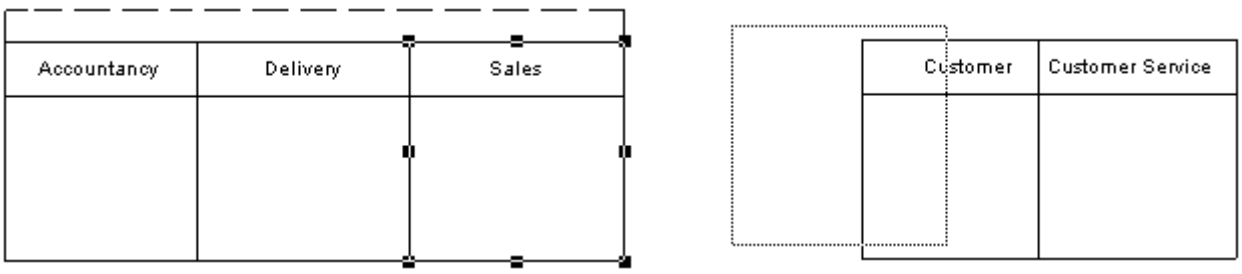
Table 10:



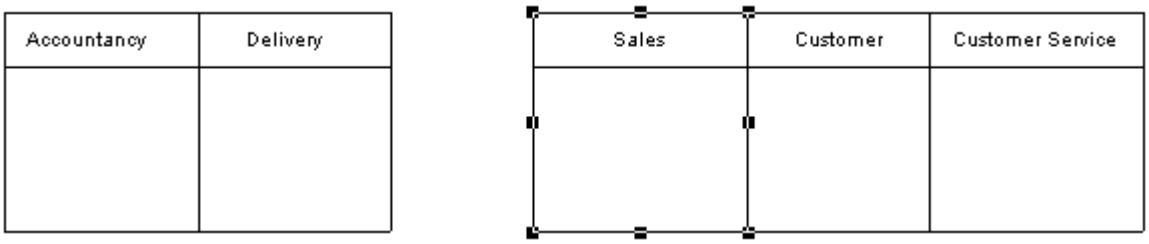
> **Note**
>
> The auto-layout function is unavailable with organization units displayed as swimlanes.

If 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). If you move or copy a swimlane or pool to another diagram, the symbols inside the swimlane(s) are not copied.

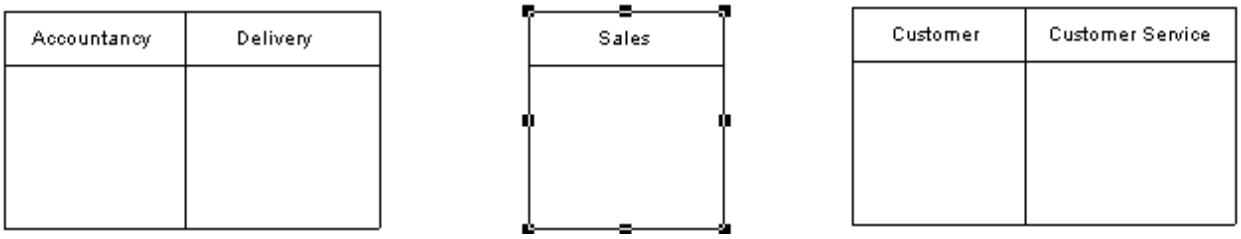
If a swimlane is dropped on or near another swimlane or pool, it joins the pool. In the following example, Sales forms a pool with Accountancy and Delivery:



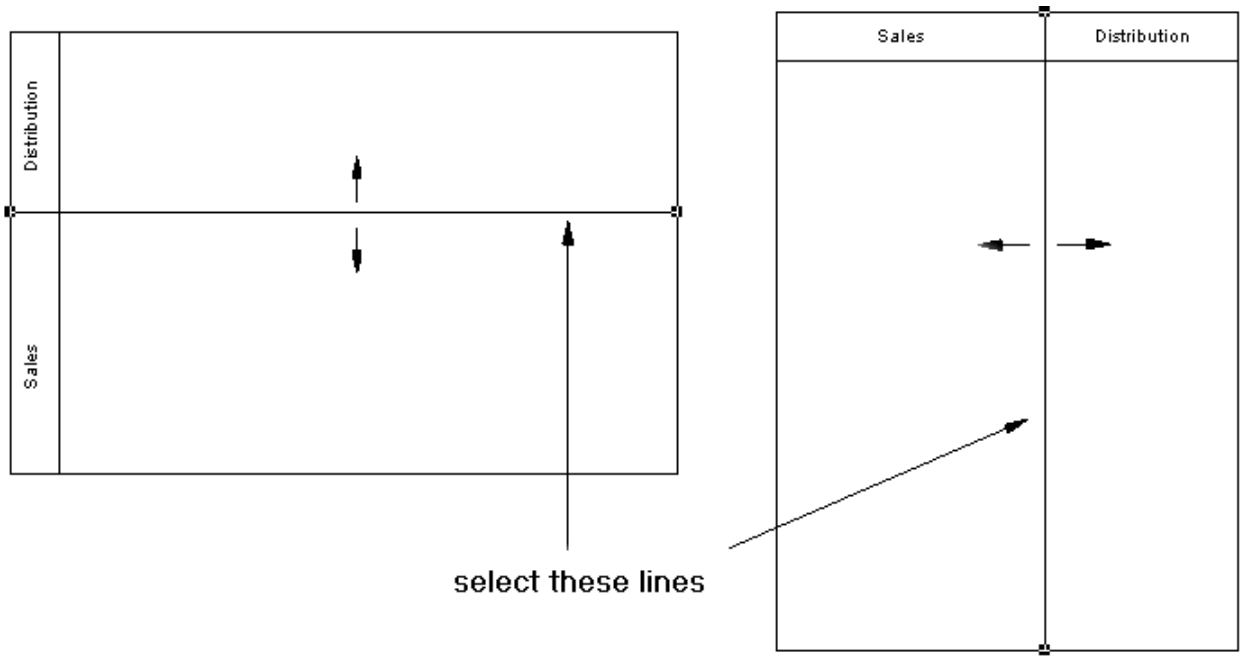
It is moved to another pool containing Customer and Customer Service



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



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



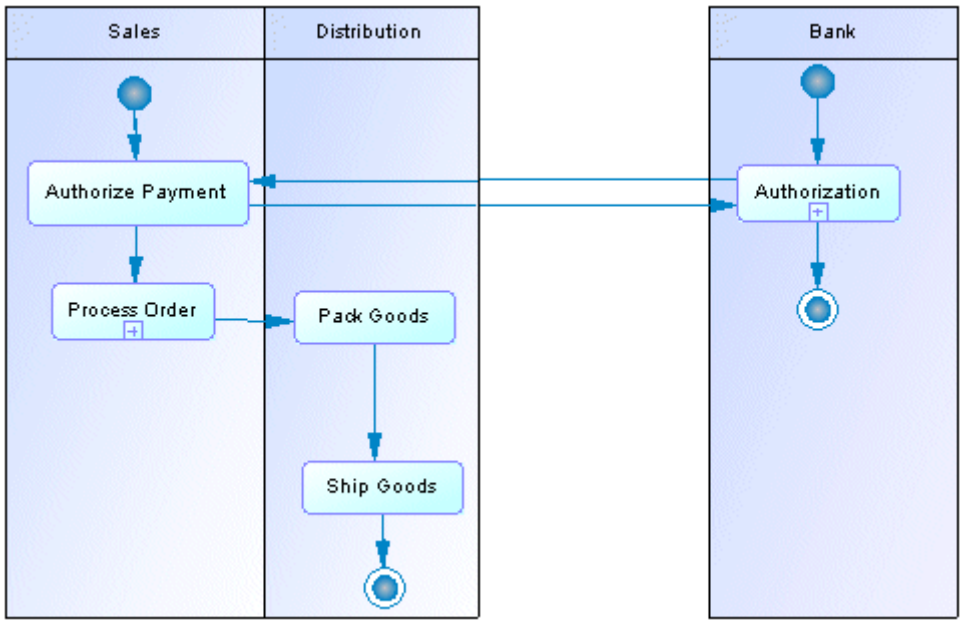
When you change the width or height of an individual swimlane, all process symbols attached to the swimlane keep their position.

2.4.6 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.

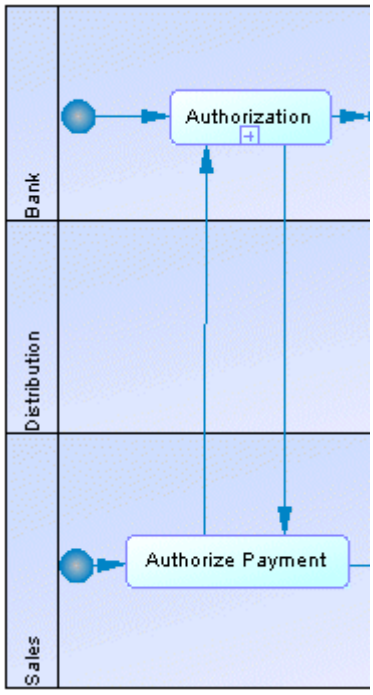
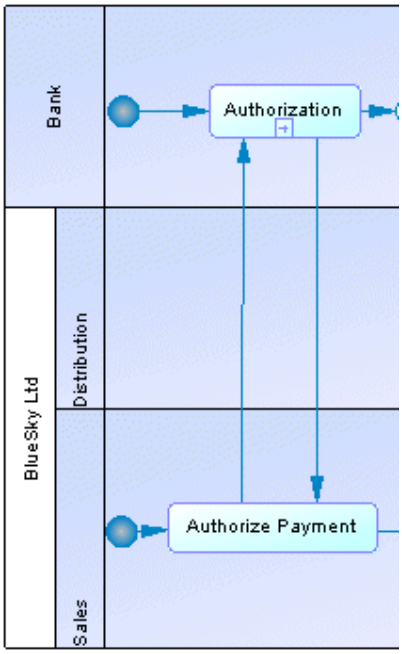
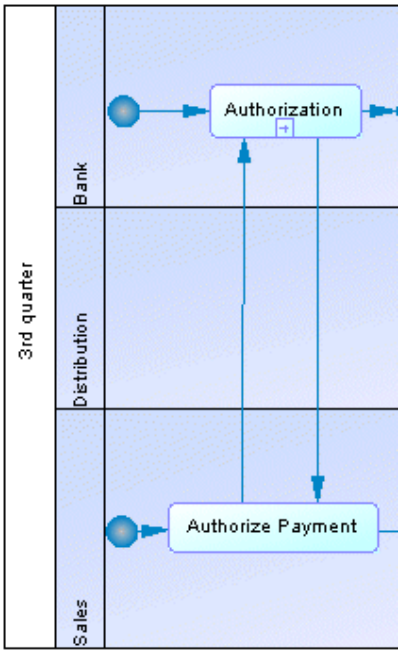
2.4.7 Grouping Swimlanes

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

To group swimlanes within a pool, select the pool, then right-click it and select *Swimlane Group Type*, and then:

- *By Parent* - to assign the name of the immediate common parent for the group
- *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.

Table 11:

No Group	Parent Group	User-Defined Group
The three swimlanes are in a pool, without grouping: 	Sales and Distribution are grouped by their parent: 	The pool is assigned a user-defined group named 3rd quarter: 

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

2.4.8 Changing the Orientation and Format 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. You can change the presentation of organization units to swimlane or actor.

Context

Select *Tools* > *Display Preferences*, select the appropriate radio button in the *Organization unit swimlane* groupbox, and click *OK*.

By default, you use organization units displayed as actors in top-level diagrams and as swimlanes in business process diagrams. To modify this default behavior:

- 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*.

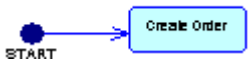
When you switch organization unit representations, the symbols are often deleted from the diagram. To residplay the symbols, you can either drag the organization units from the Browser and drop them onto the diagram or right-click the diagram background and select *Show Symbols*.

2.5 Starts and Ends (BPM)

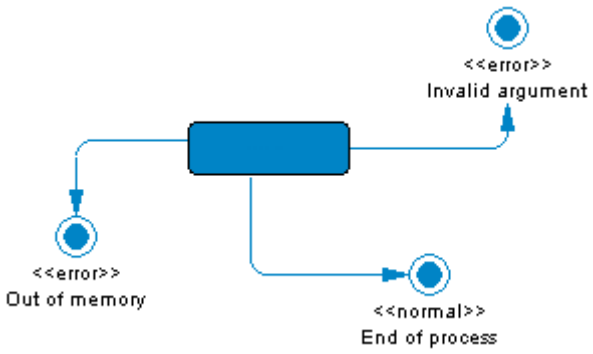
A *start* is a starting point of the flow represented in the diagram, and an *end* is a termination point of the flow.

Starts and ends can be created in models targeting any language except DFD.

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.



You can create several ends within the same diagram to show divergent end cases, such as error scenarios:



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

> **Note**
>
> The start is compared and merged when merging models to ensure that there is no additional start in decomposed processes .

2.5.1 Creating a Start or an End

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

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

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

2.5.2 Start and End Properties

To view or edit a start or 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.

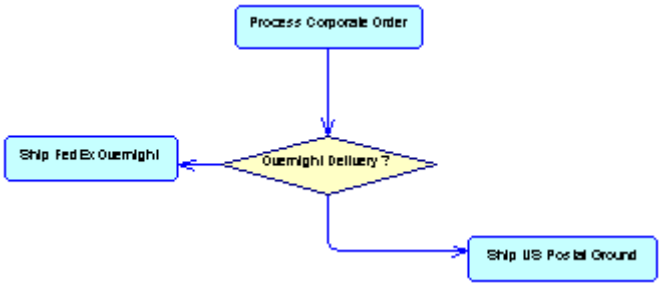
The *General* tab contains the following properties:

Table 12:

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	[ends only] Specifies the type of the end used for document purposes. You can enter your own type in the list, or choose one of the following values: • 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.

2.6 Decisions (BPM)

A *decision* specifies which path to take, when several paths are possible. A decision can have one or more input flows and one or more output flows, each labeled with a distinct *guard condition*, which must be satisfied for its 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.

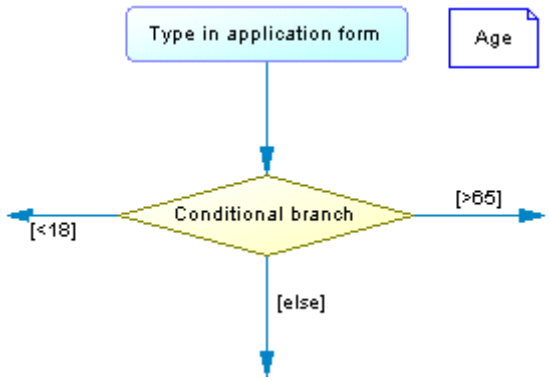
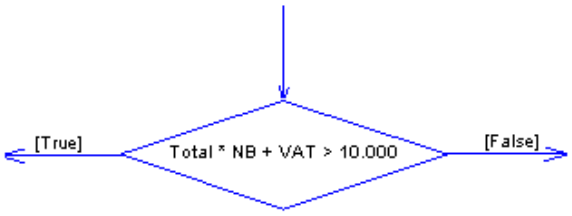


Decisions can be created in models targeting any language except DFD.

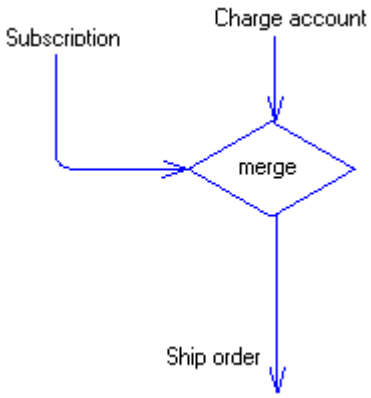
A decision can represent:

- A conditional branch: one input flow and several output flows. You can display a condition on the decision symbol in order to factorize the conditions attached to the flows:

  Table 13:

Without Condition on Symbol	With Condition on Symbol
In this 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:	In this, 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**
>
> You cannot attach two flows of opposite directions to the same corner of a decision symbol.

2.6.1 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* > *Modeling with PowerDesigner* > *Objects*.

2.6.2 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:

Table 14:

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.

The following tabs are also available:

- *Condition* - contains the following properties:

  Table 15:

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.

2.7 Synchronizations (BPM)

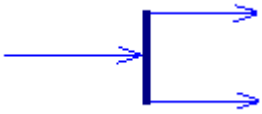
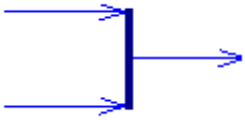
A *synchronization* enables the splitting or synchronization of control between two or more concurrent actions.

Synchronizations can be created in models targeting any language except BPMN or Solution Manager.

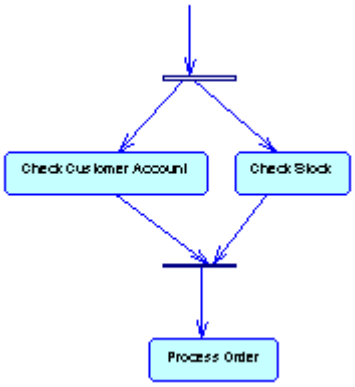
Synchronizations are represented as horizontal or vertical lines. To change the orientation of the symbol, right-click it and select *Change to Vertical* or *Change to Horizontal*.

A synchronization can be either a:

Table 16:

Fork	Join
Splits a single input flow into several output flows executed in parallel: 	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 entering the first synchronization is split into two 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:



2.7.1 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*.

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

2.7.2 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:

Table 17:

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.

The following tabs are also available:

- *Action* - contains the following properties:

  Table 18:

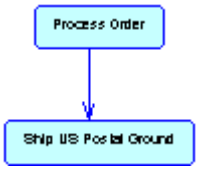
Property	Description
Action (text box)	Specifies an action to be evaluated to determine how the synchronization should be traversed. You can enter any appropriate information in this box, as well as open, insert and save text files.
Timeout	Specifies the timeout limit. The default value is zero. When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

2.8 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.

Flows can be created in models targeting any language.

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.

2.8.1 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* > *Modeling with PowerDesigner* > *Objects*.

2.8.2 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:

Table 19:

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.
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.
Source / Destination	Specify the objects that the flow leads from and 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 source and destination objects by clicking the buttons in the top section of the flow property sheet.
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: • Fax delivery • Mail • Telephone
Flow type	You can enter your own type of flow in the list, or choose one of the following values: • Success - defines a successful flow • Timeout - defines the occurrence of a timeout limit • Technical error • Business error • Compensation - defines a compensation flow The flow type is unavailable if you associate an event with the flow on the Condition tab.
Message format	[Not available for executable languages]. Specifies the format of the data exchanged between processes: • None – a simple flow with no exchange of data. • Undefined (default value) – a flow whose message format is not yet defined. Click the *Create* tool to the right of the 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.

The following tabs are also available:

- *Condition* - Contains the following properties:

  Table 20:

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.
Mapping	[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.

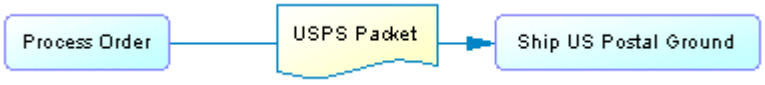
- *Data* - lists the data conveyed by the flow without any information on its format. You can add or create data using the Add Objects and Create an Object tools . You can also migrate data to a source or destination process . In an Analysis business process diagram, if you have specified data for a 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.

2.9 Message Formats (BPM)

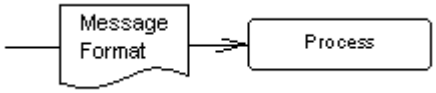
A *message format* can be an XML document or a list of parameters, which defines the format of the data exchanged between processes.

Message formats can be created in models targeting any language except DFD.

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:



In an Analysis model, 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:



In models targeting other languages, a message format is used to specify the format of the message associated with an operation (see Operations (BPM) [page 89]).

In some cases, it may be appropriate to decompose a message format into message parts that specify its contents (see Message Parts (BPM) [page 52]).

2.9.1 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* > *Modeling with PowerDesigner* > *Objects*.

2.9.2 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:

Table 21:

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.

The following tabs are also available:

- *Definition* - Contains the following properties:

Table 22:

Property	Description
Type	Specifies how the message format is defined. You can choose between: ○ 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. ○ 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 *XML Modeling*.
External definition	[External file and URL only] Specifies the location path to an external file or an URL.
Message format type	[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: ○ XML Schema ○ DTD ○ RELAX NG
Message format definition	[Embedded or External file and URL only] Specifies the content of the message.

- *Data* - [Analysis only] 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 [page 63]). 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 [page 49]).
- *Dependencies* - When working with execution languages, displays in the following sub-tabs the different uses of the message format:
  - *Operation Input Message* - All the operations that use the message format as input.
  - *Operation Output Message* - All the operations that use the message format as output.
  - *Fault Message Links* - All uses of the message format as a fault on an operation.
  - *Typed Variables* - All the variables that use the message format as a data type.

2.9.3 Message Parts (BPM)

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

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

Creating a Message Part

You create message parts on the *Definition* tab of the message format property sheet, by selecting the *Message Parts* radio button, and then using the *Add a Row* tool. Enter a name and code for each part and, if appropriate, click the *Properties* tool to specify additional properties.

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:

Table 23:

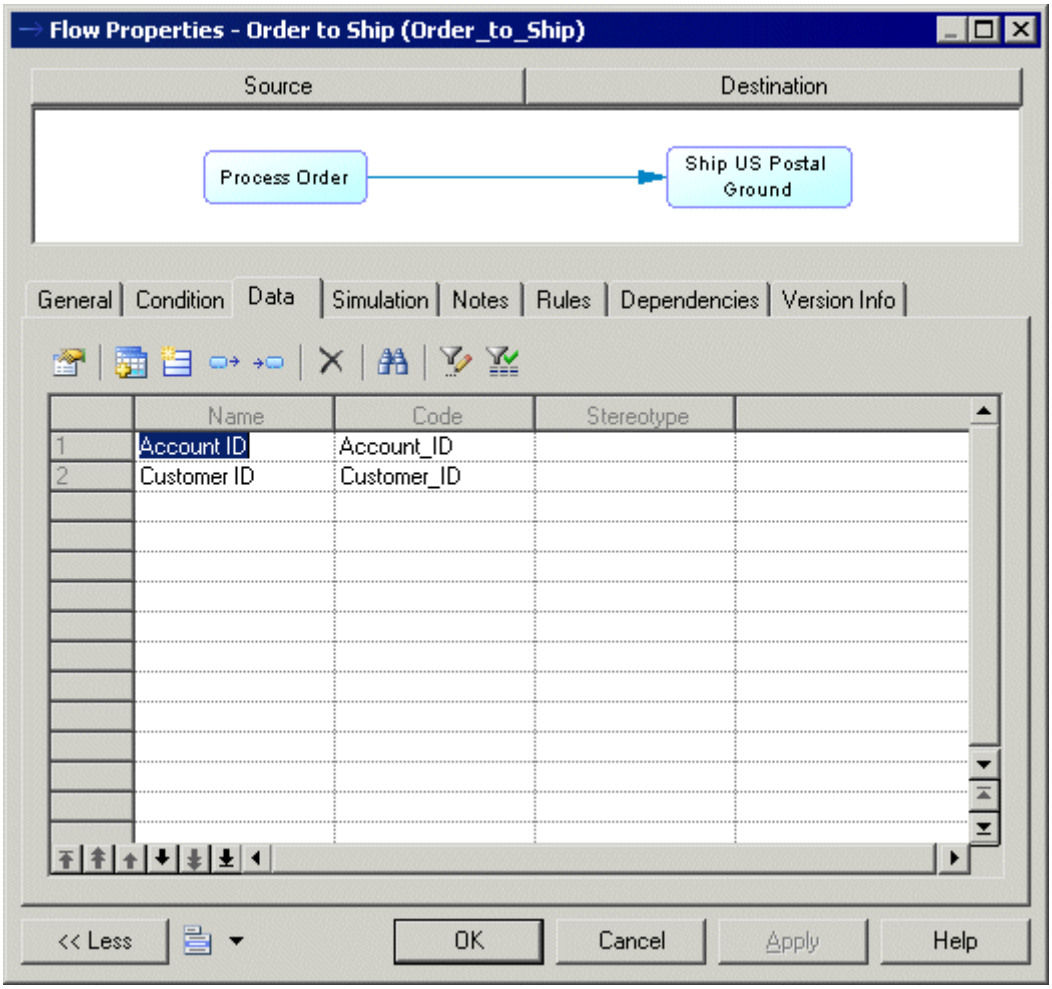
Property	Description
Parent	[read-only] Specifies the parent message format.
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include 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.
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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

2.10 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.

Data objects can be created in models targeting the Analysis or Data Flow Diagram languages. 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. 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:

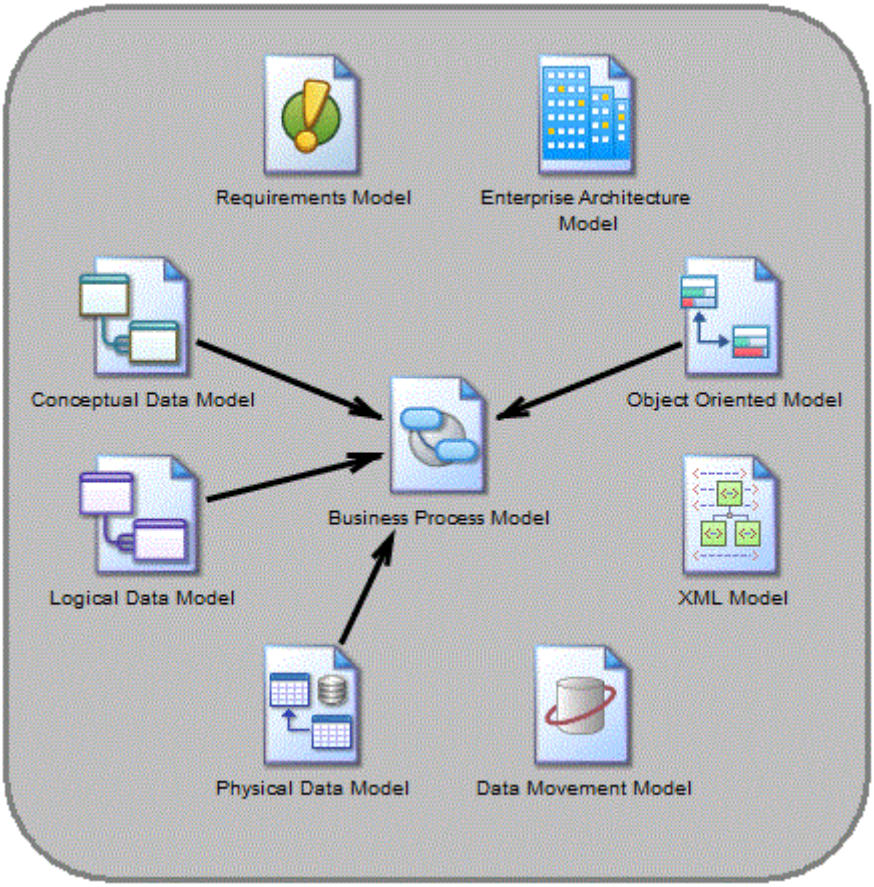


- Message format – to identify the type and the format of data exchanged between a resource and a process or between processes.
- Process – to identify the type of action (Create, Read, Update, and Delete) the process performs on the data required for its execution.

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 have 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 analyze further the nature of the piece of information exchanged (see Linking Data with Other Model Objects [page 56]):



2.10.1 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* > *Modeling with PowerDesigner* > *Objects*.

2.10.2 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:

Table 24:

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: • 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 (see Linking Data with Other Model Objects [page 56]).
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

The following tabs are also available:

- *Sub-Data* - (data objects with the structured data type] 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**
>
> Click the *Dependencies* tab of a sub-data property sheet to display the data of which it is a part.

2.10.3 Linking Data with Other Model Objects

You can model BPM data objects in more detail by linking them 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.

Context

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

Table 25:

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:

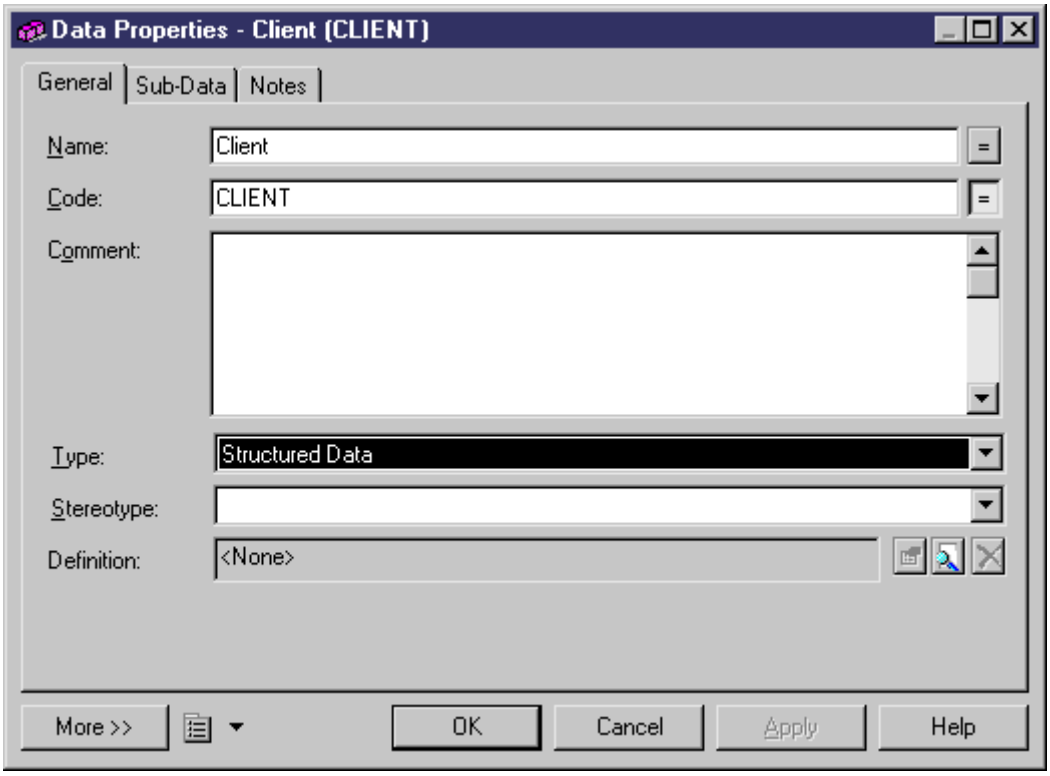
Table 26:

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

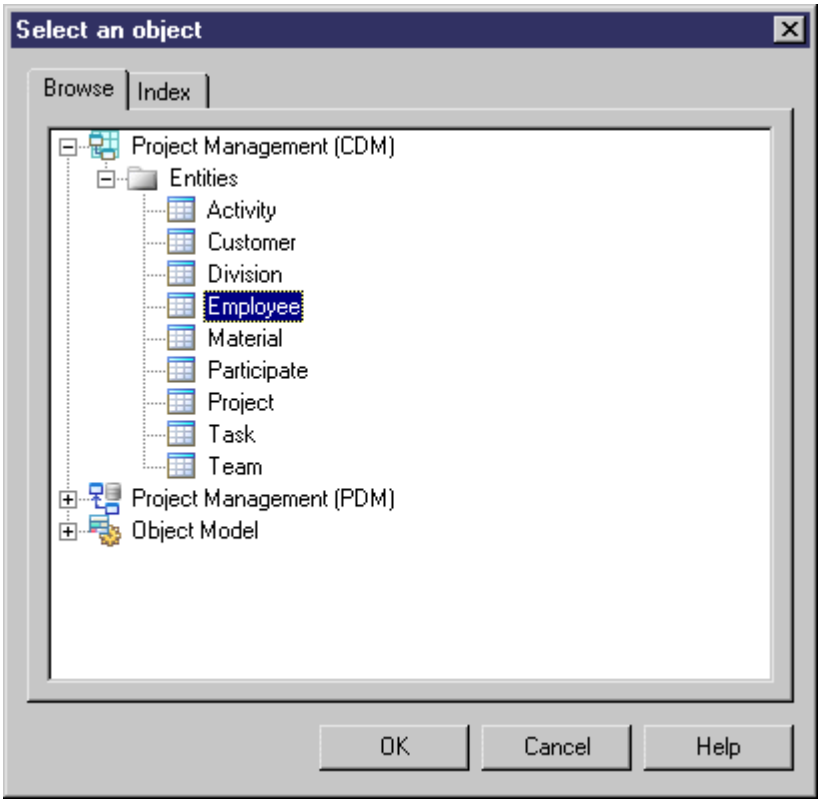
> **Note**
>
> If you export sub-data objects without their parent, then the rules above for exporting data objects are applied.

Procedure

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.

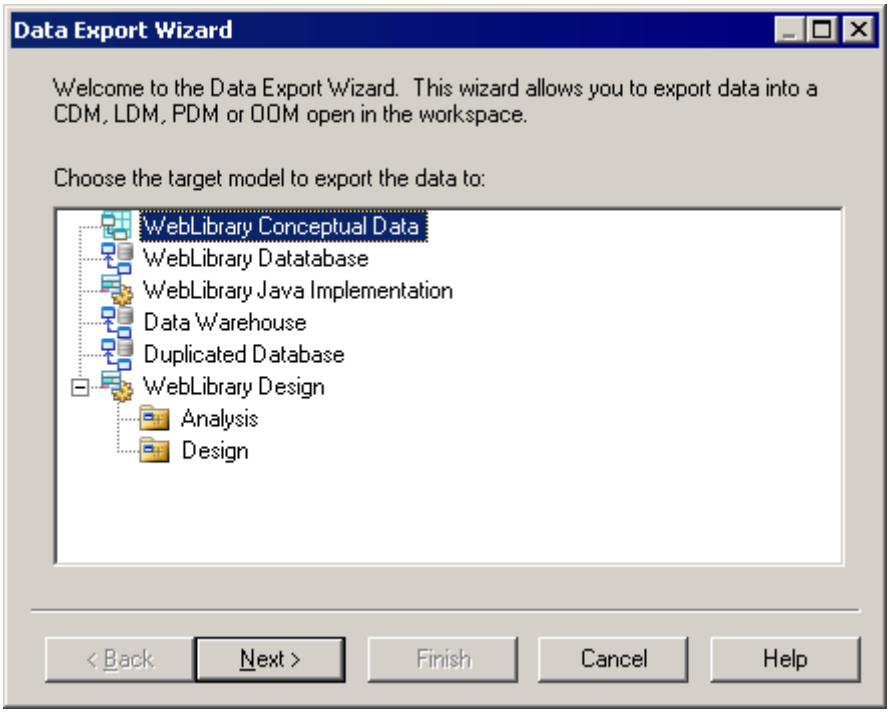
   Note that if you subsequently change the type of the data so that it no longer corresponds with the object defined in the *Definition* box, you will be prompted to confirm the change. If you do so, the link between the data and the object is removed.

2.10.3.1 Exporting Data to Other Models

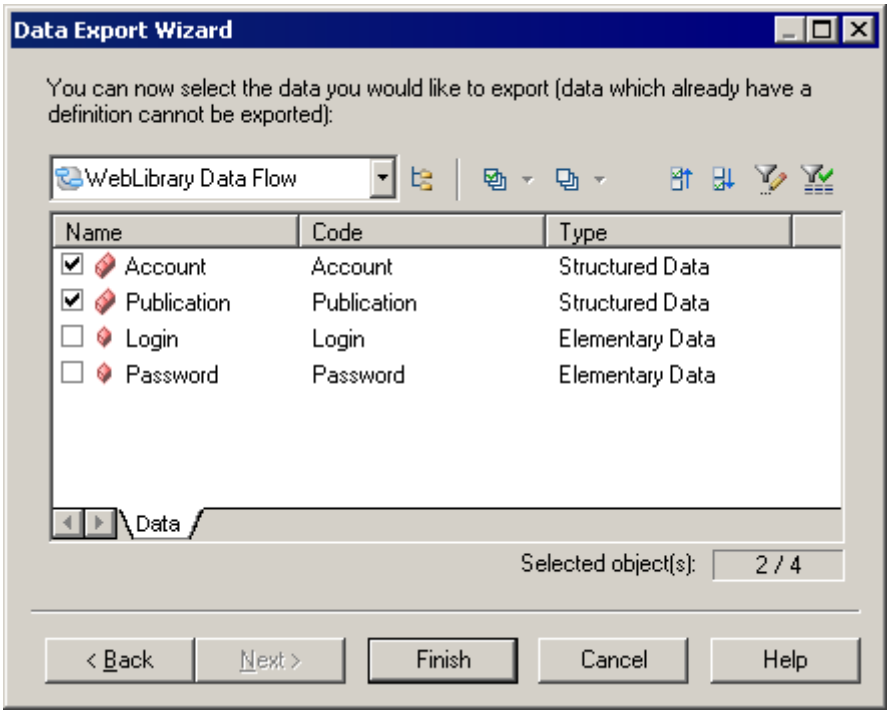
You can export BPM data objects as data items, entities, tables or classes in other PowerDesigner models. The target model must be open in the workspace, and only data objects which are not already linked to any external model objects are available for export.

Procedure

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. 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.

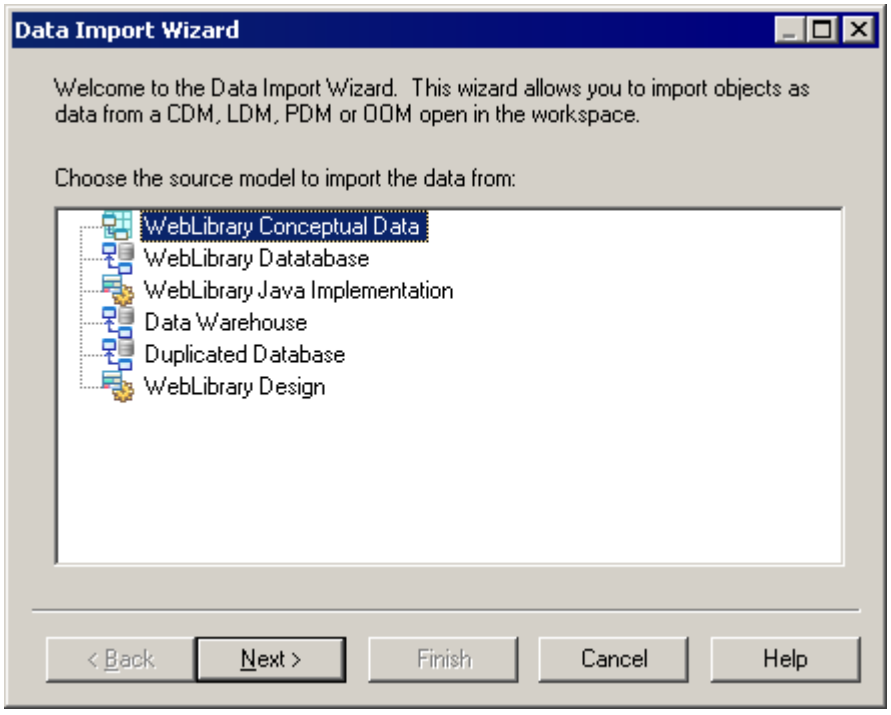
   > **Note**
   >
   > A class attribute or a table column cannot be shared, but sub-data objects can be shared by several data object parents. When you export an elementary or undefined sub-data object as an attribute in the OOM or as a column in the PDM, the link between the sub-data object and the definition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

2.10.3.2 Importing Data from Other Models

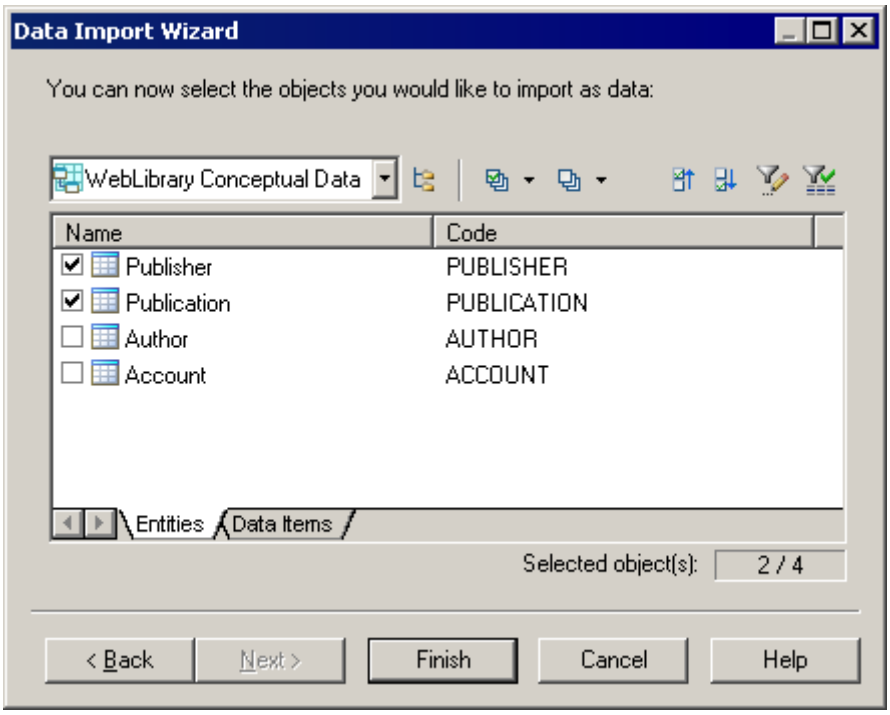
You can import CDM data items and entities, LDM entities, PDM tables, and OOM Classes as data objects in your BPM. The source model must be open in the workspace and only external model objects not already linked to data objects in the model are available for import.

Procedure

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. 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.

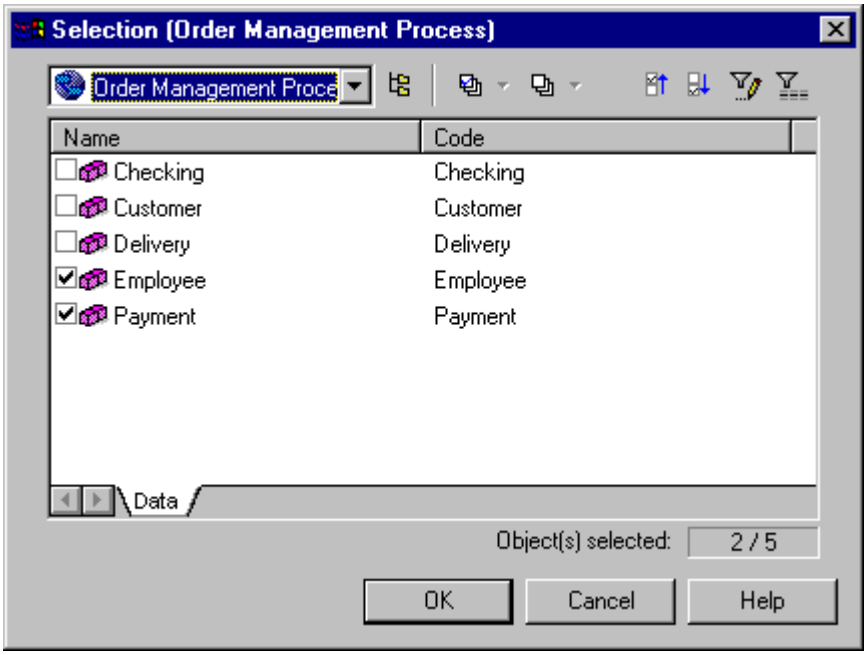
> **Note**
>
> A PDM reference between two tables is imported as a parent/child relationship between the two imported data. Foreign key column objects are not imported because they are created by the PDM reference between tables.

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

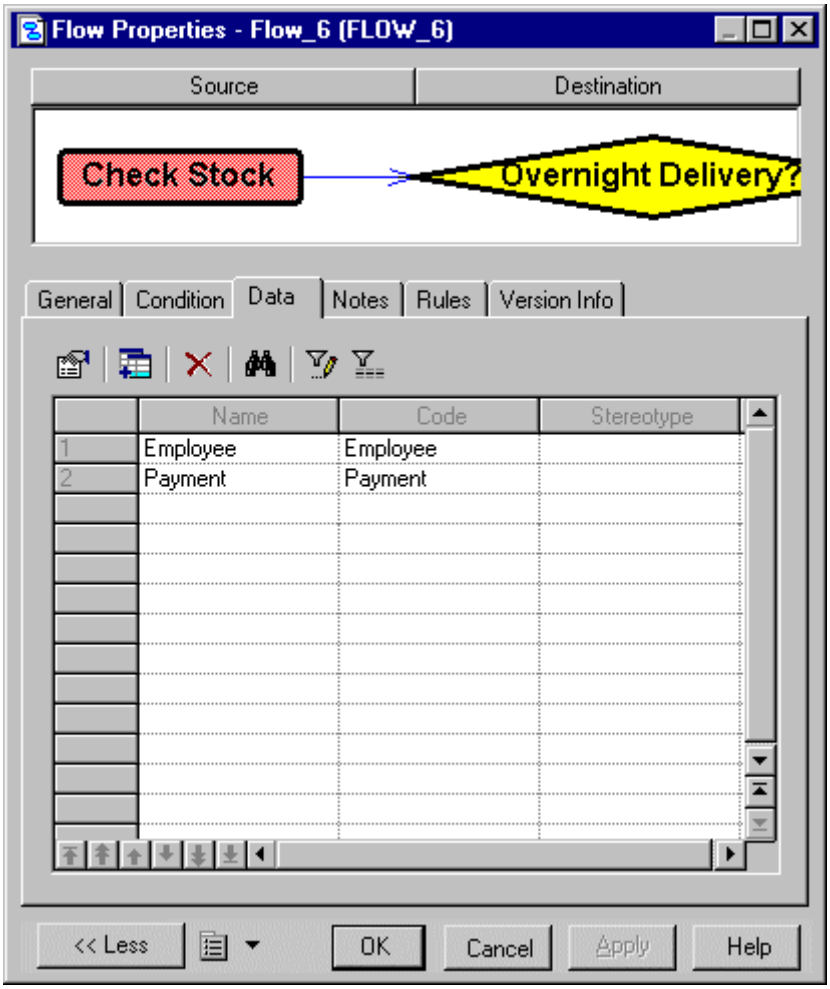
You can specify the data conveyed by flows, resource flows and message formats from the Data tab of their property sheets.

Procedure

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.

Results

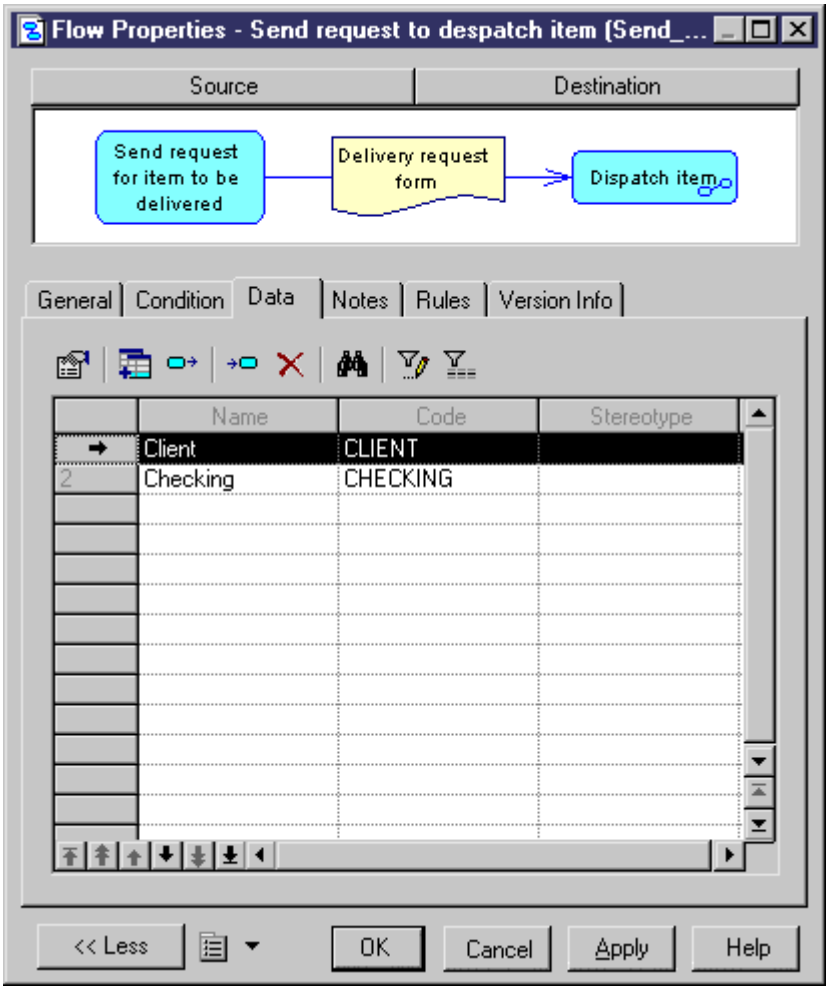
> **Note**
>
> You can display the list of data conveyed by a flow or a resource flow on its symbol by selecting *Tools* > *Display Preferences* > *Flow* (or Resource Flow) and selecting the Data List radio button.

2.10.5 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.

Procedure

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.

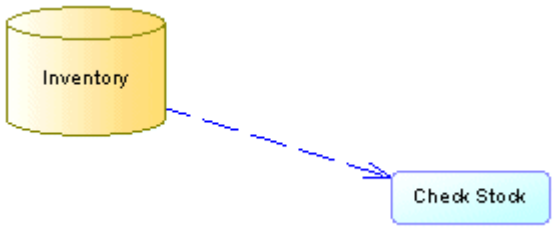
   A message box is displayed to indicate to which process the data was migrated. Data migrated to a source process is created with a "Create" access type, while data migrated to a destination process is created with a "Read" access type.
3. Click OK to close the property sheet and return to the model.

2.11 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.

Resources can be created in models targeting the Analysis, Data Flow Diagram, or BPMN languages.

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



You cannot create shortcuts to a resource.

2.11.1 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* > *Modeling with PowerDesigner* > *Objects*.

2.11.2 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:

Table 27:

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.
Number ID	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. 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 Data Flow Diagram (DFD) [page 103]).
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

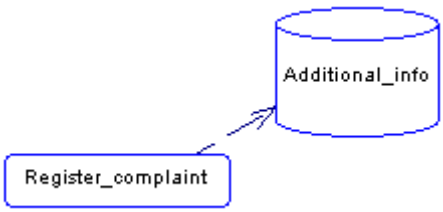
The following tabs are also available:

- *Data* - [Analysis and DFD only] Lists the data associated with the resource. Data come from the input and output resource flows (see Resource Flows (BPM) [page 67]) .

2.11.3 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:



Parallel flows, two flows between the same source and destination objects are allowed. *Reflexive flows* with the same source and destination are allowed on processes.

> **Note**
>
> A resource flow cannot link shortcuts of processes or resources.

Creating a Resource Flow

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

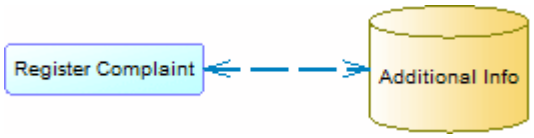
- Use the *Flow/Resource Flow* tool in the Toolbox.
- Select *Model* > *Resource Flows* to access the List of Resource Flows, and click the *Add a Row* tool.
- Right-click the model (or a package) in the Browser, and select *New* > *Resource Flow*.

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.

Table 28:

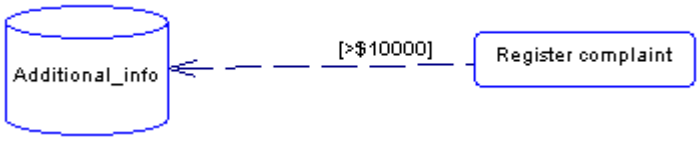
Property	Description
Name/Code/ Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide 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	[Analysis only] Specifies the format of the data exchanged between the process and the resource. You can choose from the following values: • None – no exchange of data. • Undefined [default] – Click the Create tool to the right of the list to create a message format (see Message Formats (BPM) [page 50]).

Property	Description
Access mode	Specifies the way to access data in a resource, and thus defines the resource flow direction. You can select one or more of the following: • Read – from resource to process. • Create, Update, Delete – from process to resource. If you select Read and another access mode, the resource flow symbol is bi-directional: 

The following tabs are also available:

- *Condition* - Defines the nature of the condition attached to a resource flow, and contains the following properties:

  Table 29:

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

- *Data* - 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 [page 63]).

3 Service Oriented Architecture BPM (SOA)

SOA is a process language that allows you to design the orchestration of your processes by Web services without specifying any particular platform or language. You cannot generate from SOA, but you can import WSDL files to populate your model with service providers.

Having developed your model, you can select *Tools* > *Generate Business Process Model* to generate a model targeting another process language.

> **Note**
>
> The SOA process language is very close to BPEL4WS (see BPEL4WS 1.1 and WS-BPEL 2.0 [page 158]), except that it supports attaching any type of operation to a process (BPEL4WS only supports attaching One-Way and Request-Response operations) and does not allow the definition of correlation keys for sent messages.

3.1 Business Process Diagrams (SOA)

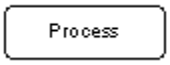
Business process diagrams for SOA and other executable languages contain additional objects that allow you to model the implementation of processes through Web services modeled as service providers.

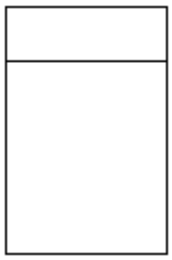
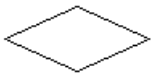
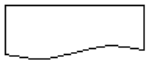
A business process diagram can be created in a model, a package, or within a decomposed process.

> **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*.

PowerDesigner supports all the objects necessary to build executable business process diagrams:

Table 30:

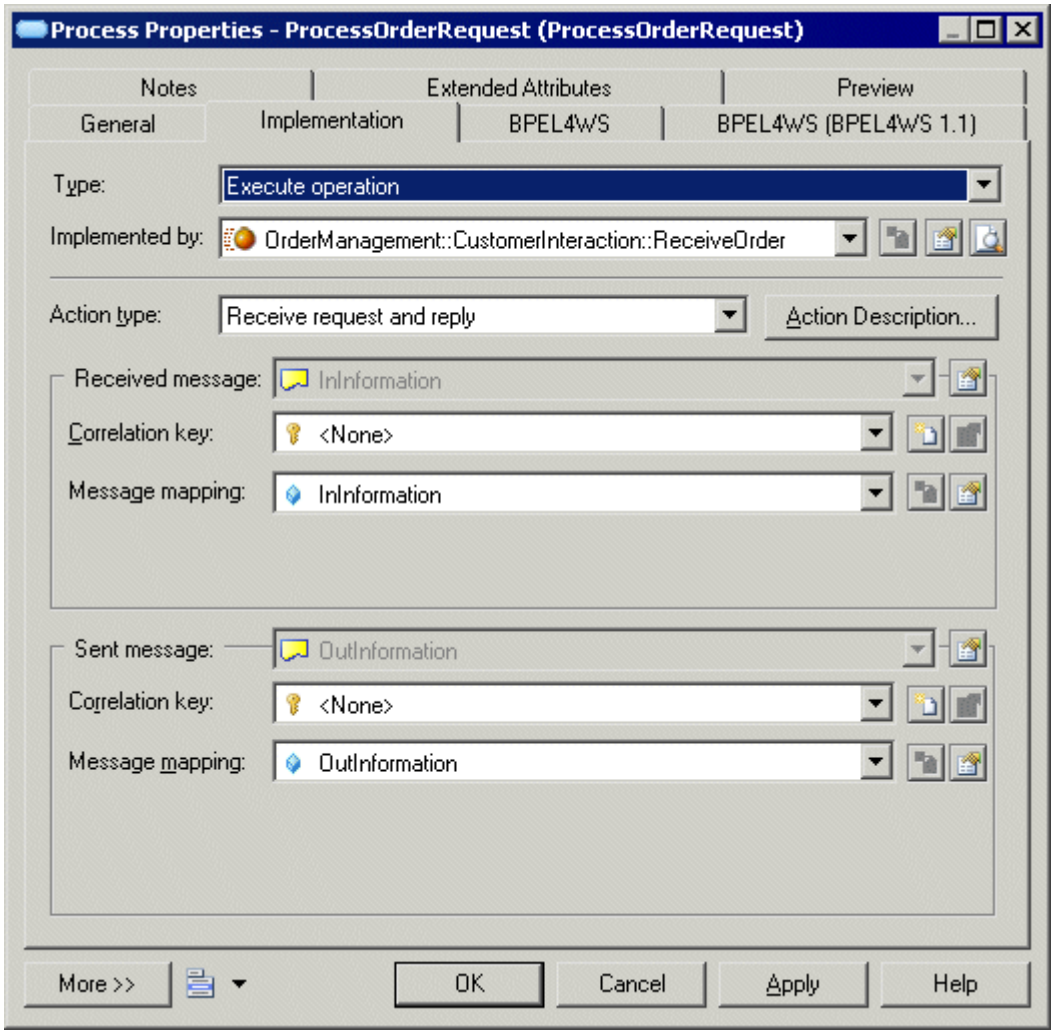
Object	Tool	Symbol	Description
Process			Task to perform (see Processes (BPM) [page 23]).

Object	Tool	Symbol	Description
Organization unit			Organization, service or person that is responsible for a process (see Organization Units (BPM) [page 33]).
Flow			Path of the control flow between processes (see Flows (BPM) [page 47]).
Decision			Decision to take when several flow paths are possible. Only one path will be triggered at execution time (see Decisions (BPM) [page 43]).
Synchronization			Enables synchronization of flows between two or more concurrent actions or allows the design of a split (see Synchronizations (BPM) [page 45]).
Start			Starting point of the processes described in the choreography diagram (see Starts and Ends (BPM) [page 41]).
End			Termination point of the processes described in the choreography diagram (see Starts and Ends (BPM) [page 41]).
Message format	None		Format definition of data exchanged between processes (see Message Formats (BPM) [page 50]).
Message part	None	None	Portion of the WSDL (Web Services Description Language) message (see Message Parts (BPM) [page 52]).
Events	None	None	Instantaneous and observable occurrence during the course of a business process (see Events (BPM) [page 74]).
Service provider	None	None	Web service containing interfaces and operations to implement your processes (see Service Providers (BPM) [page 78]).
XSD document	None	None	Contains the data schema handled by a service provider (see XSD Documents (BPM) [page 88]).

Object	Tool	Symbol	Description
Variable	None	None	Data container (see Variables (BPM) [page 96]).
Correlation key	None	None	Set of variables used to identify a process instance (see Correlation Keys (BPM) [page 97]).
Data transformation	None	None	Object that allows the copy of data from a source to a target with potential transformations (see Data Transformations (BPM) [page 99]).

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

- Import a WSDL file to obtain the Web services that will implement your processes as service providers (see Importing a Service Provider from a WSDL File [page 81]).
- Define service provider operations to implement your processes (see Linking an Operation to a Process [page 93]).
- Model breaks in the normal flow of processes using events (see Events (BPM) [page 74]), catch events using event handlers (see Event Handlers [page 77]) or generate events from processes (see Process Properties [page 24]).
- Perform transformations on data (see Implementing Processes [page 26]).
- Specify input and output messages on processes to specify data exchange between partners (see Linking an Operation to a Process [page 93]). 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**
>
> No data is specified on flows between processes when modeling with execution languages.

3.2 Process Service Diagrams (SOA)

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

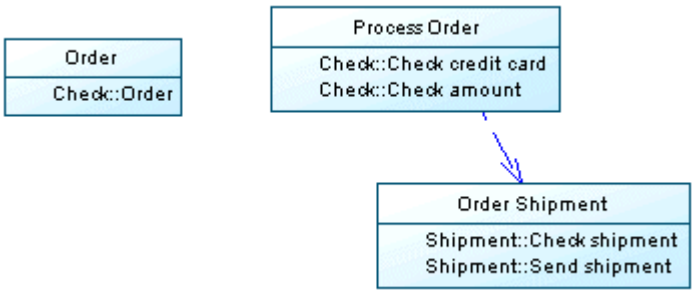
> **Note**
>
> To create a process service diagram in an existing BPM, right-click the model in the Browser and select *New* > *Process Service Diagram*. You cannot create a new BPM with a process service diagram as the first diagram.

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

Table 31:

Object	Tool	Symbol	Description
Service provider			Service containing interfaces and operations. See Service Providers (BPM) [page 78].
Traceability Link			Unidirectional link between two service providers to specify a dependency (documentation purposes only).

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



3.3 Events (BPM)

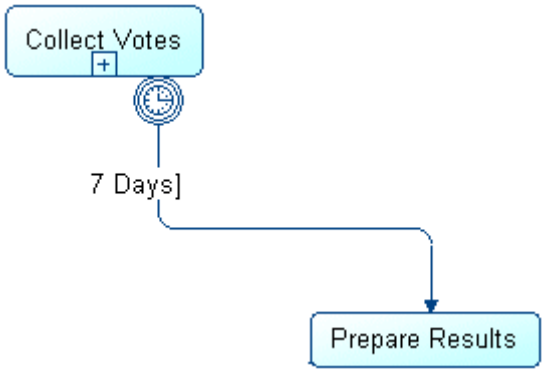
An *event* is an instantaneous and observable occurrence during the course of a business process, which triggers it to respond. For example, it can be unexpected data returned by a web service or a deadline.

Events can be created in models targeting the SOA, BPMN, and BPEL languages. An event can be associated with a:

- Flow – to be caught and handled using an event handler (see Event Handlers [page 77]).
- Process with a Generate event implementation type – to trigger an event (see Implementing Processes [page 26]).

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:

3.3.1 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* > *Modeling with PowerDesigner* > *Objects*.

3.3.2 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:

Table 32:

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. An event has the following predefined stereotypes: • `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:

Table 33:

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 [page 24]).

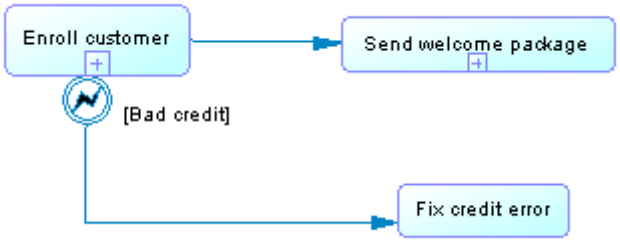
3.3.3 Event Handlers

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

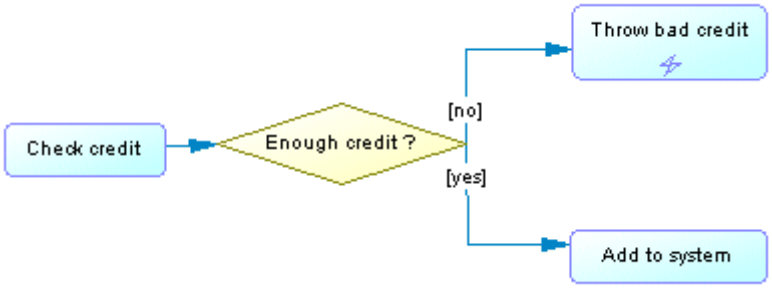
Context

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:



Procedure

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

Results

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.

3.4 Service Providers (BPM)

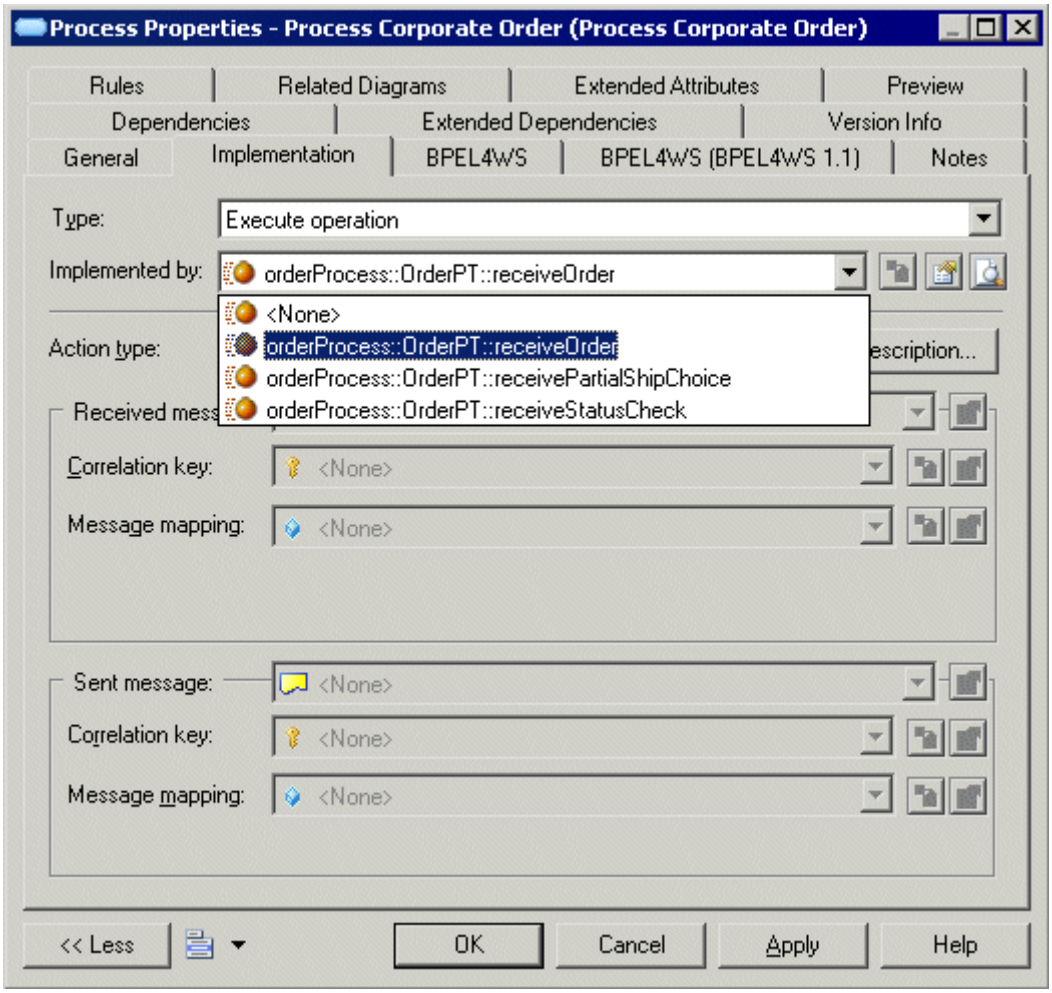
To invoke a Web service, you need the WSDL from the service, which describes the port, service name, operations, and messages that the process needs to communicate with the service. Web services are modeled in PowerDesigner as service providers.

Service providers can be created in models targeting the SOA, BPMN, and BPEL languages.

A service provider contains interfaces (see Service Interfaces (BPM) [page 86]), which in turn contain operations (see Operations (BPM) [page 89]), which can implement processes (see Linking an Operation to a Process [page 93]). 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 Other Models [page 84]). The service provider has no graphical symbol in the diagram.

> **Note**
>
> Service providers can be displayed with the interfaces and operations they contain in process service diagrams, linked with traceability links to show dependencies (see Process Service Diagrams (SOA) [page 73]).

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:



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

3.4.1 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* > *Modeling with PowerDesigner* > *Objects*.

3.4.2 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:

Table 34:

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.
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.

The following tabs are also available:

- *Service Interfaces* - Lists the interfaces contained by the provider (see Service Interfaces (BPM) [page 86]).
- *XSD Documents* - Lists the XSD documents defining the data schemas that describe the service provider (see XSD Documents (BPM) [page 88]).
- *XML Namespaces* - Lists the XML namespace prefixes used by the WSDL file, specifying the xmlns namespace declaration, an optional shorthand prefix, and the namespaceURI (Uniform Resource Identifier),

which specifes the location where element and attribute names are declared. The namespace declaration syntax is the following:

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

- *Data Schema* - Contains the data schema of the service provider, which can be created manually or come from the imported WSDL or an XSD document associated with an XML model. If there are more than one XSD document, data schemas are concatenated. You can specify the type of schema as DTD, XML Schema, or RELAX NG.

3.4.3 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).

Context

Then, you can proceed to the implementation of your processes using operations and associated messages (see Implementing Processes [page 26]).

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:

Table 35:

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

Procedure

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 [page 82]).

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.
4. [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.
5. 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* > *Modeling with PowerDesigner* > *Comparing and Merging Models*.
6. Click OK to return to the model.

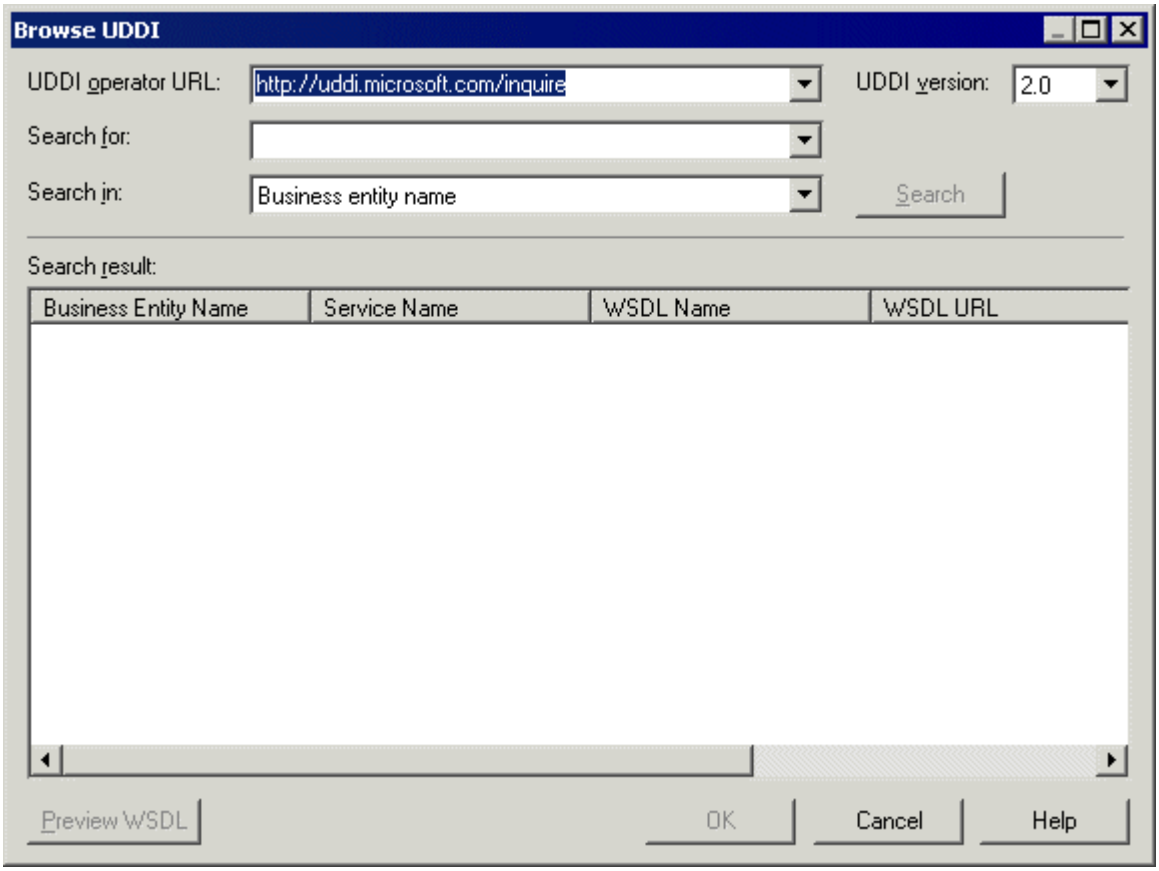
   The imported elements are added to your model, are visible in the Browser, and in the Reverse tab of the Output window. If you have selected the Create XML Model option, the XML model(s) corresponding to the WSDL schema(s) are also created in the workspace.

3.4.4 Browsing for a WSDL File on a UDDI Server

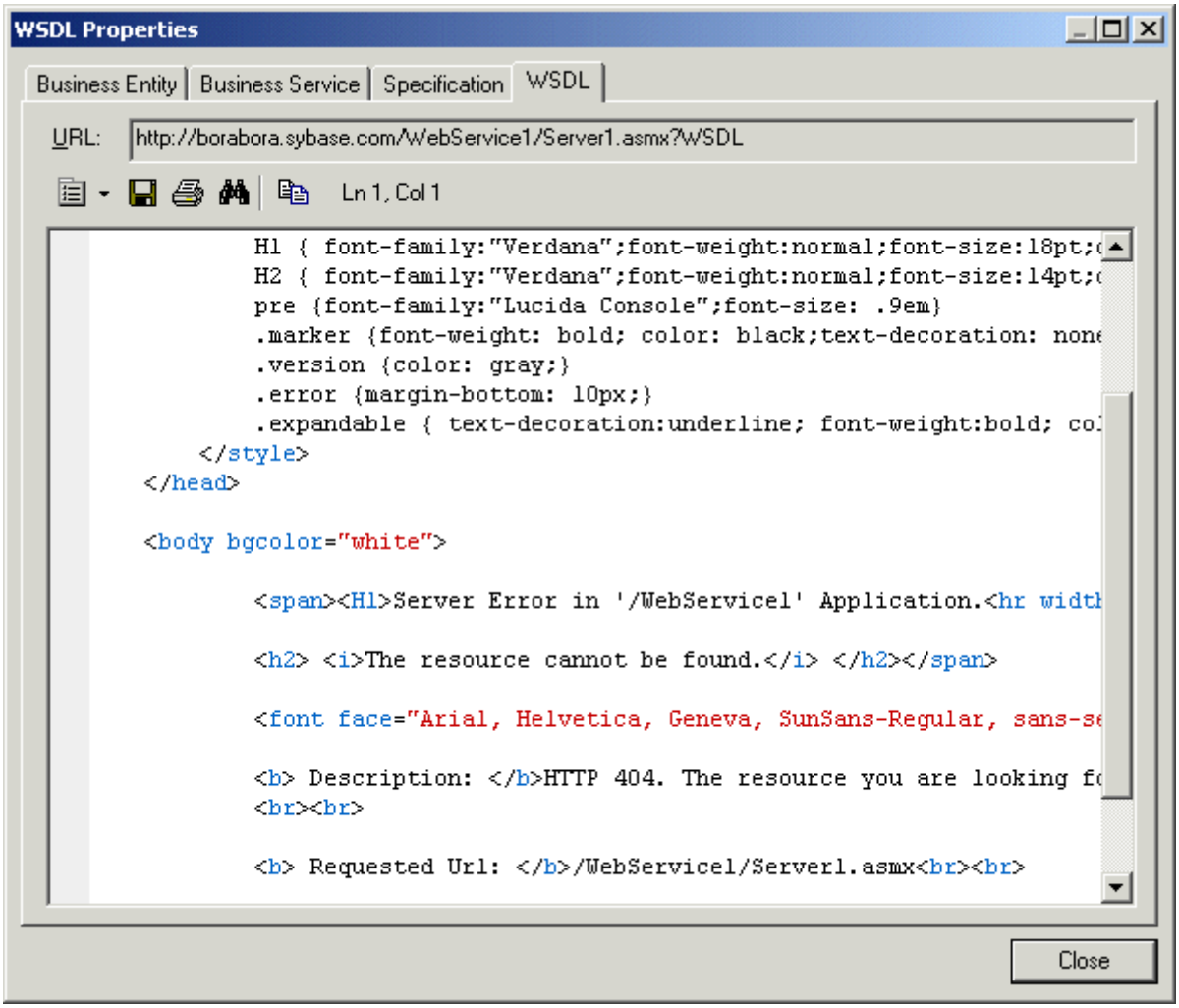
You can browse for a WSDL on a UDDI server. You need to have Internet Explorer 5 or higher to use the Browse UDDI feature.

Procedure

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



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.

3.4.5 Importing and Exporting Service Providers From/To Other Models

You can link the abstract definition of a service interface and its operations in a BPM with a concrete implementation through an OOM component or a PDM database web service. You can initialize a BPM with an existing implementation defined in an OOM or a PDM or export a requirement analysis in a BPM to initialize an OOM for the implementation.

Objects are imported or exported as follows and are referenced in the *Implementation* field of the BPM object property sheet:

Table 36:

OOM objects	BPM objects	PDM Objects (import to BPM only)
Web service, EJB, or other component (see *Object-Oriented Modeling > Building OOMs > Implementation Diagrams > Components (OOM)*)	Service provider	Web service using the SOAP protocol (see *Data Modeling > Building Data Models > Physical Diagrams > Web Services*)
Web Service implementation class, or a UML interface associated with the component	Service interface	-
Class (or interface) operation	Operation	Web operation
SOAP Input value (Input box content)	Operation Input message	-
InputSoapMessageName extended attribute	Operation Input name	-
SOAP Input schema text	Input message format text	-

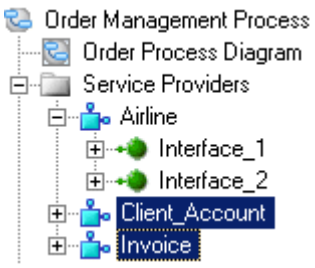
Importing a Service Provider from an OOM or PDM

The Service Provider Import Wizard is available when an implementation OOM 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.

   > **Note**
   >
   > To import from a PDM, your Web service must be generated and deployed to the database, and the database server must be running in order to have a WSDL URL.

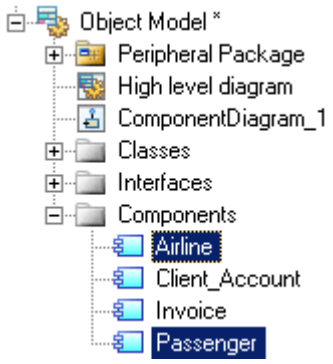
2. Select an OOM or PDM to import from, and click *Next*.
3. Select the OOM components or PDM web services you want to import and click *Finish* to import them to the BPM as service providers:



Exporting a Service Provider to an OOM

The Service Provider Export Wizard is available 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*.
3. Select the service providers you want to export and click *Next*.
4. Select the type of component. If the target OOM language supports web services, the *Web Service* type is selected by default.
5. Click *Finish* to export the service providers to the OOM:



3.4.6 Service Interfaces (BPM)

A *service interface* corresponds to the Port Type object in a WSDL file, belongs to a service provider, and contains a set of operations. For example, a **`LoanApproval`** service interface may contain the **`Request`** and **`Check`** operations.

> **Note**
>
> If you copy a service interface, you also copy its associated operations. Shortcuts for service interfaces are not permitted.

Creating a Service Interface

If you import a service provider, you also import its service interfaces. You can create a service interface manually by using the *Add a Row* tool on the *Interfaces* tab in the property sheet of a service provider, or by right-clicking the service provider in the Browser, and selecting *New* > *Service Interface*.

> **Note**
>
> Select *Model* > *Service Interfaces* to view all the interfaces in the model. You cannot create service interfaces from this list.

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:

Table 37:

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.

The following tabs are also available:

- *Operations* - Lists the operations contained in the service interface (see Operations (BPM) [page 89]).

3.4.7 XSD Documents (BPM)

An *XSD document* defines the data schema handled by a service provider, and can be associated with a PowerDesigner XML model. When reverse engineering or importing web services, you can select the *Create XML Model* option to create an XSD document for each data schema found in the source WSDL.

Creating an XSD Document

You can create an XSD document manually by using the *Add a Row* tool on the *XSD Document* tab in the property sheet of a service provider, or by right-clicking the service provider in the Browser, and selecting *New* > *XSD Document*.

> **Note**
>
> Select *Model* > *XSD Documents* to view all the document in 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:

Table 38:

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.

Property	Description
Schema model	Specifies the 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 or view the properties of the currently selected model. For information about working with XML models, see *XML Modeling*.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

The following tabs are also available:

- *XML NameSpaces* - Displays a list of XML namespace 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 [page 80].
- *Schema* - Specifies the message part definition details. You can enter any appropriate information in this field, and open, insert and save text files. The first lines of the schema display the XML version, encoding format, and namespace details.

3.5 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.

Operations can be created in models targeting the SOA, BPMN, and BPEL languages.

An operation belongs to a service interface which, in turn, belongs to a service provider (see Service Providers (BPM) [page 78]). The operation describes the implementation of an atomic process, and can be sent or received by an activity (see Linking an Operation to a Process [page 93]).

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 Linking an Operation to a Process [page 93]).
- 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.

3.5.1 Creating an Operation

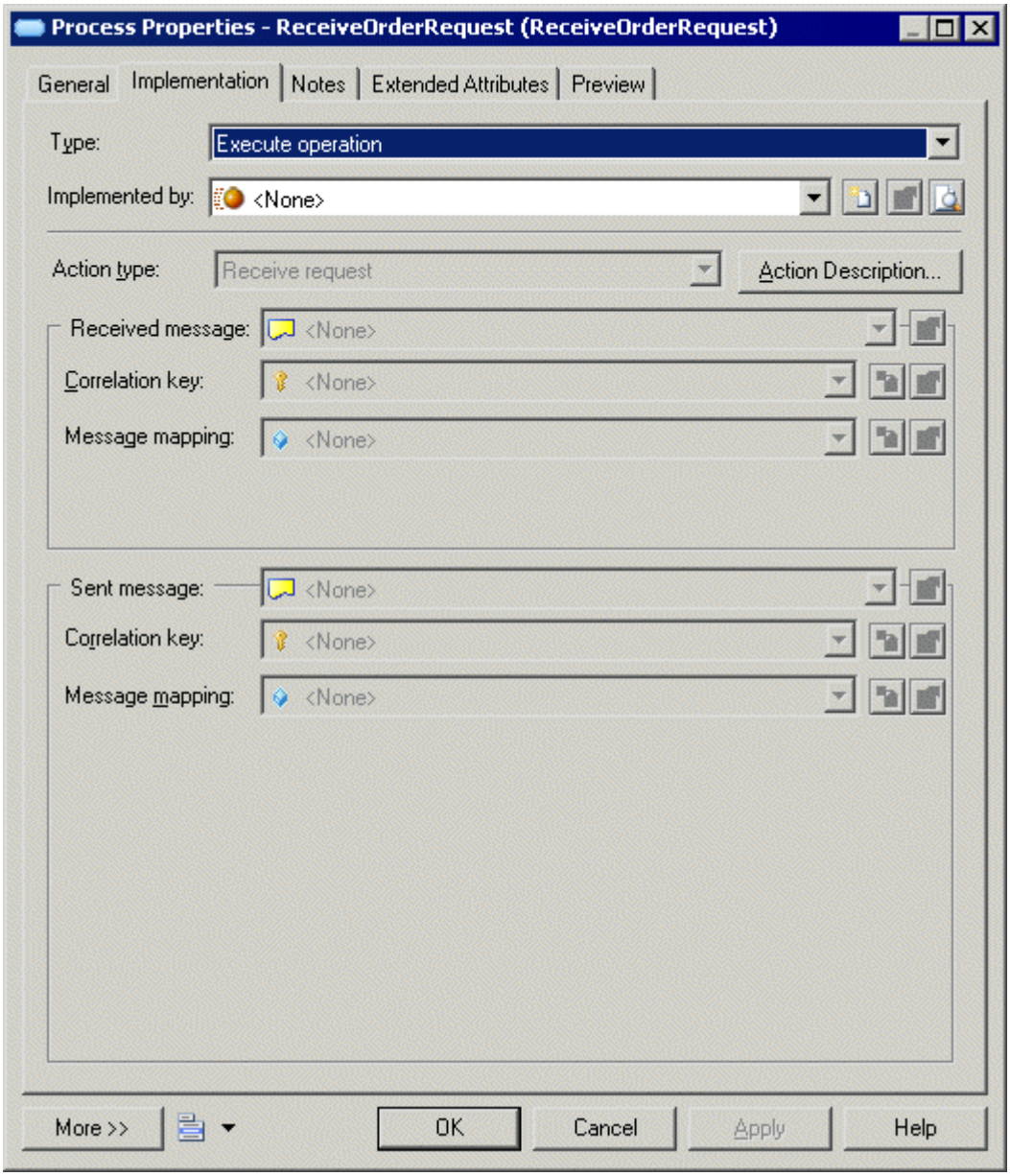
You can create an operation using a Wizard or from the property sheet of, or in the Browser under, a service interface.

Context

- 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:

Procedure

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 an existing service provider from the list or create a new one and specify a name for it, and then click *Next*.
4. Select an existing service interface from the list or create a new one and specify a name for it, and then click *Next*.
5. Click *Finish* to complete the creation and open the new operation property sheet.
6. Enter properties as appropriate, and then click *OK* to close the property sheet and return to the process.

3.5.2 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:

Table 39:

Property	Description
Provider / Interface	[Read-only] Specifies the name of the service provider and its interface that own the operation. Click the *Properties* tools next to these fields to display the properties of these objects.
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.

The following tabs are also available:

- *Input/Output* - Contains the following properties:

  Table 40:

Property	Description
Type	Specifies the type of the operation and how messages will be handled: ○ **`Undefined`** – [default: no message] ○ **`One Way`** – [input message only] The activity calls the web service and no response is expected. ○ **`Request-Response`** – [input and then output messages] The activity calls the web service and a response is expected. Fault messages can also be sent as output in case of error. ○ **`Solicit-Response`** – [output and then input messages] The web service solicits the activity and a response is expected. Fault messages can also be sent as input in case of error. ○ **`Notification`** - [output message only] The web service solicits the activity and no response is expected.

Property	Description
Input / Output message	Specify names and message formats for the input and output messages. 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.

- *Faults* - [**`Request-Response`** and **`Solicit-Response`** operations] 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.

3.5.3 Linking an Operation to a Process

You can link an operation to a process so that the operation is implemented by the process or the process invokes the operation. Both cases are modeled using the **`Execute operation`** type on the *Implementation* tab of the process property sheet. You can only link operations to atomic processes and not decomposed processes.

Context

> **Note**
>
> You can drag an operation from the Browser and drop it into a diagram to automatically create a process that invokes the operation.

Procedure

1. Open the property sheet of the process, click the *Implementation* tab, and select **`Execute operation`** from the *Type* list.
2. Select the operation that will implement the process from the *Implemented by* list or create a new operation by clicking the *Create* tool to the right of the list (see Creating an Operation [page 90]).
3. Select the *Action type* of the process to specify the type of message exchange the activity performs from the following options:
   - **`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.

   You can, optionally, click the *Action Description* button to open a text editor, and enter any appropriate information, as well as open, insert and save text files.

Selecting the action type sets how the messages defined in the operation will be handled by the process. This table summarizes the relationships between the input/output messages of the operation and the received and sent messages of the activity, which depends on both the operation type (see Operation Properties [page 92]) and the process action type:

Table 41:

Operation Type	Execute Operation Process Action Type				
	Activity Received Messages		Activity Sent Messages		
	Receive Request	Receive Request and Reply	Reply	Reply Fault	Invoke Operation
Undefined	Received = None	N/A	N/A	N/A	Sent = None
One-Way	Received = Input	N/A	N/A	N/A	Sent = Input
Request-Response	Received = Input	Received = Input Sent = Output	Sent = Output	Sent = Fault	Received = Output Sent = Input
Solicit-Response	Received = Output	Received = Output Sent = Input	Sent = Input	Sent = Fault	Received = Input Sent = Output
Notification	Received = Output	N/A	N/A	N/A	Sent = Output

4. [optional, when required by the action] Specify properties to control how the *Received message* is handled. You can specify a:
   - *Correlation key* (see Correlation Keys (BPM) [page 97]) - directs a received message to the correct activity instance. Received correlation keys are mostly used for receive request activities.
   - *Message mapping* - (see Variables (BPM) [page 96]) 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.
5. [optional, when required by the action] Specify properties to control how the *Sent message* is handled. You can specify a:
   - *Correlation key* - (see Correlation Keys (BPM) [page 97]) - contains the information that is useful to the partner in the next exchange with the activity.
   - *Message mapping* - (see Variables (BPM) [page 96]) - 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.
6. 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:

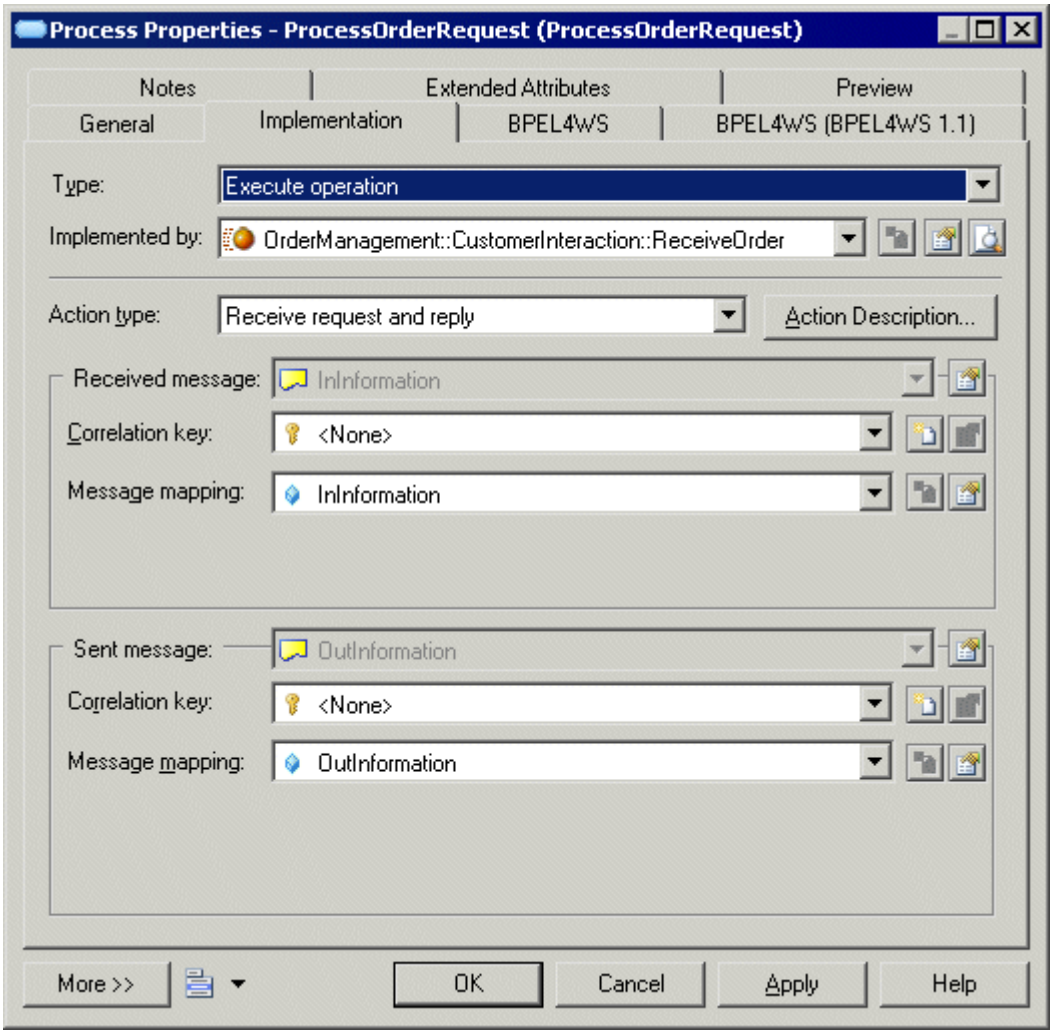
   In our example, we create a service provider called **`OrderManagement`** and an interface called **`CustomerInteraction`**, and then set the following properties for the operation:

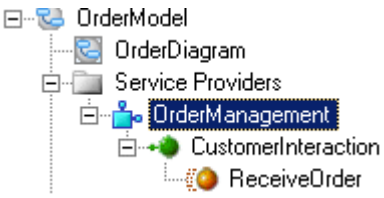
Table 42:

Tab	Property	Value
General	Name	ReceiveOrder
Input/Output	Type	Request-Response

Tab	Property	Value
Input/Output	Input Message Name	OrderRequest
Input/Output	Input Message Message Format	InInformation
Input/Output	Output Message Name	OrderRequestConfirmation
Input/Output	Output Message Message Format	OutInformation

We select the operation in the *Implemented by* field, 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:





3.6 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.

Variables can be created in models targeting the SOA, BPMN, and BPEL languages.

Variables can be used in conjunction with:

- Processes – to build the process messages (see Processes (BPM) [page 23]).
- Correlation keys – to identify a process instance using a set of variables (see Correlation Keys (BPM) [page 97]).
- Data transformations – to copy data from one variable to another (see Data Transformations (BPM) [page 99]).

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.

> **Note**
>
> If you move a variable that is used in its original package to another package in the same model, a shortcut is created in the original package. If you move the variable to another model, a copy of the variable is retained in the original model as external shortcuts are not allowed for variables.

3.6.1 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 a package) in the Browser, and select *New* > *Variable*
- [local variables] Open the property sheet of a process, select the *Local Variables* tab, and click the *Add a Row* tool.

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

3.6.2 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:

Table 43:

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 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.

The following tabs are also available:

- *Value* - Specifies the value that the variable contains. You can enter any appropriate information in the field, as well as open, insert and save text files. 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`.

3.7 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.

Correlation keys can be created in models targeting the SOA, BPMN, and BPEL languages.

A correlation key is associated with a process implemented by an operation (see Operations (BPM) [page 89]). Depending on the operation type, a process (activity) can have one correlation key associated with the input message it receives and/or one associated with the output message it sends.

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

3.7.1 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* > *Modeling with PowerDesigner* > *Objects*.

3.7.2 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:

Table 44:

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.

The following tabs are also available:

- *Variables* - Lists the variables (see Variables (BPM) [page 96]) that define the 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.

3.8 Data Transformations (BPM)

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.

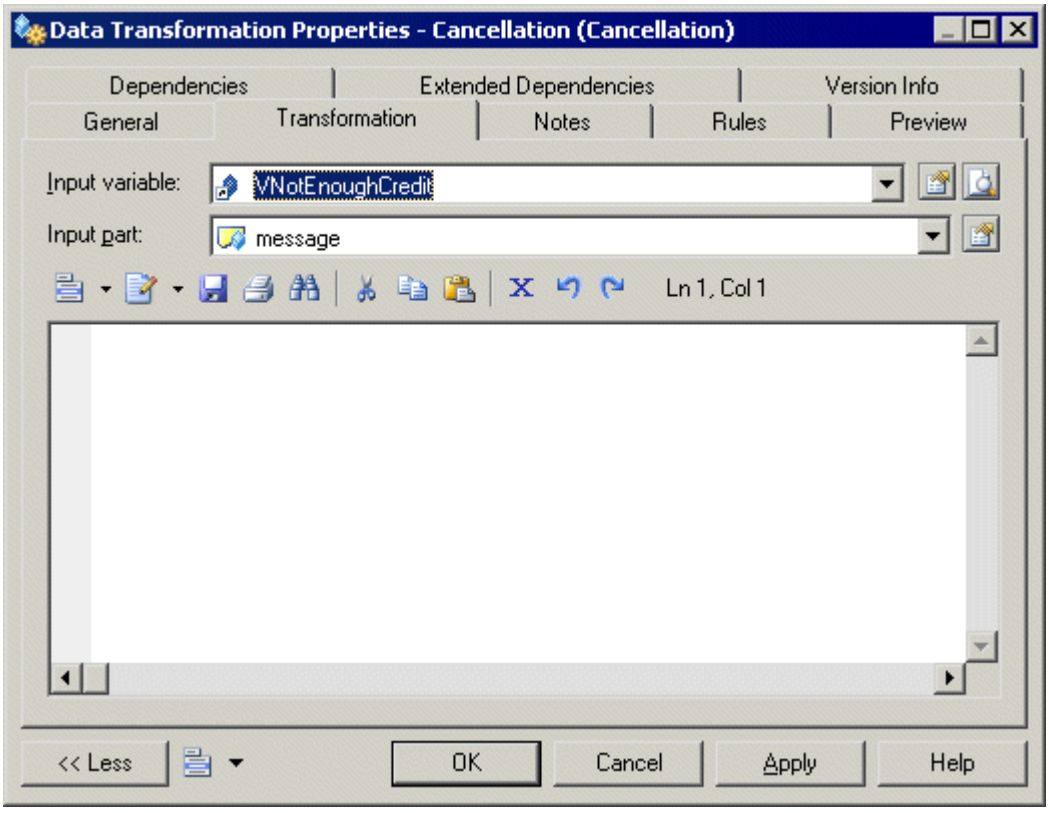
For example, a manufacturer may be asked the price of one of its products, which he calculates using parameters such as quantity, delivery location, and so on. These parameters are input data on which the manufacturer will perform a transformation, and then store the result as target data.

Data transformations can be created in models targeting the SOA and BPEL languages. They can be used in conjunction with:

- Assign activities – to design a sequence of atomic assign tasks (see Process Properties [page 24]).
- 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) [page 97]).

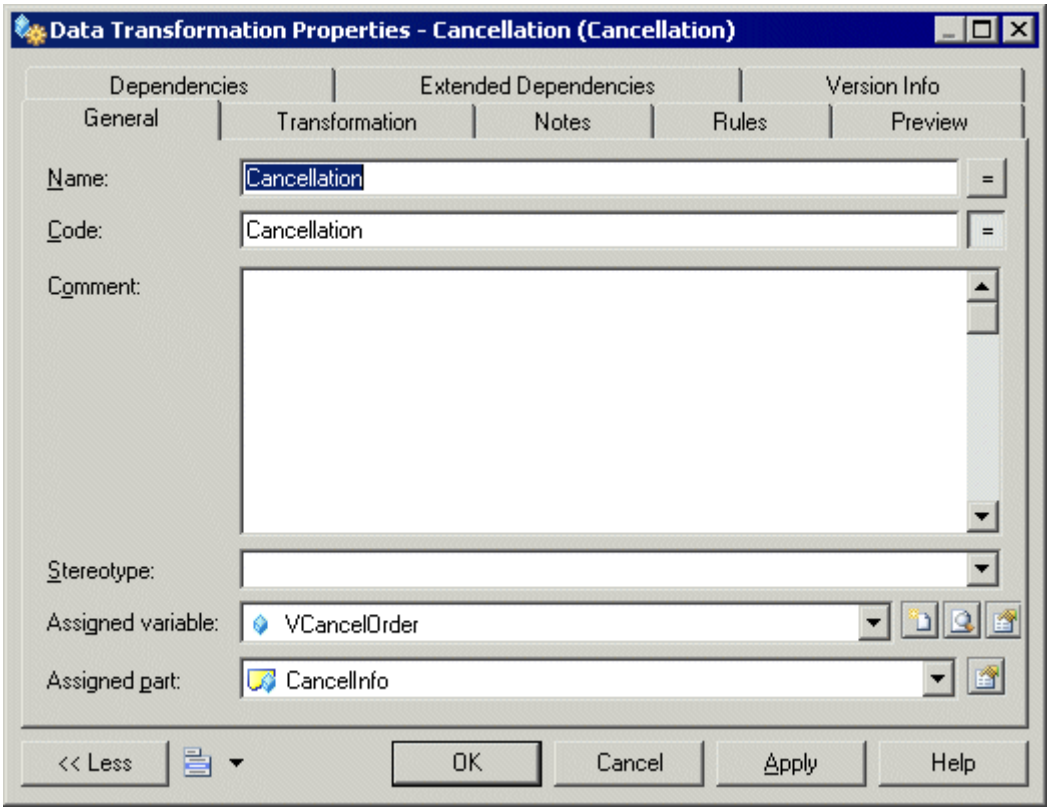
You define a data transformation from its property sheet by selecting:

- On the *Transformation* tab, select an input variable (and, optionally, part) to contain the source to transform and copy. Alternatively, you can leave the *Input variable* field empty and use the text box to write simple transformations using XSLT or more complex transformations using Xpath. In the following example, the `Cancellation` data transformation contains a variable called `VnotEnoughCredit` and a part within that variable called `message`:



- On the *General* tab, select an assigned variable (and, optionally, part) to contain the result of the transformation . In the following example, the `Cancellation` data transformation contains an *Assigned*

variable called `VCancelOrder` and an assigned part (because the source container also has a part) within that variable called `CancelInfo`:



3.8.1 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* > *Modeling with PowerDesigner* > *Objects*.

3.8.2 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:

Table 45:

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 (see Variables (BPM) [page 96]), message format (Message Formats (BPM) [page 50]) or, for BPEL languages, organization unit (see Organization Units (BPM) [page 33]) that receives the result of the transformation. Select an object from the list, click the *Select* tool to browse the available objects, or click the *Create* tool to create a new object.
Assigned Part	[If the assigned variable is set to a message format] Specifies the message part (see Message Parts (BPM) [page 52]) that receives the result of the transformation.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

The following tabs are also available:

- *Transformation* - Contains the following properties:

  Table 46:

Properties	Description
Input variable	Specifies the variable (see Variables (BPM) [page 96]), message format (Message Formats (BPM) [page 50]) or, for BPEL languages, organization unit that sends the variable (see Organization Units (BPM) [page 33]). 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. To specify more than one object as inputs, use the text field and leave the *Input variable* list empty.
Input part	Specifies a source message part when the input variable is typed by a message format. Select an object from the list.

Properties	Description
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.

4 Data Flow Diagram (DFD)

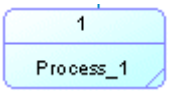
A *data flow diagram* (DFD) is a graphical representation of the flow of data through an information system without any indication of time. DFDs are commonly used to provide an initial top-down analysis of a system, identifying the processes to be carried out and the interactions and data exchanges between them. DFDs can be either *logical*, providing an implementation-independent description of the system, or *physical* describing the actual entities (devices, department, people, etc.) involved.

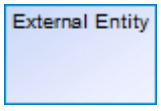
Data flow diagrams can be created in a model targeting the Data Flow Diagram process language. To create a new DFD model, select *File* > *New Model*, choose Business Process Model as the model type, Data Flow Diagram as the process language, and Business Process Diagram as the first diagram.

> **Note**
>
> To create a DFD model from an analysis or other BPM model, select *Tools* > *Generate Business Process Model* and select Data Flow Diagram as the process language. Composite processes with start and end objects are replaced with external entity objects or process shortcuts, depending on the input and output flows of the composite process. Data items are preserved.

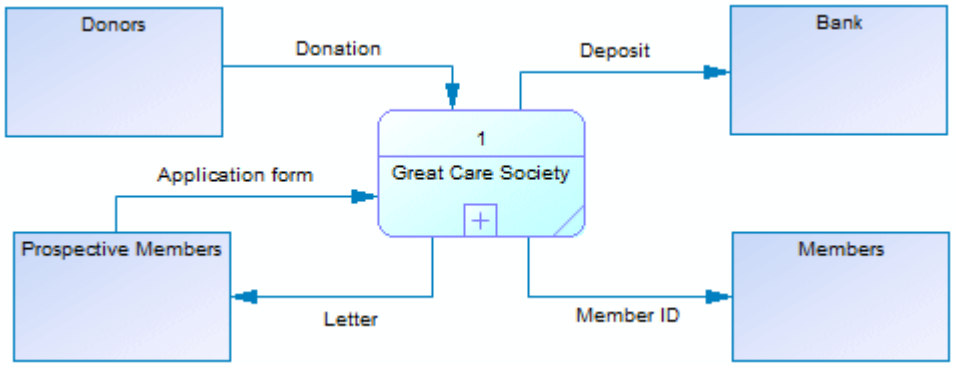
PowerDesigner supports all the objects necessary to build data flow diagrams:

Table 47:

Tool	Symbol	Description
	Gane & Sarson:  Yourdon: 	Process - An activity that transforms or manipulates input data to produce output data (see Processes (BPM) [page 23]). Flows from processes can go to external entities, data stores, split/merges, or other processes. The *Data* tab in the process property sheet displays the process CRUD accesses to data. **Note** To choose a notation, select *Tools* > *Model Options* and select the appropriate Data Flow Diagram Notation.
		Flow - Oriented link that conveys data between processes, external entities, and data stores and represents data in motion. Flows are based on standard flows (see Flows (BPM) [page 47]) with a `Flow` stereotype. Flows to or from data stores are based on standard resource flows (see Resource Flows (BPM) [page 67]), and must be created with the *Resource Flow* tool. A flow cannot directly link two data stores or two external entities. The *Data* tab in the flow property sheet displays the data transported by the flow.

Tool	Symbol	Description
	Gane & Sarson:  Yourdon: 	Data store - Location where data resides permanently or temporarily and represents data at rest. Data stores respond to requests for storing and accessing data, but cannot initiate any actions, and are based on standard resources (see Resources (BPM) [page 66]) with a `Data Store` stereotype. Flows to data stores represent write, update, or delete access, and flows from data stores represent read access.
		External entity - A person, organization, or system outside the system being modeled that sends data to or receives data from the system. Flows from external entities cannot directly access data and must pass through processes. External entities are based on standard organization units (Organization Units (BPM) [page 33]) with an `External Entity` stereotype.
		Split/Merge - Splits a flow into two or more flows or merges multiple flows into a single flow. Split/merges are based on standard synchronizations (see Synchronizations (BPM) [page 45]) with a `Split/Merge` stereotype. A split/merge can, for example, split a complex packet of data into more elementary packets, and send them to different processes or duplicate data to send to different processes, or alternatively merge multiple data packets together for onward transmission.
None	None	Data - Conceptual information exchanged between the other objects (see Data (BPM) [page 54]) You can associate the data analyzed in a DFD with conceptual, logical, and physical data models and object-oriented models (see Linking Data with Other Model Objects [page 56]).

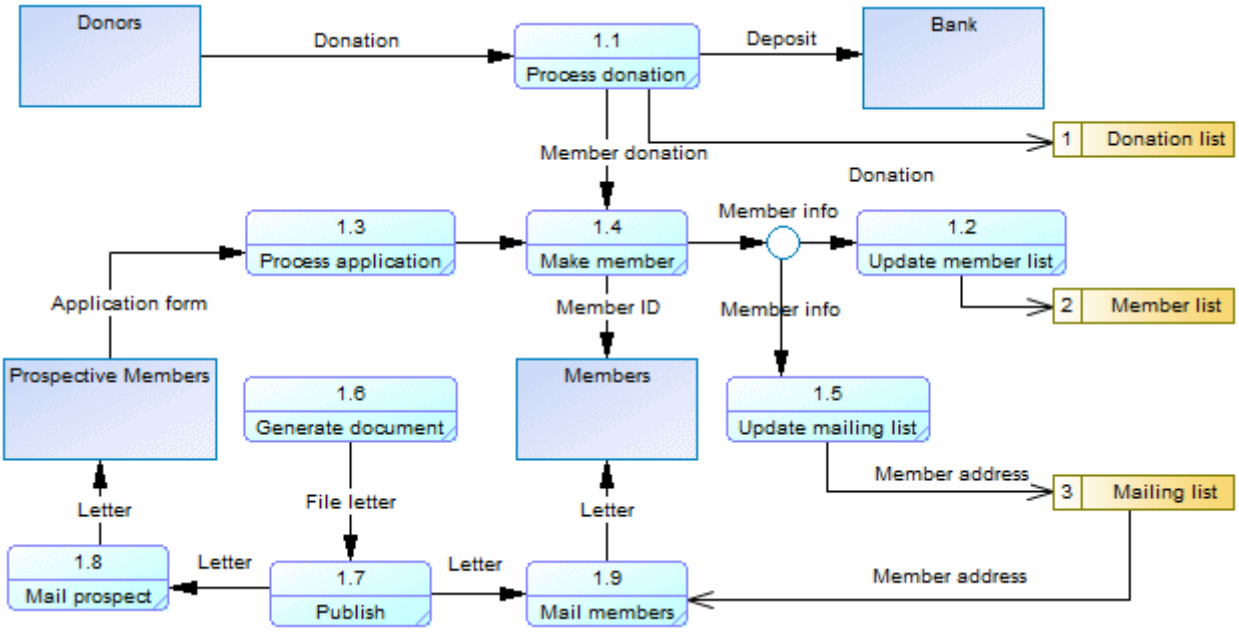
Analysts typically begin with a system context DFD to show the interactions between the system as a whole and external entities. In the following example, the **`Great Care Society`** process interacts with the **`Donors`**, **`Bank`**, **`Members`**, and **`Prospective Members`** external entities:



You can decompose processes in a DFD (see Decomposing Processes [page 27]), and PowerDesigner initializes the subdiagram with all the objects that link to the process being decomposed (external entities, data stores, and

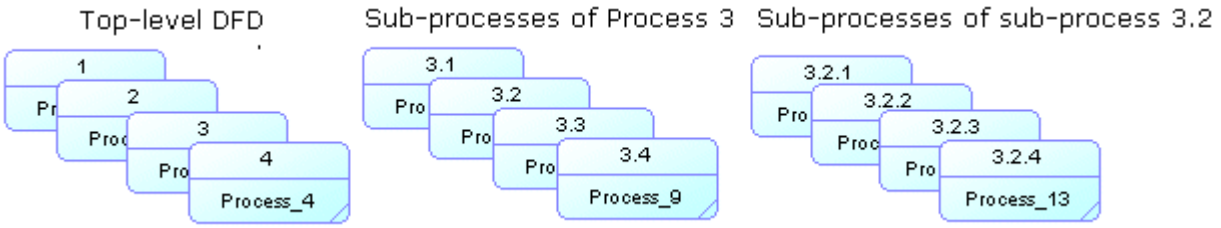
shortcuts to processes as necessary). Such *balancing* helps ensure that all the flows to and from the process being decomposed are preserved at the next level of decomposition. When the `Great Care Society` process is decomposed its diagram is initialized with the four external entities:

The diagram is enriched to show the sub-processes that handle donations, deposits, and correspondence, and the data stores that they interact with:



To test balancing, select *Tools* > *Check Model*, and select the balancing checks under the Flow and Resource Flow objects.

When using the default Gane & Sarson methodology, processes and data stores are automatically numbered. As you decompose your processes, child processes inherit the number ID of their parent process so that the first child of top level process 1 is numbered 1.1, and the first child of this process is numbered 1.1.1. This numbering scheme enables you to easily identify the lineage of a process at any level of decomposition and provides a convenient way to reference processes and data stores by numbers instead of sometimes long or complex names:



Note

Numbering does not indicate the order in which processes are performed. By default, numbers begin at 1, but you can enter any integer in a process or data store property sheet and all subsequent objects will be incremented from that number. To revert to numbering from one, right-click the diagram background and select *Renumber Process IDs* or *Renumber Data Store IDs*.

5 SAP Solution Manager

SAP® Solution Manager is an environment for managing and monitoring business processes. PowerDesigner supports round-trip importing and exporting of SAP® Solution Manager v7.1 SP08 projects. A Solution Manager project is imported as a BPM model.

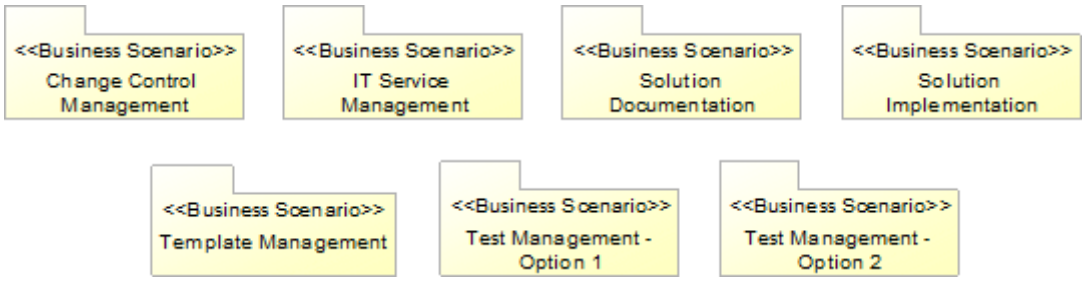
PowerDesigner supports the import, creation, modification, and export of the following types of Solution Manager objects:

- Business scenarios
- Business processes
- Process steps
- Gateways
- Start, intermediate, and end events
- Flows
- Project documentation of type URL
- Annotations and connections

PowerDesigner supports the import of the following objects and their attachment to scenarios and processes, but does not support their creation or the export of modifications to them:

- Logical components
- Organization units
- Master data
- General and non-URL project documentation
- Transactions

A Solution Manager project is imported as a BPM model. It contains a top-level diagram, which shows the business scenarios it contains:



The following tools are available in a business project diagram:

Table 48:

Tool	Description
	Business Scenario - See Business Scenarios (Solution Manager) [page 108].

> **Note**
>
> Though you should not create logical components, organization units, master data, or general or non-URL project documentation in PowerDesigner, you can add existing instances of these objects to your diagram by dragging and dropping them from the Browser.

Projects have the properties of standard BPMs (see) and also specify the person responsable for the project, as well as its language and type, and the path to the archive model used for determining model changes when updating Solution Manager.

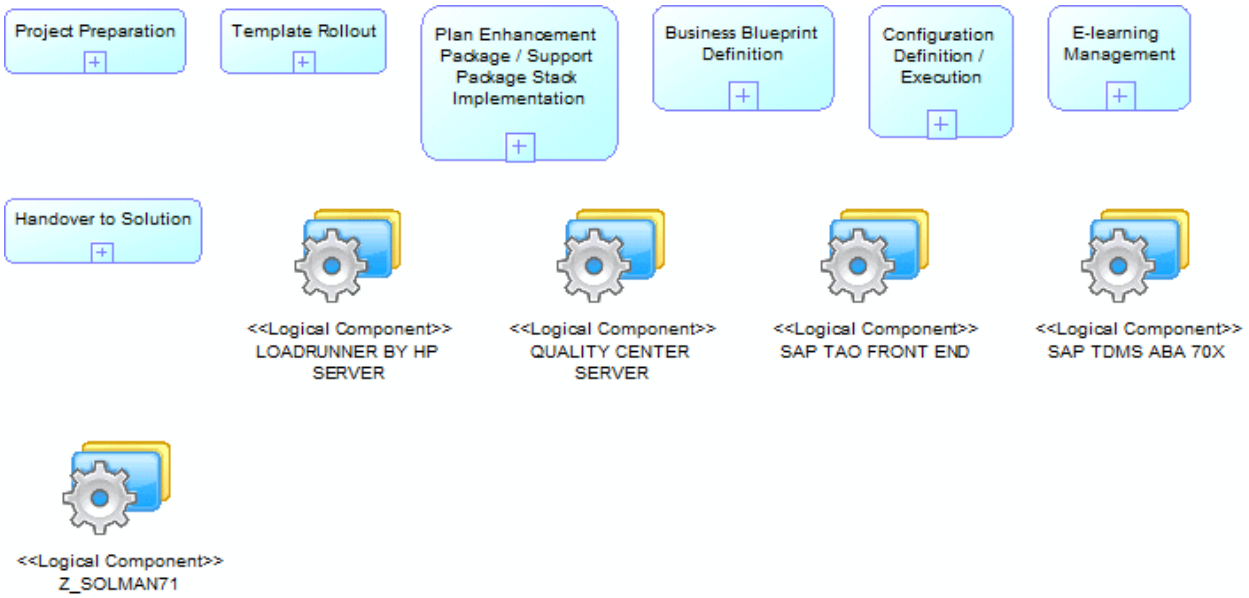
The following additional tabs are also available:

- *General Documentation*, and *Project Documentation* - List the documentation (see General and Project Documentation (Solution Manager) [page 113]) associated with the project.

5.1 Business Scenarios (Solution Manager)

A Solution Manager business scenario contains a set of processes that define a business task in a comprehensive and self-contained manner. A business scenario is imported as a package with the `BusinessScenario` stereotype

Each business scenario contains a diagram, which shows the business processes it contains:



The following tools are available in a business scenario diagram:

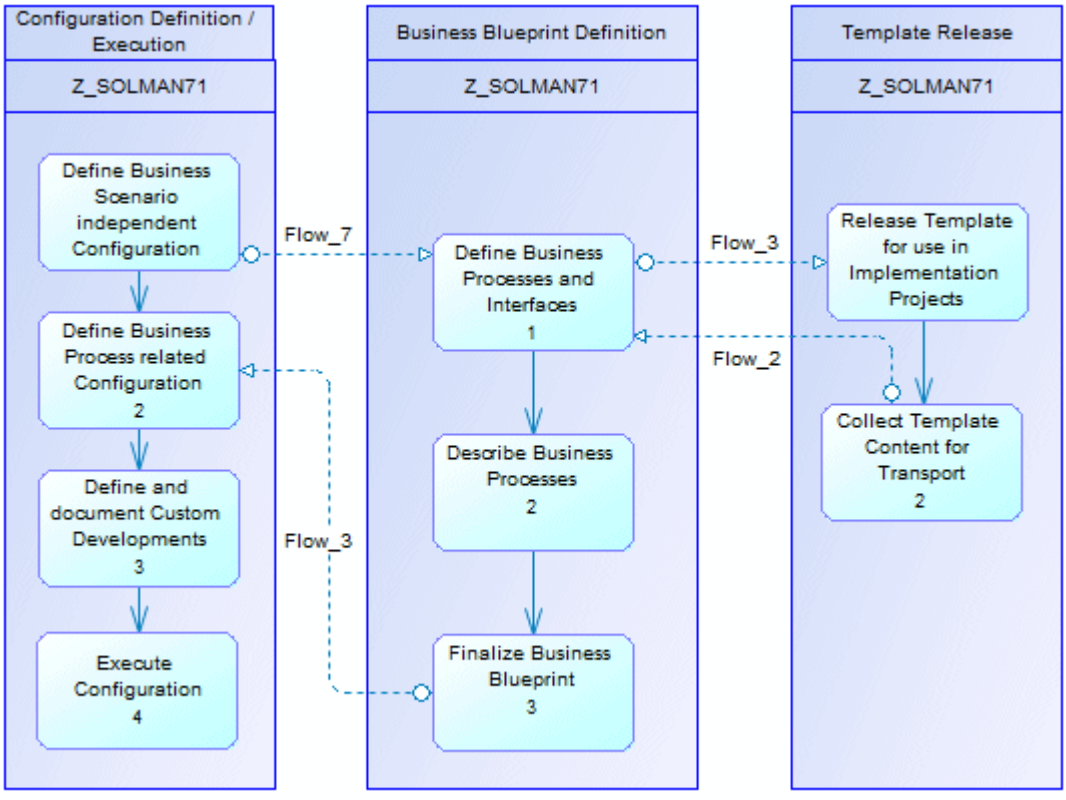
Table 49:

Tool	Description
	Business Process - See Business Processes (Solution Manager) [page 110].

> **Note**
>
> Though you should not create logical components, organization units, master data, or general or non-URL project documentation in PowerDesigner, you can add existing instances of these objects to your diagram by dragging and dropping them from the Browser.

A scenario may also include a scenario flow diagram, which can show the processes in the scenario as swimming pools, in which you can draw message flows between steps in different processes. To prepare your scenario flow diagram:

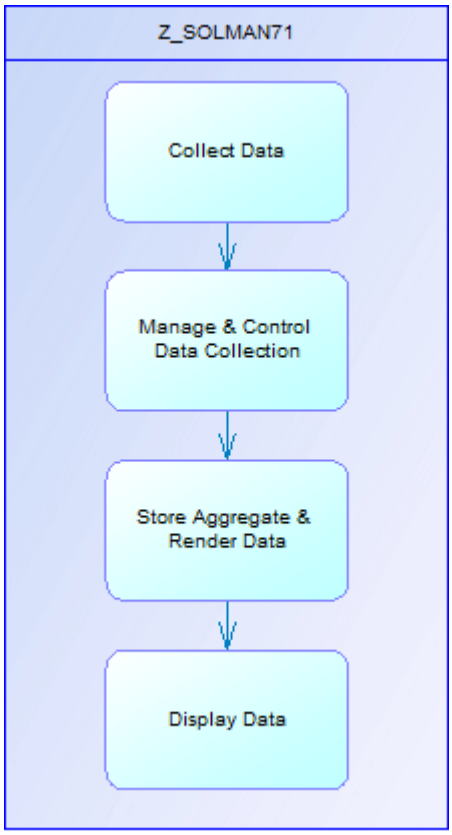
1. If you have not imported a scenario flow diagram for the scenario from the server, then right-click the scenario and select *Show Scenario Flow Diagram* to display it.
2. Drag and drop processes which must exchange messages from the Browser to the diagram.
3. Right-click the scenario and select *Update Scenario Flow Diagram Layout* to show the steps and other sub-objects in the process symbols.
4. Draw messages between process steps as necessary, using the *Flow* tool:



5.2 Business Processes (Solution Manager)

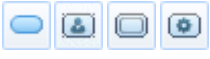
A Solution Manager business process contains a set of logically related activities (process steps) performed to achieve a business outcome. A process step is an elementary activity performed to accomplish a process, which is carried out by a user or a system and runs in only one software component (logical component). Both business processes and process steps are based on standard BPM processes.

Each business process contains a diagram, which shows the process flow passing from step to step, with each step located in the swimlane of the logical component it requires:



The following tools are available in a business process diagram:

Table 50:

Tool	Description
	Process Step, Human Process (activity with user input), SubProcess (references and executes another process), or Automated Process (system activity) - the activity or work being performed.
	Gateway, Parallel Gateway (AND), or Exclusive Gateway (OR) - splits or merges the sequence flow.

Tool	Description
	Start Event (the start of a process flow), Intermediate Event (a message is sent or received in the process), and End Event (the end of the process flow).
	Flow
	Annotations and Connections

> **Note**
>
> Though you should not create logical components, organization units, master data, or project or general documentation in PowerDesigner, you can add existing instances of these objects to your diagram by dragging and dropping them from the Browser.

Process and Process Step Properties

Business processes and process steps have the properties of standard processes (see Processes (BPM) [page 23]), and the following additional properties:

Table 51:

Property	Description
Logical Component	[steps only] Specifies the logical component that the step accesses to perform its action (see Logical Components (Solution Manager) [page 112]).
Number ID	Specifies the place of the process or step in its parent.
Source / Source location	Specifies the name and location of the repository where the source is stored.
Original name	Specifies the name of the process or step in Solution Manager

In addition, the following tabs are available:

- *Process Steps* - [business processes only] - lists the steps, human processes, subprocesses, automated processes, and start, intermediate, and end events contained in the process.
- *General Documentation* and *Project Documentation* - lists the documentation associated with the process or step (see General and Project Documentation (Solution Manager) [page 113]).
- *Transactions* - lists the transactions associated with the process or step.

5.3 Logical Components (Solution Manager)

Logical components are software systems identified by their product, version, and instance name, and are displayed as swimlanes in business process diagrams. When you select scenarios and processes for import, PowerDesigner automatically imports the logical components that they use.

Context

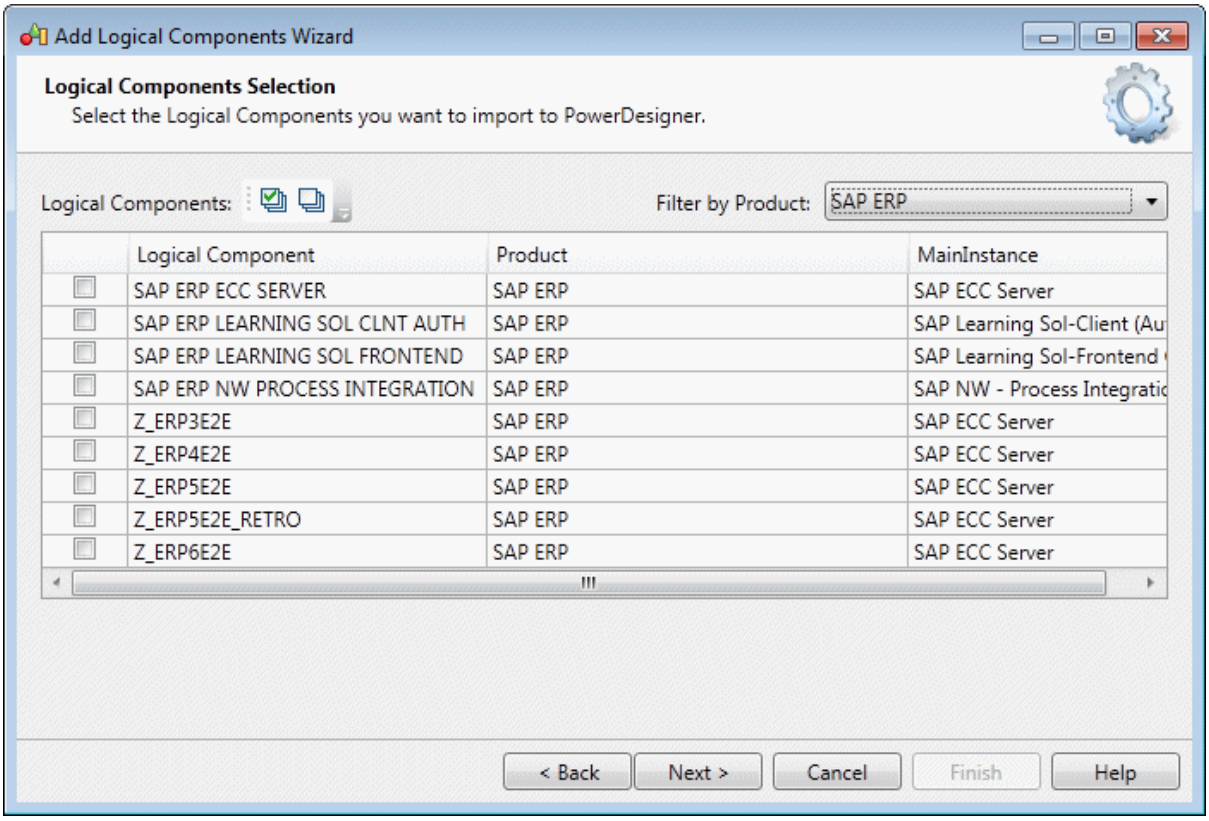
To add a logical component to your business process diagram, drag it from the Browser and drop it in the diagram to create a swimlane, which can contain process steps and other objects (see Business Processes (Solution Manager) [page 110]).

You cannot create new logical components in PowerDesigner, but you can import other logical components from the system landscape defined on your server for use in your processes.

> **Note**
>
> This feature depends on your having correctly configured the advanced configuration parameters (see Specifying Advanced Solution Manager Connection Parameters [page 117]).

Procedure

1. Right-click the model node in the Browser or a diagram background and select *Add Logical Components* to open the wizard, and click *Next* on the Welcome page.
2. Enter your Solution Manager host name and port number, along with your user name and password, and then click *Next*.
3. Optionally filter the list by product, select the logical components that you want to add to your project from the list, and then click *Next*.



4. Review the logical components that will be imported and then click *Finish* to import them to your model. For servers containing many logical component definitions, the import may take some considerable time.
5. Drag a logical component from the browser and drop it onto a business process diagram to add it as a swimlane in the diagram and assign process steps to it (see Business Processes (Solution Manager) [page 110]).

5.4 General and Project Documentation (Solution Manager)

When you select scenarios and processes for import, PowerDesigner automatically imports the SAP general reference documentation or the project-specific documentation that they reference, generally including a link to the represented document.

PowerDesigner will also import:

- Document Types - Provide standard types to categorize documents.
- Document Statuses - [if you have correctly configured the advanced configuration parameters (see Specifying Advanced Solution Manager Connection Parameters [page 117]] Provide standard status names to categorize documents. If the parameters are not set or no statuses are defined on the server, the standard `Copy Editing`, `In Processing`, `Released`, and `Review` statuses will be provided.

You can create project documentation of type URL in PowerDesigner by clicking the *Create an Object* tool on the *Project Documentation* tab of a scenario, process, or process step property sheet. To add existing documentation to an object, use the *Add Objects* tool.

You cannot create general documentation or documentation types and statuses, but you can associate existing instances of these objects with your scenarios, processes, and process steps.

> ⚠ Caution
>
> Project documents that are predefined in a Solution Manager project template (and are created automatically in a project when you create it from the template) cannot be subsequently associated with new or existing scenarios, processes, or process steps. If you do add such a document to a scenario, process, or step in PowerDesigner, then this change and any other changes you make to the list of documents associated with the object will not be exported to the Solution Manager server. A warning message will be displayed during the export, but PowerDesigner will not be aware of the export failure or of the differences that will subsequently exist between the documentation associated with the object in the model and the documentation associated with it on the server.

5.5 Organization Units, Transactions, and Master Data (Solution Manager)

PowerDesigner can import organization units, master data, and transactions to provide context for your processes. You cannot export changes to or new instances of these objects created in PowerDesigner, but you can associate existing instances with your processes and scenarios.

To add these objects to your project, scenario, or process diagram, drag and drop them from the Browser:

- Organization Units - Represent company groups, divisions, or locations.
- Transactions - Specify an action performed by a logical component. Each process step generally initiates one or more transactions, and transactions can also be associated with processes, organization units, and master data.
- Master data - Represent general cross-scenario information, such as material masters, business partners, countries, and measurement units.

5.6 Importing Business Processes from Solution Manager

PowerDesigner provides a wizard to allow you to import your projects, scenarios, and processes from Solution Manager v7.1 for editing in a BPM.

Context

> ℹ Note
>
> In order to import business processes into PowerDesigner, you must have correctly installed and configured the Business Process Blueprinting tool on your Solution Manager server. For detailed information, see the

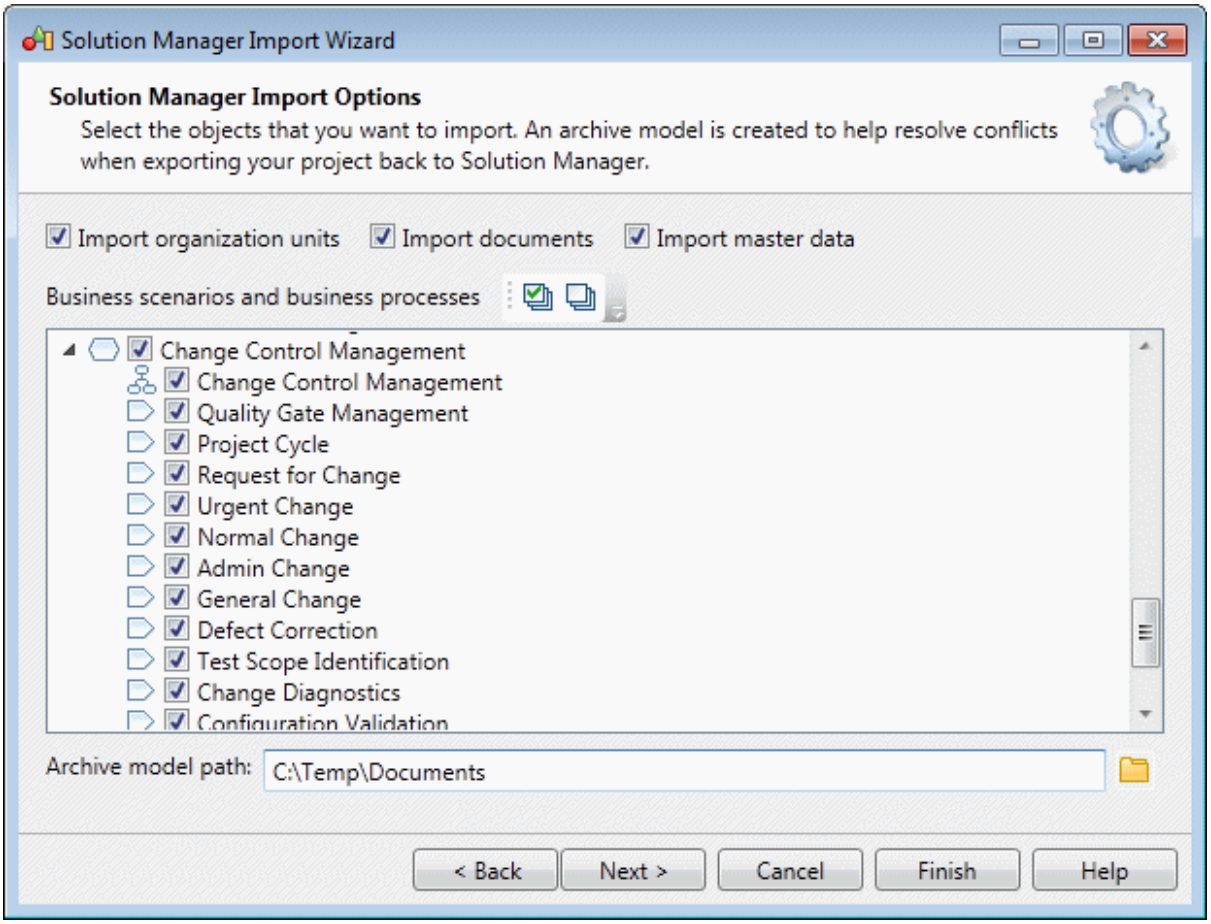
Business Process Blueprinting Installation and Configuration Guide or *Upgrade Guide*, which are available in the SAP Solution Manager 7.1 section of the SAP Support Portal Installation Guides site.

Procedure

1. Select *Language* > *Update Model from Solution Manager* to open the wizard, and click *Next* on the Welcome page.
2. Enter your Solution Manager host name and port number, along with your user name and password, and then click *Next*. PowerDesigner automatically creates a default connection in the *Connection name* list, and you can manage this and other Solution Manager connections with the tools to the right of the list.

   > **Note**
   >
   > If PowerDesigner is unable to obtain the default context (package ID) for the Solution Manager server, the *Context (package ID)* field will be displayed. If you do not know the ID, click the field name to go to the Business Process Blueprinting page on your server, and then click the configuration file download link for the selected Solution Manager Related Package, open the text file, and copy the ID from it.

3. Select the project that contains the processes you want to import, and then click *Next*.
4. Select the business scenarios, scenario flow diagrams, and processes that you want to import from the list, and then click *Next*.



By default, all scenarios, scenario flow diagrams, and processes are selected, and the options to import organization units, documents, and master data are also selected. The *Archive model path* field provides the location where the archive model (which preserves the current state of the project and is used during generation to help resolved conflicts in the case of concurrent changes by other users) will be saved.

5. Review the objects that will be imported and then click *Finish* to import them to your model. For large projects containing many scenarios and processes, the import may take some considerable time.

5.6.1 Specifying Advanced Solution Manager Connection Parameters

To enable support for importing and exporting document statuses and adding logical components to your scenarios, you must specify the WSDL URL for binding the `BSIPROJECTDIRECTORYINTERFACE` and `BSISMSYINTERFACE` service definitions in the Modify Connection dialog.

Procedure

1. Click the *Properties* tool to the right of the *Connection name* field on the Solution Manager Connection page of the Import wizard to access the *Modify Connection* dialog.

   The *Connection name*, *Host name*, *Port number*, *Version*, *Language*, *User name*, *Context (package ID)* fields are generally automatically completed.
2. Click the Project Directory service field name to connect to your Solution Manager server in your default Web browser, enter your User and Password in the SAP NetWeaver logon screen and click *Log on*.
3. On the *Service Administration* tab of the SOA Management screen, click the *Web Service Configuration* link.
4. Enter a *Search Pattern* of `*bsi*` and click *Go* to find the `BSIPROJECTDIRECTORYINTERFACE` and `BSISMSYINTERFACE` service definitions.
5. Click the `BSIPROJECTDIRECTORYINTERFACE` service definition in the Search Results list and then click *Apply Selection* to display its details.
6. Scroll down if necessary, click the *Show / hide selected Binding's or Service's WSDL URL* link, copy all the text in the WSDL URL for Binding field, and paste it into the *Project Directory service* field in the Modify Connection dialog.
7. Return to the SOA Management screen, click the `BSISMSYINTERFACE` service definition in the Search Results list and then click *Apply Selection* to display its details.
8. Click the *Show / hide selected Binding's or Service's WSDL URL* link, copy all the text in the WSDL URL for Binding field, and paste it into the *SMSY service* field in the Modify Connection dialog.
9. Click *OK* to return to the Import wizard.

5.7 Exporting Business Processes to Solution Manager

PowerDesigner provides a wizard to allow you to export changes made to your projects, scenarios, and processes for testing and implementation in Solution Manager v7.1.

Context

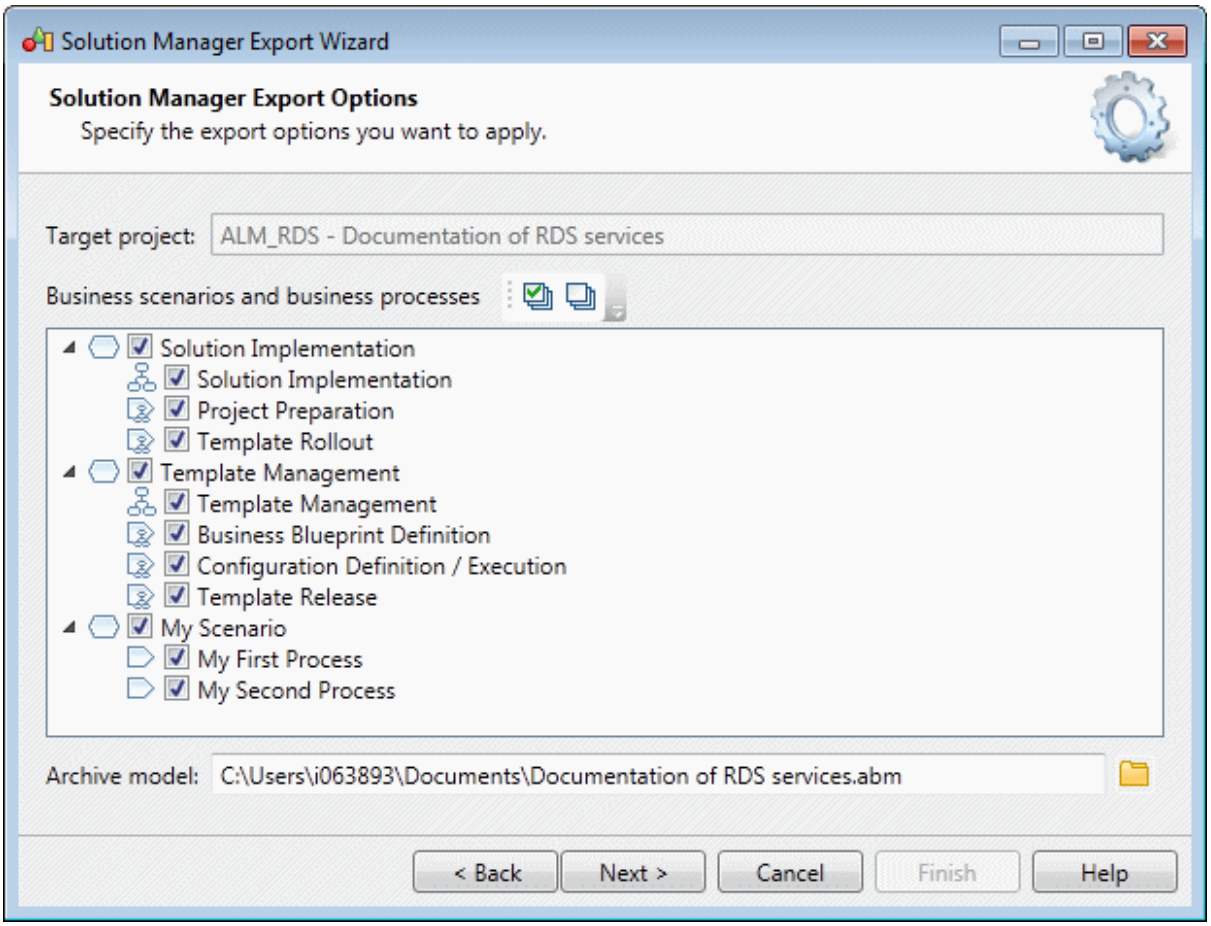
> **Note**
>
> Changes to logical components, organization units, master data, and general and project documentation (or new instances of these objects created in PowerDesigner) cannot be exported to Solution Manager. If you

delete business scenarios or processes in your model, these deletions will not be propagated to Solution Manager during the export, and must be performed directly on the server.

Procedure

1. Select *Language* > *Apply Model Changes to Solution Manager* to open the wizard, and click *Next* on the Welcome page.

   The wizard checks your model for consistency and displays any errors which may compromise the generation.
2. Enter your Solution Manager host name and port number, along with your user name and password, and click *Next*. PowerDesigner automatically creates a default connection in the *Connection name* list, and you can manage this and other Solution Manager connections with the tools to the right of the list.
3. Select the business scenarios, scenario flow diagrams, and processes you want to export, and click *Next*.



4. Review the objects that will be exported and then click *Finish* to generate them to Solution Manager.

> **Note**
>
> If PowerDesigner detects conflicts between changes made in the model and changes to the same objects on the server, then a merge dialog (see *Core Features Guide* > *Modeling with PowerDesigner* > *Comparing*

and Merging Models) will open to allow you to select, for each conflict, which of the conflicting changes will prevail. The resolutions that you select will first be applied to the model, and then your changes will be exported to the server.

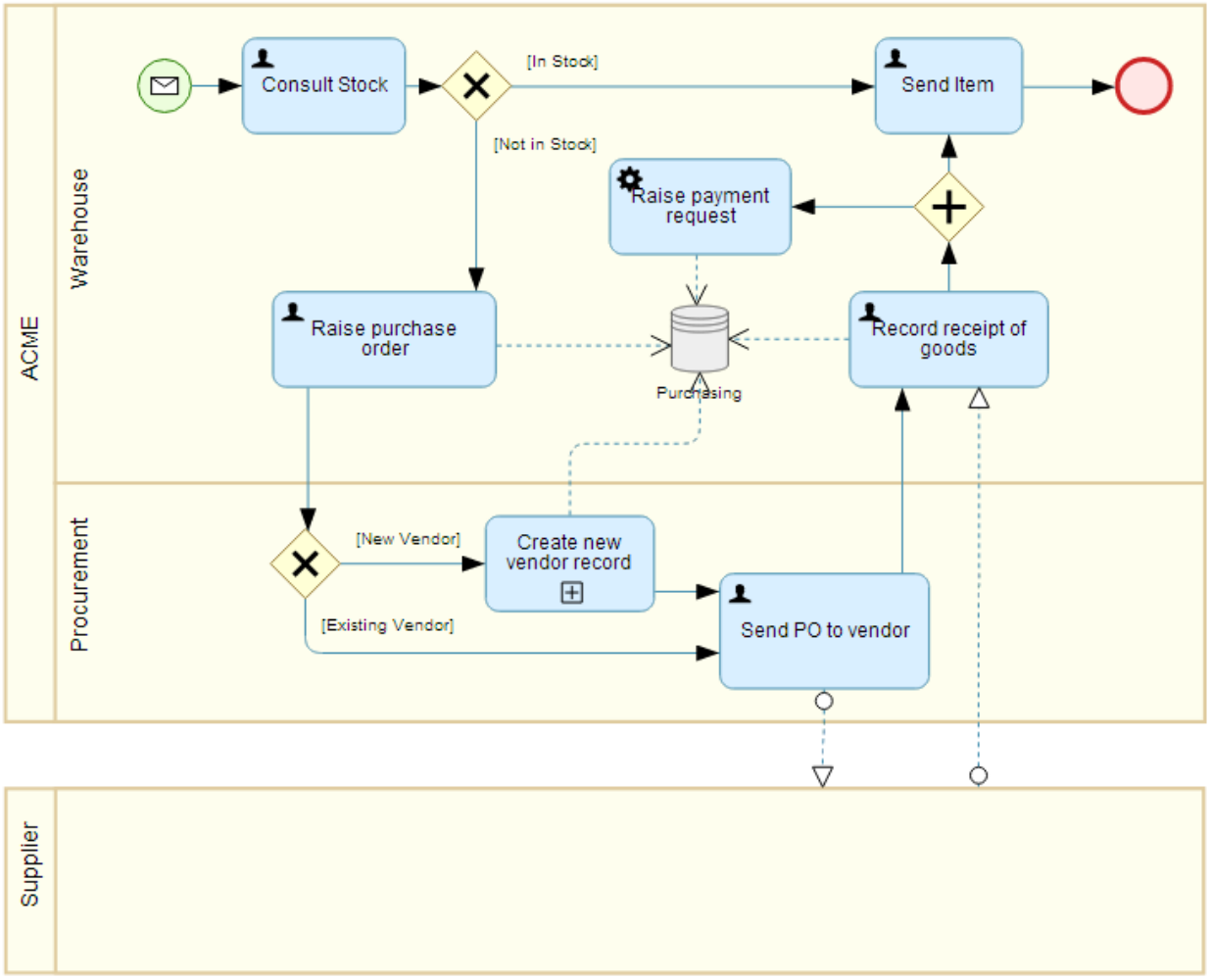
6 BPMN 2.0 Descriptive

Business Process Modeling Notation (BPMN) 2.0 is a standardized graphical notation intended to promote communication between non-technical business users who must document their processes and developers seeking to implement them using business execution languages. BPMN 2.0 Descriptive is aimed at business users and contains a subset of the BPMN 2.0 objects suitable for business process design and analysis.

PowerDesigner provides support for two variants of BPMN 2.0. For information about BPMN 2.0 Executable, see BPMN 2.0 Executable [page 134].

> **Note**
>
> When opening BPMN models created in previous versions of PowerDesigner, you will be invited to choose between converting them to BPMN Descriptive or keeping them in the BPMN Executable format that was originally supported.

PowerDesigner supports creating BPMN 2.0 Descriptive process diagrams, which focus on the sequence flow in a single process (which can be in a pool), and collaboration diagrams, which include two or more pools, with messages passing between them:



The following tools are available in collaboration and process diagrams:

Table 52:

Tool	Description
...	Start Events - Initiate a process (see Start and End Events (BPMN Descriptive) [page 125]). Use the specialized tools to create a: • Timer Start Event - To trigger the process at a particular time or date. • Message Start Event - To trigger the process by receipt of a message.
...	Activities - Work performed within a process (see Tasks (BPMN Descriptive) [page 126]). Use the specialized tools to create a: • Service Task - Represents work performed by a system without human intervention. • User Task - Represents work performed by a human with the aid of software. • Sub-Process - Represents a set of tasks that can be detailed in the sub-diagram created under the sub-process. To enter the sub-diagram, press *CTRL* and double-click the sub-process symbol. • Call Activity - Represents a reference to another process, which you select when creating the symbol.

Tool	Description
 ...	End Events - Terminate a process (see Start and End Events (BPMN Descriptive) [page 125]). Use the specialized tools to create a: • Message End Event - To terminate the process and send a message. • Terminate End Event - To terminate the process along with any parallel processes.
	Parallel Gateways - Split the control flow, routing the flow to all outgoing branches simultaneously (see Gateways (BPMN Descriptive) [page 128]). When merging, wait for all incoming branches to complete. Use this gateway with two or more outgoing flows when you want to represent activities that must be performed simultaneously.
	Exclusive Gateways - Split the control flow, routing the flow to one outgoing branch (see Gateways (BPMN Descriptive) [page 128]). When merging, waits for one incoming branch to complete before triggering the outgoing flow. Use this gateway with two or more outgoing flows when you want to represent a choice between different activities, and enter the reason for each choice in the *Condition* field on the *General* tab of each outgoing flow. For example, `Order > $10,000`.
	Pools - Represent companies, departments, or roles, and contain one or more lanes representing sub-entities within these organizations (see Pools and Lanes (BPMN Descriptive) [page 122]). To add further lanes to a pool, click on it with the tool. To create an additional pool in the diagram, click in empty space. You can drag lanes from one pool to another, or into empty space to create a new pool, and a single lane can be reused and appear in multiple different pools. **Note** The name of the lane in a single-lane pool is hidden by default, but you can display it by selecting the pool and then right-clicking in the selection area above it and disabling *Hide Lane Names*. This option is automatically disabled and cannot be enabled for multi-lane pools.
 ...	Data Objects - Represent a document or data store providing information used in a process (see Data (BPMN Descriptive) [page 130]).
	Message Flow - Links a pool (or one of its activities) to another pool (or one of its activities), and passes a message between them.
	Sequence Flow - Links two elements (events, activities, gateways) to show the progress in a process.
	Data Association - Links a data object to an activity or event.

6.1 Pools and Lanes (BPMN Descriptive)

Pools represent companies, departments, or roles. Lanes represent sub-entities within these organizations and appear as swimlanes inside the pool. Many BPMN diagrams contain one or more pools, with all the other objects placed in the lanes of these pools.

Note

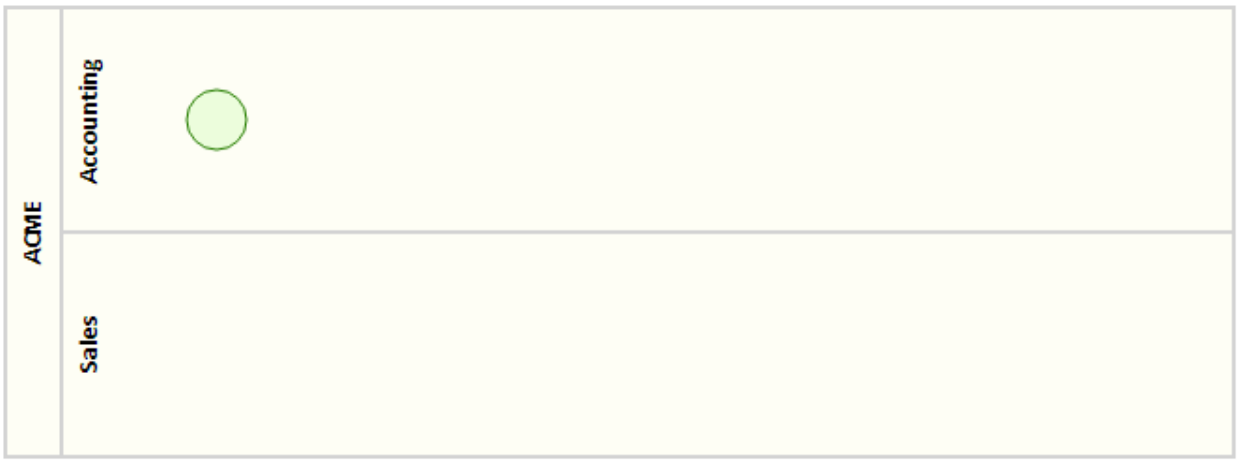
Pools can be vertical (top to bottom) or horizontal (left to right). You can change the orientation of your diagram at any time by right-clicking the diagram background, selecting *Display Preferences* and selecting the appropriate option in the *Organization unit swimlane* groupbox.

To create a pool, click the *Pool/Lane* tool to select it, and then click in empty space in the diagram.

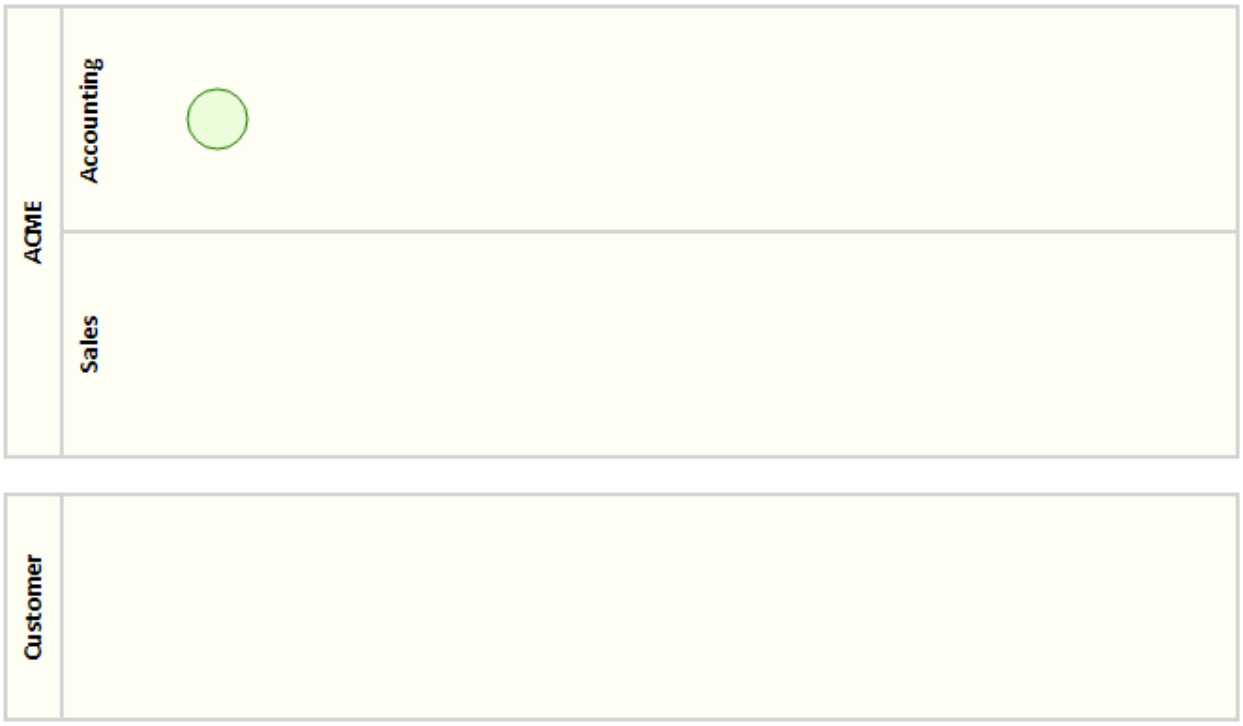
A single pool in a diagram generally represents the organization:

To add a lane to a pool, click the *Pool/Lane* tool to select it, and then hover over an existing pool in the diagram. Click to create the lane in this position.

Each lane in the primary pool represents a department or other sub-entity within the organization:



BPMN diagrams may contain a second pool to represent a partner, such as a customer or supplier with whom the organization interacts. To create a second pool, click the *Pool/Lane* tool to select it, and then click in empty space in the diagram:



Since you generally will not know the details of a partner's processes, the second pool is commonly treated as a "black box". No tasks or other objects are created within it, and it is linked to the first pool only via message flows.

> **Note**
>
> You can drag lanes from one pool to another, or into empty space to create a new pool, and lanes can be reused and appear in multiple different pools.

Pools and lanes can have the following properties:

Table 53:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Multi-instance	[pools] The pool represents multiple instances of the specified role.

6.2 Start and End Events (BPMN Descriptive)

A process begins with a start event and terminates with one or more end events.

To create a start, click the appropriate *Start* tool to select it, and then click in the diagram. In BPMN 2.0 Descriptive, PowerDesigner supports the following types of start events:

Table 54:

Symbol	Description
	Undefined Start Event - The process simply starts without any specific triggering event.
	Message Start Event - The process begins following receipt of a message, such as an order or enquiry.
	Timer Start Event - The process begins on a specific date or at a specific time, such as Monday morning at 9am.

To create an end, click the appropriate *End* tool to select it, and then click in the diagram.

In BPMN 2.0 Descriptive, PowerDesigner supports the following types of ends:

Table 55:

Symbol	Description
	Standard End Event - The process simply ends when all of the tasks are completed.
	Message End Event - The process terminates by sending a message, such as a quotation, invoice, or report.
	Terminate End Event - All tasks in any parallel sequence flows are terminated immediately when one branch reaches a terminate end event.

Starts and ends can have the following properties:

Table 56:

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

6.3 Tasks (BPMN Descriptive)

The main contents of a process are the tasks that are performed during its execution.

To create a task, click the appropriate *Task* tool to select it, and then click in the diagram.

By default, a standard task is created. To change the type of task, click the *Properties* tool and select a type from the list. In BPMN 2.0 Descriptive, PowerDesigner supports the following types of tasks:

Table 57:

Symbol	Description
	Standard Task - Can be used for any kind of activity.
	Service Task - A task performed by an application or web service without any human input.
	User Task - A task performed by a human interacting with a software application.
	Call Activity - A task which reuses a globally defined process. For example, you may define the login process and then reuse it in multiple processes (see Call Activities (BPMN Descriptive) [page 127]).
	Sub-Process - A task that is, itself, broken down into subtasks (see Sub-Processes (BPMN Descriptive) [page 127]).

Tasks can have the following properties:

Table 58:

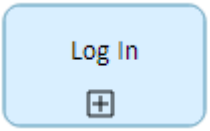
Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide 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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Composite status	Specifies whether the process contains sub-processes. You can choose between: • Atomic Process (default) – the process does not contain any sub-processes. • Decomposed Process – the process can contain sub-processes, which are listed on a Sub-Processes tab and can be displayed in a business process diagram under the process. If you revert the process to Atomic status, then any sub-processes that you have created will be deleted.
Reusable process	Specifies that the task can be referenced for reuse by a call activity (see Call Activities (BPMN Descriptive) [page 127]).
Called object	[Call Activities] Specifies the global task or process that is reused by the call activity.

6.3.1 Sub-Processes (BPMN Descriptive)

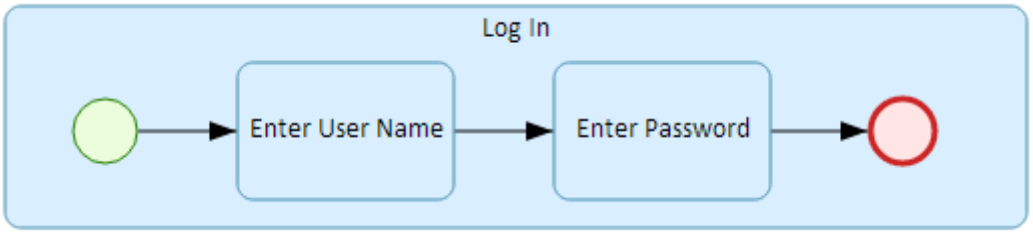
A sub-process is a task that is broken down into sub-tasks. For example, you may break the Log In task into the sub-tasks Enter User Name and Enter Password.

To create a sub-process, click the *Sub-Process* tool to select it, and then click in the diagram.

The sub-process is initially empty:



Add tasks and other objects as necessary to complete the definition of the sub-process:



The symbol grows to accommodate these objects, and you can resize it and reposition them as necessary.

> **Note**
>
> Objects created inside a sub-process are listed in the Browser inside the sub-process. A sub-diagram is also created under the sub-process in the Browser. To open it, double-click it in the Browser or press CTRL and double-click the sub-process symbol.

6.3.2 Call Activities (BPMN Descriptive)

Call activities are tasks that reuse an existing global process or task. For example, you may define a process called Log In and then reuse it in various other processes.

To create a call activity, click the *Call Activity* tool to select it, and then click in the diagram. In the dialog that opens, select the task to reuse and then click *OK*. The tasks that are available to be reused from this list must be open in the Workspace and have the *Reusable Process* property selected. The reused task is recorded in the *Called object* property on the *Implementation* tab.

6.4 Gateways (BPMN Descriptive)

Gateways control the sequence flow of the process, and can split or merge the flow to show many decisions or simultaneous actions are required.

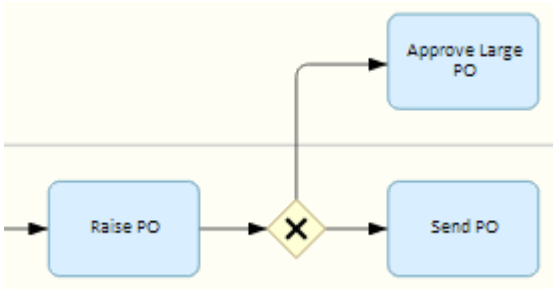
To create a gateway, click the appropriate *Gateway* tool to select it, and then click in the diagram.

In BPMN 2.0 Descriptive, PowerDesigner supports these two types of gateways:

Table 59:

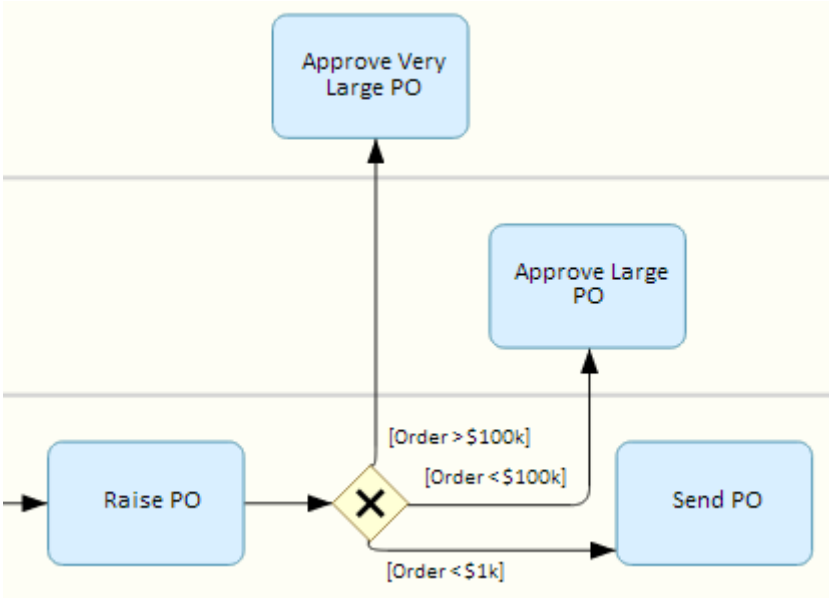
Symbol	Description
	Exclusive gateway - Only one outgoing branch is performed, depending on the condition.
	Parallel gateway - All outgoing branches are performed simultaneously.

Gateways generally have two or more tasks or other objects after them in the control flow:



To specify a condition on a sequence flow connecting the gateway to an object, open the property sheet of the flow and enter an appropriate value in the *Condition* field. The value is displayed in the diagram on the sequence flow near to the gateway:

You should add a condition to all sequence flows leaving the gateway. You can add further alternate sequence flows as necessary. In this example, once a purchase order is raised, an exclusive gateway controls the subsequent sequence flow based on the value of the order:

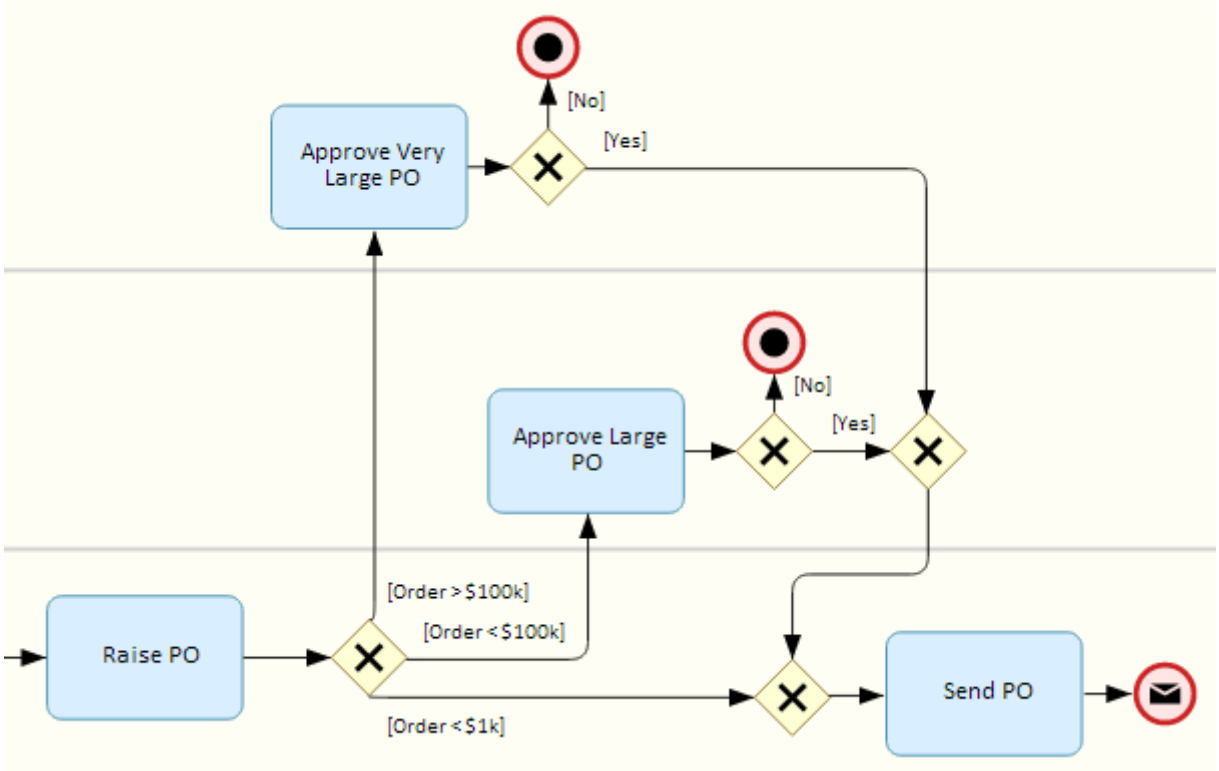


Gateways are also used to merge sequence flows when the two or more parallel or exclusive flows reunite to continue the process. In this case the two types of gateway have the following meanings:

Table 60:

Symbol	Description
	Exclusive gateway - Waits for one incoming branch to complete before continuing.
	Parallel gateway - Waits for all incoming branches to complete before continuing.

In this example, one of the approval tasks will arrive at the final exclusive gateway, which then triggers the sending of the purchase order:



Gateways can have the following properties:

Table 61:

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

6.5 Data (BPMN Descriptive)

Data objects represent data used in the process.

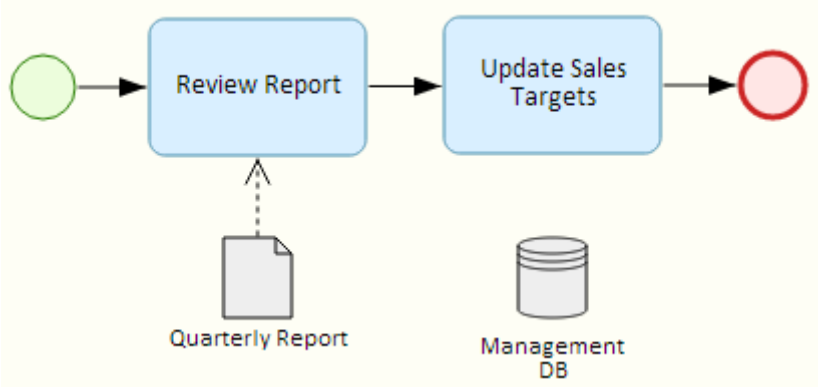
To create data, click the *Data Store* or *Data Object* tool to select it, and then click in the diagram.

In BPMN 2.0 Descriptive, PowerDesigner supports the following types of data:

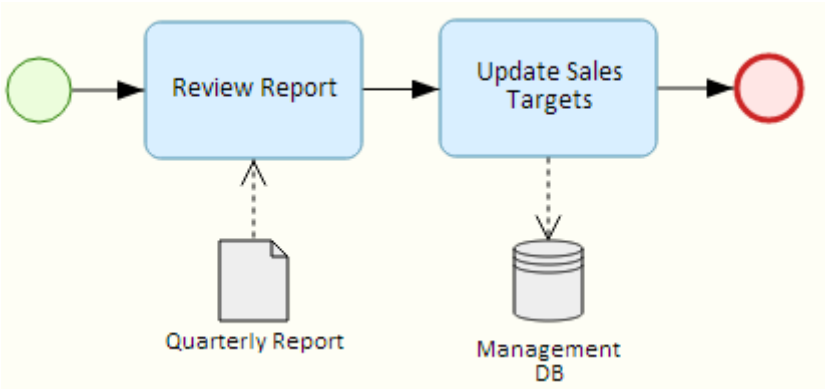
Table 62:

Symbol	Description
	Data object - Information used in the process.
	Data store - A database, filing cabinet, or other location from which the process can read or to which it can write data, and which persists beyond the lifetime of the process instance.

To show a task (or other object) reading from a data object or data store, select the *Data Association* tool and draw a link from the data to the task to create a data association (a dashed line) pointing to the task:



To show a task (or other object) writing to a data object or data store, select the *Data Association* tool and draw a link from the task to the data to create a data association (a dashed line) pointing to the data:



Data can have the following properties:

Table 63:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Collection	[data objects] Specifies that the object represents a collection of elements.
Capacity / Unlimited	[data stores] Specify the capacity of the store either as a numeric value or as unlimited.

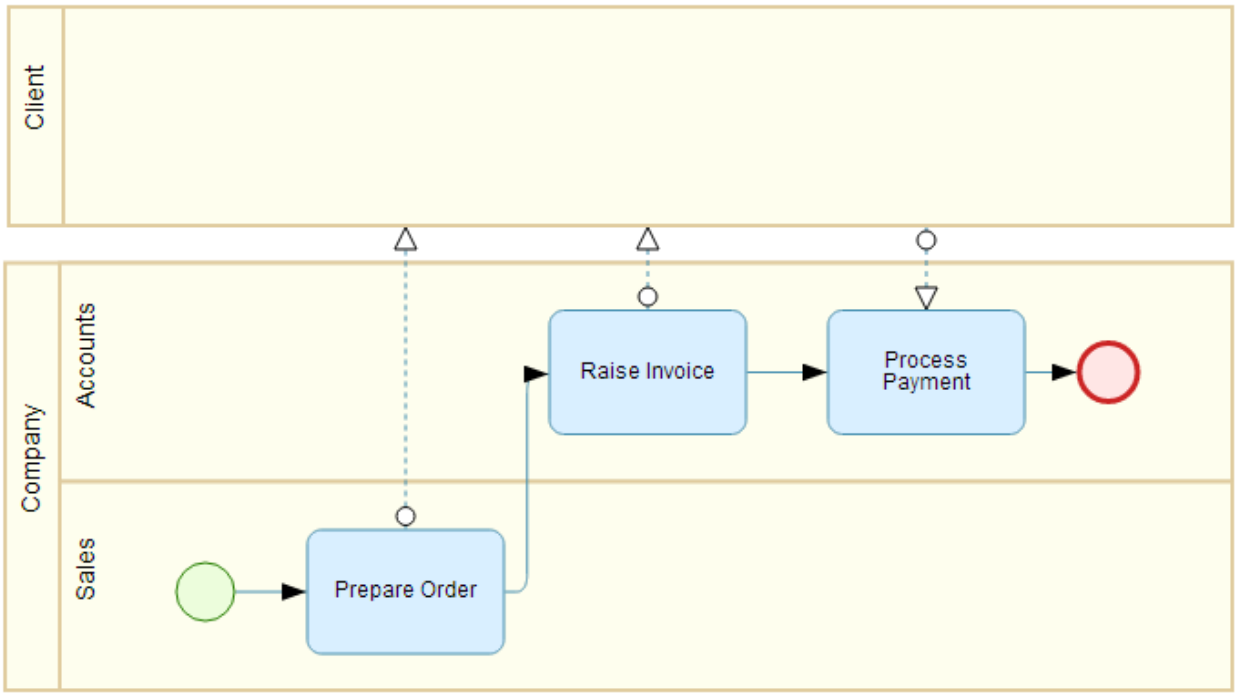
6.6 Sequence and Message Flows (BPMN Descriptive)

Sequence flows are solid lines with an arrow at one end, which link the elements in a process in the diagram or in a single pool and show the order in which they are performed. Message flows are dotted lines with an arrow at one end, which link two separate pools (or elements in two separate pools) and show the direction in which the message is sent.

To create a sequence flow, click the *Sequence Flow* tool to select it, and then click and drag from one object to another in a single pool in the diagram.

To create a message flow, click the *Message Flow* tool to select it, and then click and drag from one object in one pool to another object in a different pool in the diagram.

In the following example, note how the flows between tasks in a single pool are solid line sequence flows, while the flows between pools are dotted line message flows:



Sequence and message flows can have the following properties:

Table 64:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Source/ Destination	Specify the objects that are linked by the flow. For sequence flows, the source object comes before the destination object in the process. For message flows, the source object emits the message and the destination object receives it.
Condition	[sequence flows] Specifies the condition that must be fulfilled for the process to take this branch following a gateway.

7 BPMN 2.0 Executable

Business Process Modeling Notation (BPMN) 2.0 is a standardized graphical notation intended to promote communication between non-technical business users who must document their processes and developers seeking to implement them using business execution languages. BPMN 2.0 Executable includes all the standard BPMN 2.0 objects, and is aimed at technical modelers and those who are reverse-engineering from SAP BPM or Eclipse BPMN2 Modeler.

PowerDesigner provides support for two variants of BPMN 2.0. For information about BPMN 2.0 Descriptive, see BPMN 2.0 Descriptive [page 120].

> **Note**
>
> When opening BPMN models created in previous versions of PowerDesigner, you may be invited to choose between converting them to BPMN Descriptive or keeping them in the BPMN Executable format that was originally supported.

PowerDesigner supports the following BPMN Executable 2.0 diagrams:

- Process diagrams - 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.
- Collaboration diagrams - Can additionally show 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.
- Conversation diagrams - Provide an overview of the communications between participants. Conversation diagrams can be created and edited in the PowerDesigner desktop client, but are read-only in PowerDesigner Web.
- Choreography diagrams - Focus on the detail of the conversation between two or more participants, and which are often linked to specific conversation nodes. Choreography diagrams can be created and edited in the PowerDesigner desktop client, but are read-only in PowerDesigner Web.

>For information about working with PowerDesigner Web, see *PowerDesigner Web*

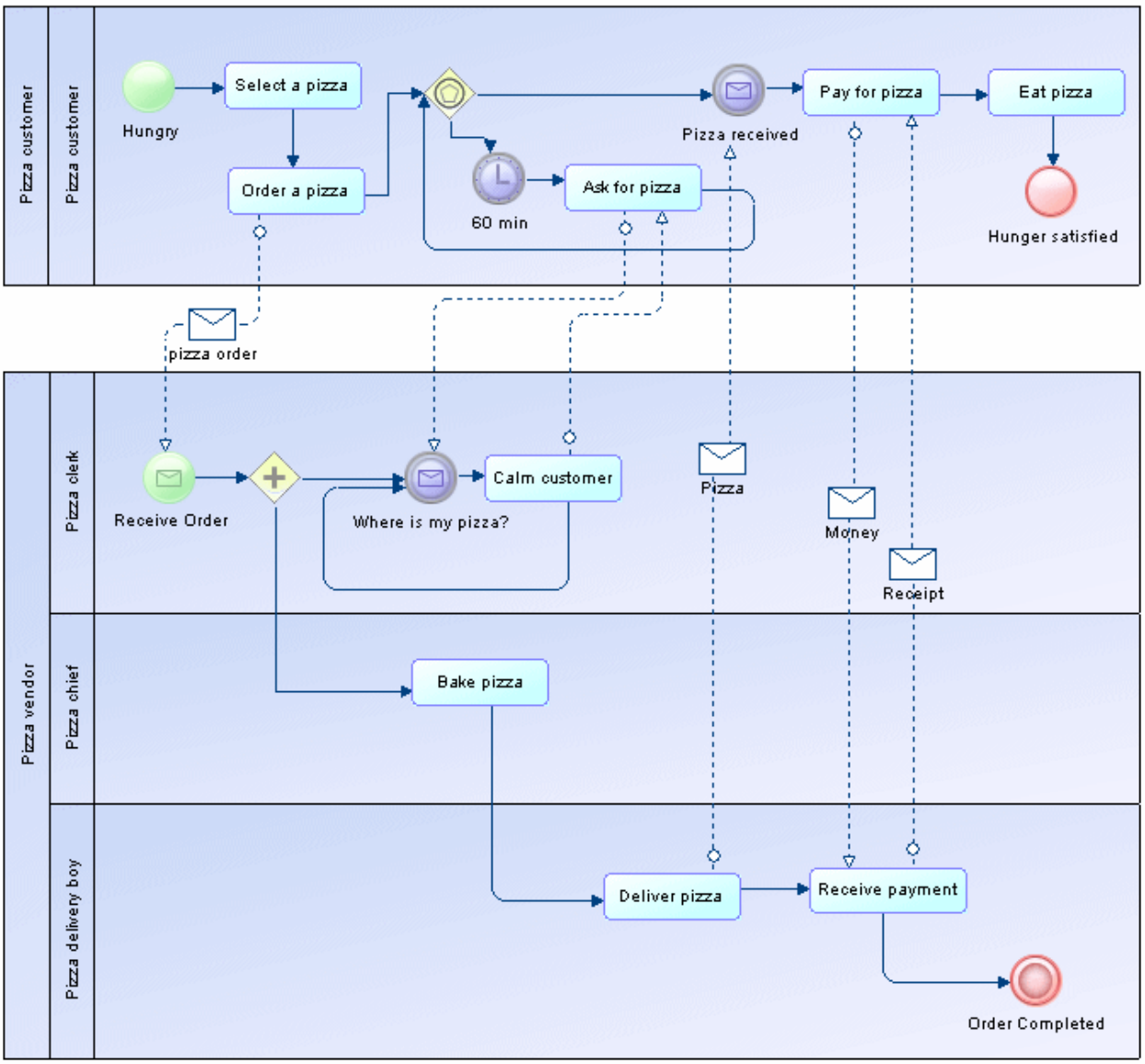
> **Note**
>
> When creating reports for your BPMN 2.0 models, we recommend that you start from one of the BPMN 2.0 report templates, which provide a framework for organizing all the extensions in logical groups. We recommend that you perform a model check on your completed model (or after major changes) to verify the validity of your diagrams.

7.1 Collaboration and Process Diagrams (BPMN Executable)

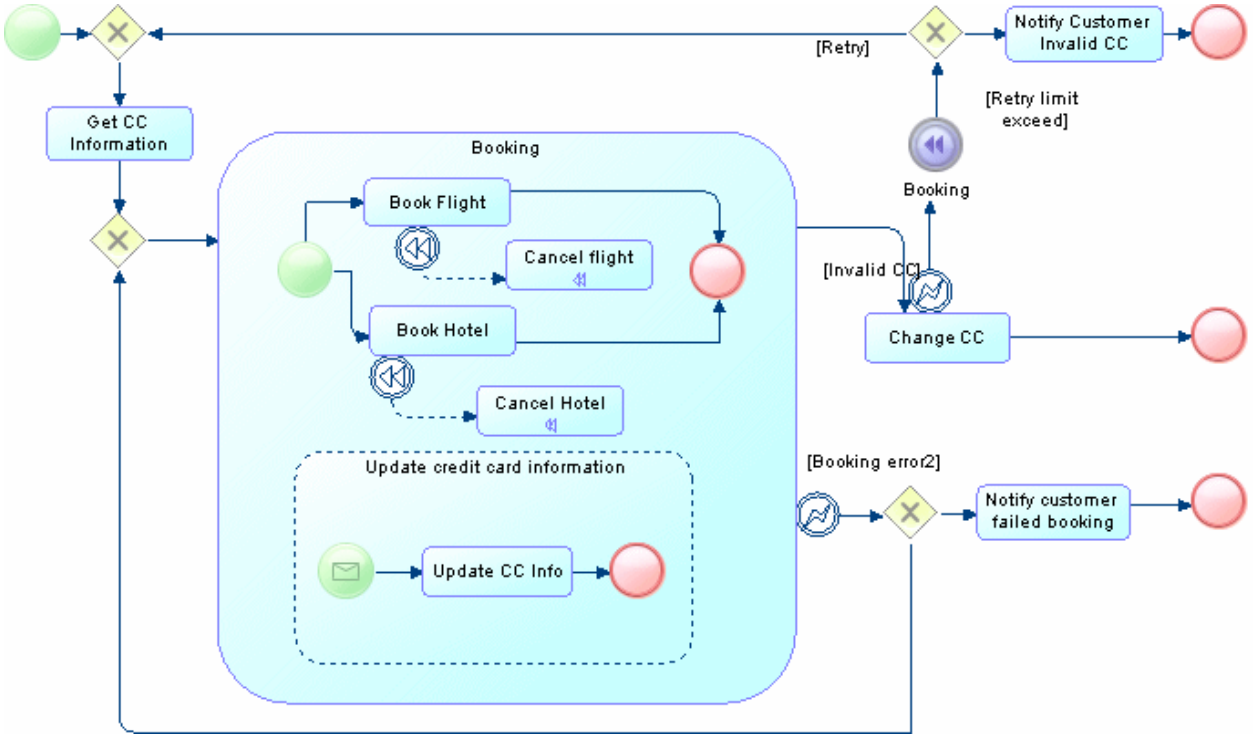
A collaboration diagram analyzes the sequence flow of processes and the exchange of messages between participants (represented as swimlanes and pools). Each pool contains an implicit process with a start event and one or more end events. A process diagram analyzes the sequence flow in a single process 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:



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:

Table 65:

Tool	Description
...	Start Events - Initiate a process (see Start, Intermediate, and End Events (BPMN Executable) [page 142]). The various types of start events each have their own tools.
...	Intermediate Events - Trigger further activity during a process (see Start, Intermediate, and End Events (BPMN Executable) [page 142]). The various types of intermediate events each have their own tools.
...	End Events - Conclude a process (see Start, Intermediate, and End Events (BPMN Executable) [page 142]). The various types of end events each have their own tools.
...	Activities - Work performed within a process (see Activities (BPMN Executable) [page 145]). The various types of activities each have their own tools.
	Participant - Organization, business unit, or role represented as swimlanes and pools (see Pools and Lanes (BPMN Executable) [page 141]).
...	Gateways - Merge or split the sequence flow (see Gateways (BPMN Executable) [page 146]). The various types of gateways each have their own tools.
...	Data Objects - Information item used in a process (see Data and Data References (BPMN Executable) [page 148]). 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 Sequence and Message Flows (BPMN Executable) [page 153]).
	Sequence Flow - Links two elements (events, activities, gateways) in a process (see Sequence and Message Flows (BPMN Executable) [page 153]).

Tool	Description
	Data Association - Links a data object to an activity or event (see Sequence and Message Flows (BPMN Executable) [page 153]).

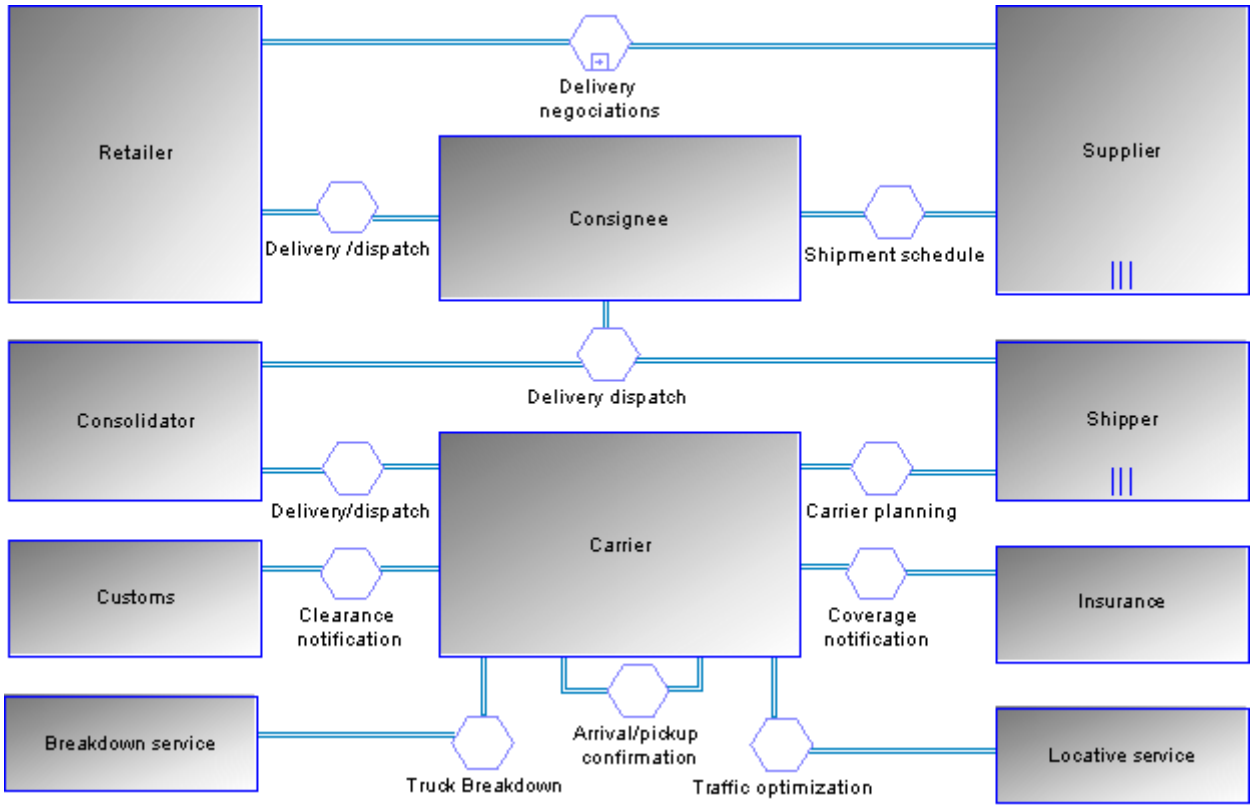
> **Note**
>
> You can change the type of an event, activity, or gateway by right-clicking its symbol and selecting the appropriate *Change to...* command.

7.2 Conversation Diagrams (BPMN Executable)

A conversation diagram focuses on the communications between participants. You cannot create or display processes or choreographies in this diagram.

> **Note**
>
> Conversation diagrams can be created and edited in the PowerDesigner desktop client, but are read-only in PowerDesigner Web.

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 conversation diagrams:

Table 66:

Tool	Description
	Participant - Organization, business unit, or role (see Pools and Lanes (BPMN Executable) [page 141]).
	Conversation Node - Links two participants and regroups a set of message exchanges that share the same correlation.
	Conversation Link - Links participants via a conversation node. Click in one participant and draw a link to another participant to automatically create a conversation node between them.

Conversation nodes have the following properties:

Table 67:

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.
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.
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 Correlation Keys (BPMN Executable) [page 150]). Each flow connected to the node must have the same key as the node.

> **Note**
>
> 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:

- 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.

Conversation links have the following properties:

Table 68:

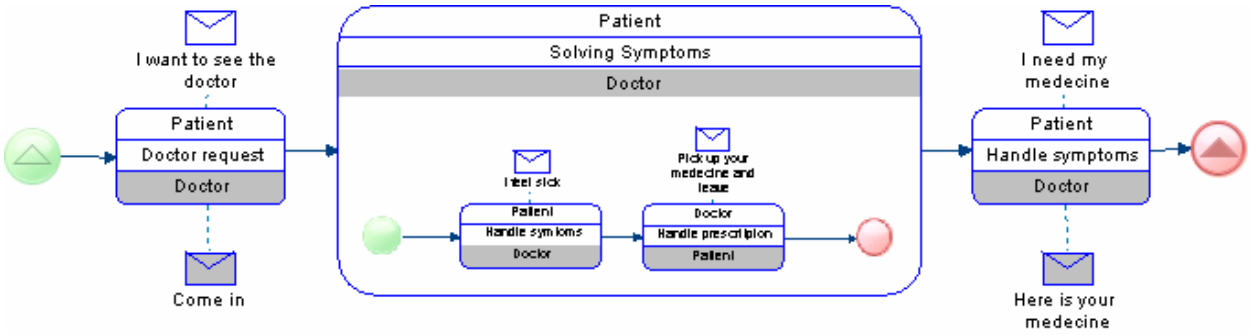
Property	Description
Name/Code/Comment	Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide 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 object.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

7.3 Choreography Diagrams (BPMN Executable)

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.

> **Note**
>
> Choreography diagrams can be created and edited in the PowerDesigner desktop client, but are read-only in PowerDesigner Web.

In the following example, the exchange of messages between a patient and a doctor is analyzed:



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:

Table 69:

Tool	Description
...	Start Events - Initiate a process (see Start, Intermediate, and End Events (BPMN Executable) [page 142]). The various types of start events each have their own tools.
...	End Events - Conclude a process (see Start, Intermediate, and End Events (BPMN Executable) [page 142]). The various types of end events each have their own tools.
	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.
...	Gateways - Merge or split the sequence flow (see Gateways (BPMN Executable) [page 146]). The various types of gateways each have their own tools.
	Message - Message sent to the choreography task by a participant (see Messages (BPMN Executable) [page 151]). 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 Sequence and Message Flows (BPMN Executable) [page 153]). 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 Sequence and Message Flows (BPMN Executable) [page 153]).

Choreography tasks have the following properties:

Table 70:

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.
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 object.
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.
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 object. Select the *Multiple* check box 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 object.
Loop characteristics	Specifies that the task is a loop or multiple-instance (parallel or sequential) choreography task.

7.4 Pools and Lanes (BPMN Executable)

Pools represent companies, departments, or roles. Lanes represent sub-entities within these organizations and appear as swimlanes inside the pool. Many BPMN diagrams contain one or more pools, with all the other objects placed in the lanes of these pools.

BPMN Executable pools and lanes behave in the same way as those in BPMN Descriptive (see Pools and Lanes (BPMN Descriptive) [page 122]).

In BPMN Executable, pools and lanes can also appear in conversation diagrams as square nodes (see Conversation Diagrams (BPMN Executable) [page 137]) and in choreography diagrams, where they do not have a separate symbol but are displayed on the top or bottom band of the choreography task symbol (see Choreography Diagrams (BPMN Executable) [page 139]).

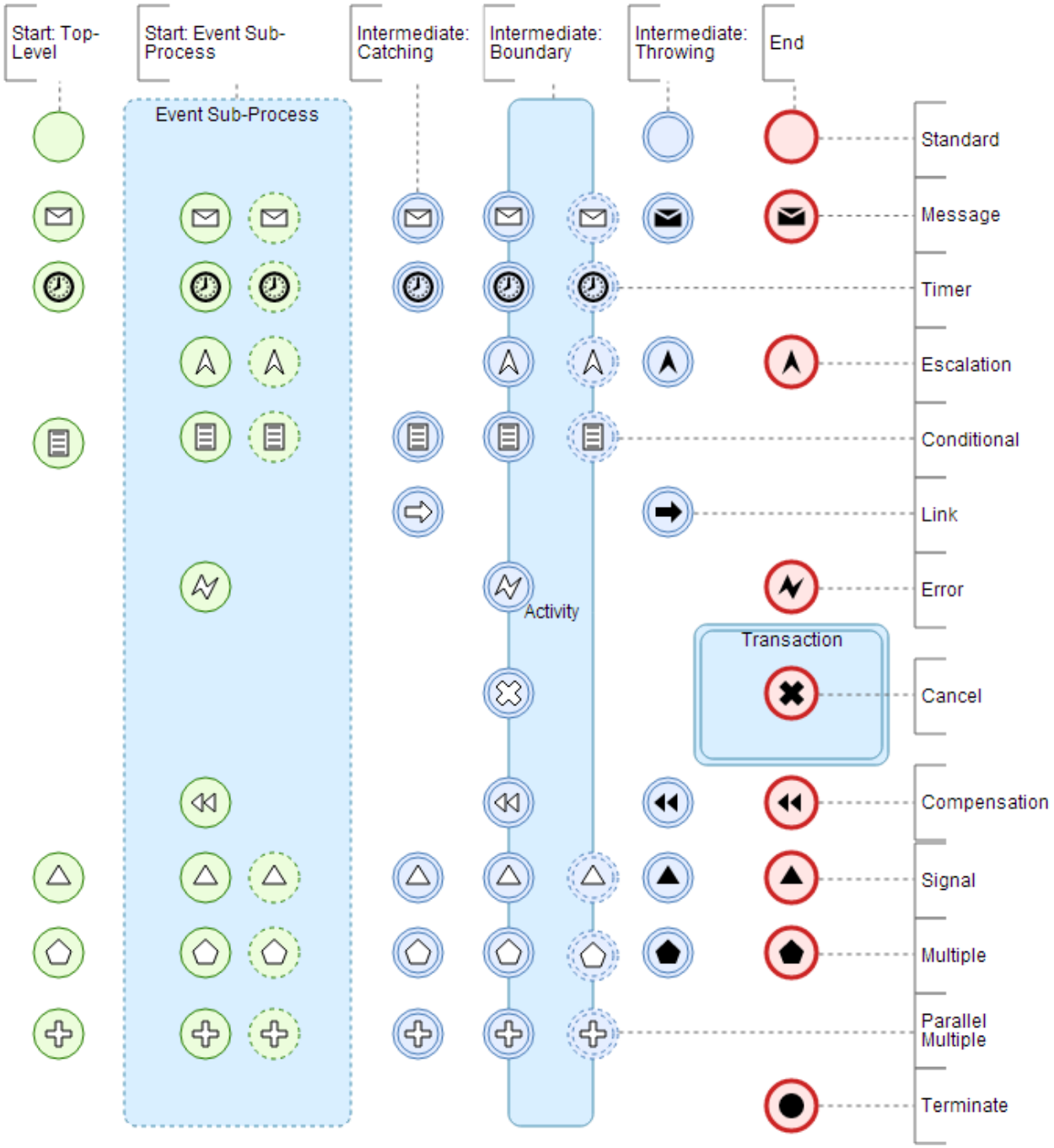
7.5 Start, Intermediate, and End Events (BPMN Executable)

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 happenings (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.

In BPMN Executable, various different types of start, intermediate and end events can be created depending on the context, and their type is indicated by their symbols. PowerDesigner supports all the types of events defined in BPMN 2.0:

- 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 canceled transactions or triggering cancellation.
- Compensation - Handling or triggering compensation.
- Signal - Signaling 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.

> **Note**
>
> You can change the type of any event by right-clicking its symbol or Browser entry and selecting the appropriate *Change to...* command.



You can create events in the contexts shown above in the following ways:

- Start events - can be created directly in the diagram, in a pool or lane, in a standard or transaction sub-process (standard starts only), or in an event-based sub-process. Use the appropriate tool from the Toolbox.

> **Note**
>
> Start events are not permitted in ad hoc sub-processes.

> **Note**
>
> To make an event-based sub-process start event non-interrupting (with the dashed outer circle), deselect the *Interrupting* property.

- Intermediate catch events - can be created directly in the diagram, in a pool or lane, or in any type of sub-process. Use the appropriate tool from the Toolbox.
- Intermediate boundary events - can be created on the edge of a task or of any type of sub-process. Right-click the task or sub-process and select *Add Boundary Events* > `<type>` or open the task or sub-process property sheet and use the tools on the *Boundary Events* tab.

> **Note**
>
> To make an intermediate boundary event non-interrupting (with the dashed outer circles), deselect the *Interrupting* property.

- Intermediate throw events - can be created directly in the diagram, in a pool or lane, or in any type of sub-process. Use the appropriate tool from the Toolbox.
- End events - can be created directly in the diagram, in a pool or lane or in any type of sub-process. Use the appropriate tool from the Toolbox.

> **Note**
>
> End events are not permitted in ad hoc sub-processes. Cancel end events are only permitted in transactions.

Executable events can have the following properties:

Table 71:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Interrupting	While most events interrupt the process modeled, event sub-process starts and intermediate boundary events can be specified as non-interrupting by deselecting this property. Non-interrupting events are marked by dashed outer circles.
Message	[message events] Specifies the message that is associated with the event (see Messages (BPMN Executable) [page 151]).

> **Note**
>
> For multiple and parallel multiple events, click the *Event Definitions* tab and use the *Add Objects* and *Create an Object* tools to reuse or create new event definitions to associate with the event.

7.6 Activities (BPMN Executable)

Activities are work that is performed within a process. You can create activities in collaboration and process diagrams.

To create an activity, click the appropriate *Activity* tool to select it, and then click in the diagram.

In BPMN 2.0 Executable, PowerDesigner supports the following types of activities:

Table 72:

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 (see Call Activities (BPMN Descriptive) [page 127]).
	Sub-process - An activity whose internal details have been modeled using activities, gateways, events, and sequence flows (see Sub-Processes (BPMN Descriptive) [page 127]).
	Event sub-process - An activity that 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.

> **Note**
>
> You can change the type of an activity by right-clicking its symbol or Browser entry and selecting the appropriate *Change to...* command.

Activities can have the following properties:

Table 73:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Composite status	Specifies whether the process contains sub-processes. You can choose between: • Atomic Process (default) – the process does not contain any sub-processes. • Decomposed Process – the process can contain sub-processes, which are listed on a Sub-Processes tab and can be displayed in a business process diagram under the process. If you revert the process to Atomic status, then any sub-processes that you have created will be deleted.
Reusable process	Specifies that the process can be referenced by a call activity (see Call Activities (BPMN Descriptive) [page 127]).
Start quantity/ Completion quantity	Specify the number of tokens that must arrive before the activity can begin and the number of tokens that must be generated from the activity. The default value is 1 and it is only changed in very advanced modeling situations.
Loop characteristics	Specifies that the activity may be repeated in one of the following ways: • <none> - default • Loop • Multi-Instance parallel • Multi-Instance sequential
Compensation	Specifies that the activity is intended for the purposes of compensation.
Called object	[Call Activities] Specifies the global task or process that is reused by the call activity.

7.7 Gateways (BPMN Executable)

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.

To create a gateway, click the appropriate *Gateway* tool to select it, and then click in the diagram.

In BPMN 2.0 Executable, PowerDesigner supports these types of gateways:

Table 74:

Symbol	Description
	Normal/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.

> **Note**
>
> You can change the type of a gateway by right-clicking its symbol or Browser entry and selecting the appropriate *Change to...* command.

Gateways can have the following properties:

Table 75:

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

Property	Description
Direction	Specifies how the gateway may be used. You can select: • **Unspecified** - The gateway may have any number of incoming and outgoing sequence flows. • **Converging** - The gateway may have multiple incoming sequence flows but must have no more than one outgoing sequence flow. • **Diverging** - The gateway may have multiple outgoing sequence flows but must have no more than one incoming sequence flow. • **Mixed** - The gateway contains multiple outgoing and multiple incoming sequence flows.
Expression / Expression alias	Specifies the condition that will be evaluated to decide which path the process follows following the gateway. The alias provides a short version of the condition which is displayed under the gateway in the diagram.

7.8 Data and Data References (BPMN Executable)

Data are physical or information items that are created, manipulated, or otherwise used during the execution of a process. Data references are objects that reference data objects for reuse. You can create data objects and references in collaboration and process diagrams.

> **Note**
>
> PowerDesigner does not support the association of data objects with sequence flows.

To create data, click the appropriate *Data* tool to select it, and then click in the diagram. When you click in the diagram with the *Data Object Reference* tool, you will be prompted to choose the data object to reuse.

By default, a standard data object is created. To change the type of data object, click the *Properties* tool and select a type from the list. To create a data reference, select **Data Object Reference** from the *Properties* list, and specify the data object to reuse in the *Data object* field. In BPMN 2.0 Executable, PowerDesigner supports the following types of data:

Table 76:

Symbol	Description
	Data object / Collection data object - Information flowing through the process.
	Data input / Collection data input - External input for the entire process, which can be read by an activity.
	Data output / Collection data output - Variable available as the result of the entire process.
	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 right-clicking its symbol or Browser entry and selecting the appropriate *Change to...* command.

Data can have the following properties:

Table 77:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Capacity/ Unlimited	[data stores] Specify the size of the data store or the fact that it has an unlimited capacity.
Data object	[data references] Specifies the data object to reference. Select an object from the list, or use the tools to the right of this field to create, delete, select an object, or review the properties of the selected object.
State	[data references] Specifies the state of the data object. You can select: • **`Initial`** • **`Processing`** • **`Completed`**
Collection	Specifies that the data object represents a collection of data, such as a list of order items.

Data associations can have the following properties:

Table 78:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Transformation type	Specifies the direction of the transfer of the item-aware element (see Item-Aware Elements (BPMN Executable) [page 152]): • **`Input`** - Specifies a read. The data association goes from the data object to the activity. • **`Output`** - Specifies a write. The data association goes from the activity to the data object.

Property	Description
Source item	[**Output**] Specifies the item-aware element (of type **Data Output**) defined on the activity to be transferred from it to the data object.
Target item	[**Input**] Specifies the item-aware element (of type **Data Input**) defined on the activity to be transferred to it from the data object.

7.9 Correlation Keys (BPMN Executable)

Correlation keys are sets of correlation properties used to associate a message to a particular instance of a process.

> **Note**
>
> Correlation keys can be created only in the PowerDesigner desktop client, but they can be selected and edited in PowerDesigner Web.

To create a correlation key, click the *Create* tool to the right of the *Correlation key* field in the *General* tab of a message flow property sheet.

To create a correlation property, click the *Create* tool to the right of the *Correlation property* field in the *General* tab of a message flow property sheet, or use the tools on the *Correlation Properties* tab of a correlation key.

BPMN executable correlation keys and correlation properties can have the following properties:

Table 79:

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

The following tabs are also available for correlation keys:

- *Correlation Properties* - Lists the correlation properties regrouped in the correlation key.

7.10 Messages (BPMN Executable)

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.

> **Note**
>
> Messages can be created only in the PowerDesigner desktop client, but they can be selected and edited in PowerDesigner Web.

To create a message, click the *Create* tool to the right of the *Message format* field in the *General* tab of a message flow property sheet.

Messages can have the following properties:

Table 80:

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

The following tabs are also available:

- *Definition* - Contains the following properties:

  Table 81:

Property	Description
Type	Specifies how the message format is defined. You can choose between: ○ 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. ○ 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 *XML Modeling*.
External definition	[External file and URL only] Specifies the location path to an external file or an URL.

Property	Description
Message format type	[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: ○ XML Schema ○ DTD ○ RELAX NG
Message format definition	[Embedded or External file and URL only] Specifies the content of the message.

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.

7.11 Item-Aware Elements (BPMN Executable)

Item-aware elements are variables used to store or convey information during process execution. You can associate these elements with processes, activities, and events.

> **Note**
>
> Item-aware elements can be created and added to an object only in the PowerDesigner desktop client, but their properties can be edited in PowerDesigner Web.

To create an item-aware element:

1. Open the property sheet of a process, activity, or event, and select the *Item-Aware Elements* tab.
2. Click the *Add a Row* tool to create a element, and enter an appropriate name. By default, the element is created as a property, which has only a name, code, and comment for properties.
3. [optional] Right-click the item in the list, and select *Change to* > `<Element Type>`.

The different types of element are available in the following types of object:

Table 82:

Object	Property	Data Object / Data Reference	Data Input	Data Output
Composite processes	X	X	X	X
Tasks	X		X	X
Start events / Intermediate catching events	X			X
End events / Intermediate throwing events	X		X	

Item-aware elements can have the following properties:

Table 83:

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

7.12 Sequence and Message Flows (BPMN Executable)

Sequence flows are solid lines with an arrow at one end, which link the elements in a process in the diagram or in a single pool and show the order in which they are performed. Message flows are dotted lines with an arrow at one end, which link elements in two separate pools and show the direction in which the message is sent.

For information about creating sequence and message flows, see Sequence and Message Flows (BPMN Descriptive) [page 132]. BPMN executable sequence and message flows can have the following properties:

Table 84:

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.
Keywords	Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.
Source/ Destination	Specify the objects that are linked by the flow. For sequence flows, the source object comes before the destination object in the process. For message flows, the source object emits the message and the destination object receives it.
Condition	[sequence flows] Specifies the condition that must be fulfilled for the process to take this branch following a gateway.
Message format	[message flows] Specifies the message that transits the flow (see Messages (BPMN Executable) [page 151])
Correlation key	[message flows] Specifies the correlation key used to associate the message to a particular instance of a process (see Correlation Keys (BPMN Executable) [page 150]). In a conversation diagram, each flow must have the same key as the conversation node to which it is connected.
Correlation property	[message flows] Specifies the correlation property that acts as the unique identifier for this instance of the message .

> **Note**
>
> To set a sequence flow exiting an inclusive, exclusive, or complex gateway or an activity to the default flow, right-click the flow and select *Set to Default Flow*.

7.13 Importing and Exporting BPMN 2.0 Files

PowerDesigner can import and export BPMN 2.0 files, with a particular emphasis on supporting SAP BPM and the Eclipse BPMN2 Modeler. The import and export supports business process diagrams only; conversation and choreography diagrams and their objects are not supported.

Context

PowerDesigner supports round-trip importing and exporting of SAP BPM v7.3 and higher BPMN 2.0 files. PowerDesigner Web makes your process models available for browsing and importing directly into the SAP NetWeaver Developer Studio Process Composer v7.3 EHP1 SP15 or higher.

To import a BPMN 2.0 file and create a new model, select *File* > *Import* > *BPMN2 File*. To export a BPMN 2.0 model, select *File* > *Export* > *BPMN2 File*. You can choose between exporting a standard BPMN 2.0 file, or one targeting SAP BPM, with addition model checks.

The following BPMN 2.0 objects are supported in import and export:

Table 85:

Supported Objects
• DocumentRoot, Definitions, Property, Documentation, TextAnnotation • Process • Task, GlobalTask, GlobalBusinessRuleTask, GlobalManualTask, GlobalScriptTask, GlobalUserTask, BusinessRuleTask, ManualTask, ReceiveTask, ScriptTask, SendTask, ServiceTask, UserTask • Transaction, CallActivity, SubProcess, AdHocSubProcess • Message, MessageFlow • Participant, ParticipantMultiplicity • Collaboration • Lane, LaneSet • SequenceFlow • MultiInstanceLoopCharacteristics, StandardLoopCharacteristics • Association • Group • InputOutputSpecification, InputSet, OutputSet • DataObject, DataStore, DataInput, DataOutput, DataState • DataAssociation, DataInputAssociation, DataOutputAssociation • DataObjectReference, DataStoreReference • BoundaryEvent, IntermediateCatchEvent, StartEvent, EndEvent, ImplicitThrowEvent, IntermediateThrowEvent • CancelEventDefinition, CompensateEventDefinition, ConditionalEventDefinition, ErrorEventDefinition, EscalationEventDefinition, LinkEventDefinition, MessageEventDefinition, SignalEventDefinition, TerminateEventDefinition, TimerEventDefinition • ComplexGateway, EventBasedGateway, ExclusiveGateway, InclusiveGateway ParallelGateway • Operation, Interface • BPMNDiagram, BPMNPlane, BPMNShape, BPMNEdge, Bounds, Point

The following BPMN2 objects are supported for modeling in the desktop PowerDesigner client, but are not supported for modeling in PowerDesigner Web, or for import and export:

Table 86:

Modeling Objects Not Supported in Import/Export
• ChoreographyTask, GlobalChoreographyTask, Choreography, SubChoreography, CallChoreography • CorrelationKey, CorrelationPropertyBinding, CorrelationPropertyRetrievalExpression, CorrelationSubscription, CorrelationProperty • Conversation, SubConversation, ConversationNode, ConversationLink, GlobalConversation, CallConversation

The following BPMN2 objects are not supported for modeling or in import and export:

Table 87:

Unsupported Objects
• CorrelationPropertyBinding, CorrelationPropertyRetrievalExpression, CorrelationSubscription • MessageFlowAssociation, ParticipantAssociation, ConversationAssociation • ResourceAssignmentExpression, ResourceParameter, ResourceParameterBinding, ResourceRole • InputOutputBinding • ItemDefinition • ImplicitThrowEvent • Assignment • Auditing • ComplexBehaviorDefinition • Monitoring • Relationship • Rendering • Expression, FormalExpression • PotentialOwner • HumanPerformer • Category, CategoryValue • EndPoint • Error • Escalation • PartnerEntity, PartnerRole • Resource • Signal • Extension, ExtensionAttributeDefinition, ExtensionAttributeValue, ExtensionDefinition Import

7.13.1 Importing from SAP BPM

PowerDesigner supports importing a SAP BPM v7.3 or higher process for editing in a new business process model.

Procedure

1. Start SAP NetWeaver Developer Studio and expand the *Process Modeling* folder.
2. Expand the *Process* folder, right-click a process, and select *Exporting for BPMN 2.0...*.
3. Select a file folder and name, and then click *Export*.
4. Open PowerDesigner and select *File* > *Import* > *BPMN2 File*, navigate to the file you exported from NetWeaver, select it, and click *Open* to import it.

7.13.2 Exporting to SAP BPM

PowerDesigner supports exporting a BPMN 2.0 business process diagram as a process to SAP BPM v7.3 or higher.

Context

> **Note**
>
> PowerDesigner Web makes your process models available for browsing and importing directly into the SAP NetWeaver Developer Studio Process Composer v7.3 EHP1 SP15 or higher without the need to perform an export. For further information, see SAP BPM Developer's Guide *Modeling Processes with Process Composer* at http://help.sap.com/saphelp_nw73ehp1/helpdata/en/ff/165a665c16482e9c282ce6b0e67776/frameset.htm

Procedure

1. In PowerDesigner, open your BPMN 2.0 model, and select *File* > *Export* > *BPMN2 File*.

   > **Note**
   >
   > The export will generate a single file containing only one BPMN 2.0 diagram. When using this feature, you should limit your BPM to a single business process diagram.

2. Select to export as *SAP BPM file* (we recommend that you retain the option to *Check diagram validity*) and click *OK*.

   If your diagram is error-free, select a file folder and name, and then click *Save*.

   > **Note**
   >
   > If your diagram has errors, they will be reported and the export canceled.

3. Start SAP NetWeaver Developer Studio and create a new *Process Composer Development Component* project.
4. Expand the *Process Modeling* folder, right-click the *Processes* folder, and select *Importing BPMN 2.0 diagram...*.
5. Navigate to the file you exported from PowerDesigner, select it, and click *Open* to import it.

8 BPEL4WS 1.1 and WS-BPEL 2.0

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.

Procedure

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 Generating a BPEL Model from an Analysis Model [page 174]) or reverse engineer BPEL files into a BPM (see Reverse Engineering BPEL Languages [page 176]).

   A valid BPEL model must contain 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) [page 33]) and role associations (see Role Associations (BPEL) [page 160]) 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 [page 81]) to retrieve service description objects (service providers, service interfaces, and operations). You can also create these objects from scratch (see Service Providers (BPM) [page 78], Service Interfaces (BPM) [page 86], and Operations (BPM) [page 89]).
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 [page 36]), and specify its implementation (see Process Properties [page 24]).
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 [page 27]).
8. [optional] Generate BPEL code from your BPM objects to be interpreted by orchestration engine (see Generating BPEL Code [page 175]).

8.1 Top-Level Diagrams (BPEL)

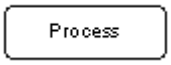
A top-level diagram is a special form of business process diagram required by BPEL models, which provides a high-level representation of a system and identifies its business partners in order to specify the scope of the system and its interactions with those partners.

> **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*.

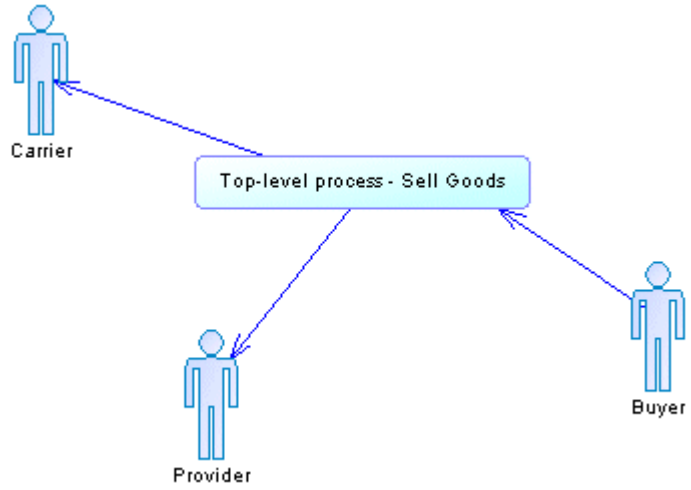
For other languages, the top-level diagram is simply the highest level of choreography diagram (see Business Process Diagrams (Analysis) [page 20]).

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

Table 88:

Object	Tool	Symbol	Description
Process			Top-level process that interacts with business partners. BPEL4WS 1.1 models top-level processes as standard processes with additional properties (see Top-Level Processes (BPEL) [page 162]).WS-BPEL models top-level processes as *empty activities* (see Stereotype Activities in Choreography Diagrams (BPEL) [page 163]).
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 Organization Units (BPM) [page 33]).
Partner link			Interaction between a top-level process and a business partner (see Role Associations (BPEL) [page 160]).

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:



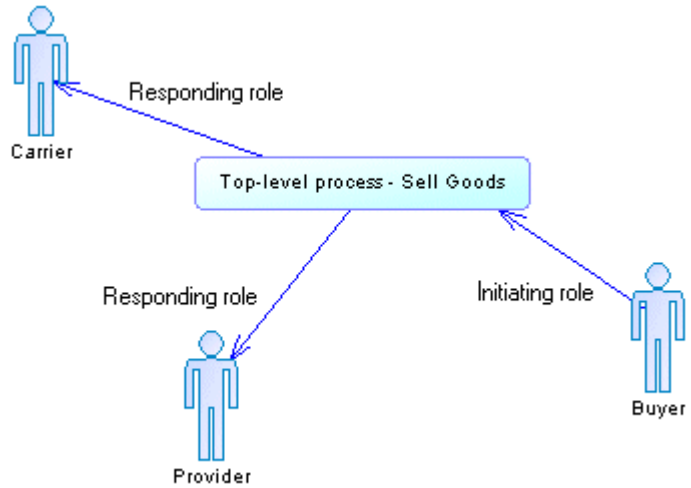
Having created a top-level diagram, you can then decompose your top-level process to create a choreography diagram (see Choreography Diagrams (BPEL) [page 163]).

8.1.1 Role Associations (BPEL)

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

You can create a role association in a BPEL4WS or WS-BPEL top-level diagram (see Top-Level Diagrams (BPEL) [page 159]).

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:



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*.

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.

Table 89:

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. You can choose between: • Initiating role – from the organization unit to the process • Responding role – rom the process to the organization unit **Note** If the orientation of the role association is not displayed, you can enable it by selecting *Tools* > *Display Preferences* > *Role Association*.
Source / Destination	Specify the organization unit or process at the extremities of the role association. 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 object property sheet by clicking the *Source* or *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.

The following properties are available on the WE-BPEL or BPEL4WS tab:

Table 90:

Name	Description
Name	Specifies the link name. Scripting name: `PartnerLinkTypeName`
Process / Organization unit role	Specify the roles played by the top-level process and the partner in the collaboration, which are by default the name of the process and the organization unit. Scripting name: `ProcessRole, OrganizationUnitRole`
Partner / Process Port Type	[WSBPEL2.0] Specify the service interfaces of the partner and process services, which are by default calculated from the invoke activities and the interface activities (receive, reply). Scripting name: `PartnerPortType, ProcessPortType`

8.1.2 Top-Level Processes (BPEL)

BPEL top-level process property sheets contain all the standard process tabs, along with the BPEL tab, the properties of which are listed below:

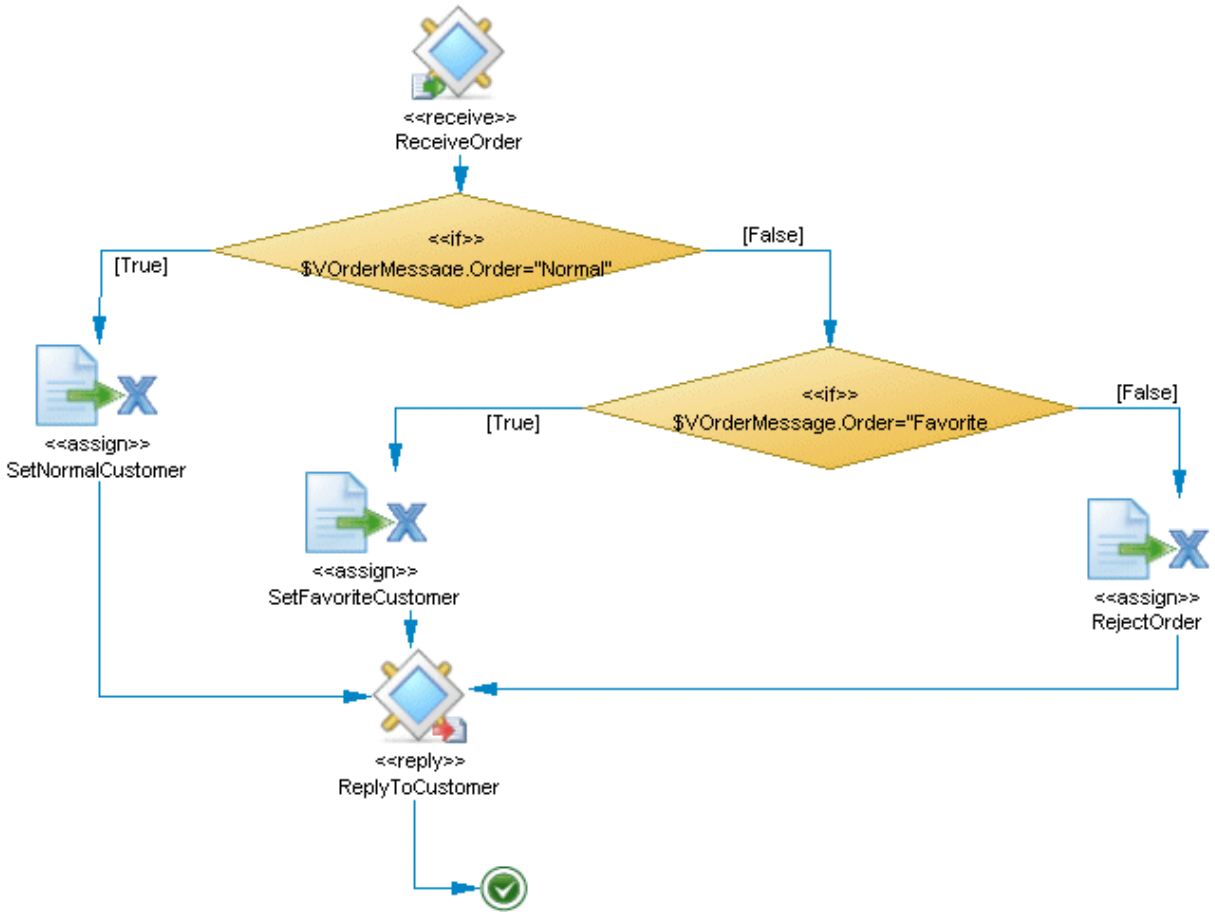
Table 91:

Name	Description
Target namespace	Specifies the target namespace of the process which is necessary in the generated file. Default value: %urnName% Scripting name: `targetNamespace`
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`
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:sublang: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`

Name	Description
Abstract process	Specifies whether the process being defined is abstract. Default value: No Scripting name: `abstractProcess`
[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`
[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`

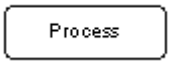
8.2 Choreography Diagrams (BPEL)

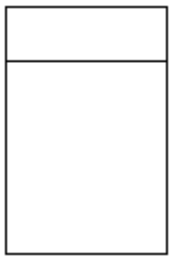
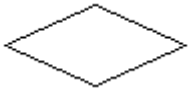
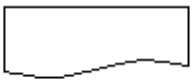
BPEL choreography diagrams help you analyze the control flow between your activities when modeling a BPEL environment.



PowerDesigner supports all the objects necessary to build BPEL choreography diagrams:

Table 92:

Tool	Symbol	Description
		Process (see Processes (BPM) [page 23]). Basis of many BPEL activities
 ...	 ...	Activity (see Activities (WS-BPEL 2.0) [page 166] and Activities (BPEL4WS 1.1) [page 168])

Tool	Symbol	Description
		Partner in scope - Specify the people, groups or organizations which are responsible for a process. PowerDesigner models partners as standard organization units (see Organization Units (BPM) [page 33]).
		Flow (see Flows (BPM) [page 47]).
		Decision (see Decisions (BPM) [page 43]).
		Synchronization (see Synchronizations (BPM) [page 45]).
		Start (see Starts and Ends (BPM) [page 41]).
		End (see Starts and Ends (BPM) [page 41]). If you need to immediately stop a business process instance use the exit activity.
None		Message format (see Message Formats (BPM) [page 50]).
None	None	Event (see Events (BPM) [page 74]).
None	None	WSDL file - Describes services provided by business partners and the way to access them. PowerDesigner models WSDL files as service providers (see Service Providers (BPM) [page 78]), port types as interfaces (see Service Interfaces (BPM) [page 86]), and operations as standard operations (see Operations (BPM) [page 89]).
None	None	XSD document (see XSD Documents (BPM) [page 88]).

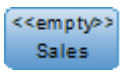
Tool	Symbol	Description
None	None	Variable (see Variables (BPM) [page 96]).
None	None	Correlation key (see Correlation Keys (BPM) [page 97]).
None	None	Data transformation (see Data Transformations (BPM) [page 99]).

8.3 Activities (WS-BPEL 2.0)

PowerDesigner provides support for all the activities defined in WS-BPEL 2.0 and provides custom symbols and tools to create them.

Table 93:

Tool	Symbol	Description
		*Scope activity* [composite] - Provides 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.
		*Sequence activity* [composite] - Specifies a set of activities that must be executed sequentially in lexical order, and acts as a container for your sequence activities.
		*Flow activity* [composite] - Specifies a set of activities that must be executed concurrently, and acts as a container for your flow activities. A link modeled as a flow with a link stereotype can express synchronization dependencies between activities.
		*While activity* [loop] - Specifies that the nested activities must be repeated until their specified condition becomes true.
		*ForEach activity* [loop] - Executes its scope for a specified count. The execution iterations can occur in parallel or in sequence.
		*RepeatUntil activity* [loop] - Executes the nested activities at least once until the specified condition becomes true.
		*Wait activity* [generate event] - Specifies a delay for a certain period of time or until a certain deadline is reached.
		*Compensate activity* [generate event] - Causes all immediately enclosed scopes to be compensated in default order.

Tool	Symbol	Description
		*Compensate scope activity* [generate event] - Causes one specified child scope to be compensated. 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.
		*Throw activity* [generate event] - Specifies a business process which needs to signal an internal fault explicitly.
		*Rethrow activity* [generate event] - Rethrows the fault that was caught by the immediately enclosing <catch> and <catchAll> elements within a <faultHandlers> element.
		*Receive activity* [execute operation] - Specifies that the business process waits for a matching message to arrive. For receive, reply, reply fault, and invoke activities use the *Assignments* tab to copy the values of message variables or to calculate the value of an expression, and store it in a variable.
		*Reply activity* [execute operation] - Sends a response to a request accepted by a receive activity.
		*Reply fault activity* [execute operation] - Sends a faulted response to a requests accepted by a receive activity.
		*Invoke activity* [execute operation] - Calls a Web service.
		*Assign activity* - update the values of variables with new data. The assign activity should be composed of at least one atomic assign task (see Process Properties [page 24]).
		*Validate activity* - Validates the values of variables against their associated XML and WSDL data definition. Specify variables (see Variables (BPM) [page 96]) in the *ValidatedVariables* tab.
		*Extension activity* - An activity that is not defined by the specification.
		*Empty activity* - An activity that does nothing (for example a fault which needs to be caught and suppressed). Can also provide a synchronization point in a flow activity, and is used to model top-level processes in top-level diagrams (see Top-Level Diagrams (BPEL) [page 159]).
		*If activity* - Selects one activity to execute from a set of activities. You can also create composite processes with an `<<if>>` stereotype.

Tool	Symbol	Description
		*Pick activity* - Waits for the occurrence of one event from a set of events, then executes the activity associated with that event.
		*Exit activity* - Immediately ends the business process instance.

8.4 Activities (BPEL4WS 1.1)

PowerDesigner provides support for all the activities defined in BPEL4WS 1.1.

BPEL4WS 1.1 activities are based on standard BPM objects with additional properties (see BPEL4WS 1.1 Object Properties [page 172]):

- *Scope activity* [composite] - Composite process (see Processes (BPM) [page 23]).
- *Sequence activity* [composite] - Composite process with a **<<sequence>>** stereotype.
- *Flow activity* [composite] - Synchronization (see Synchronizations (BPM) [page 45]) with a **<<split>>** or **<<join>>** stereotype. To gather your flow activities in a container, use a composite process with a **<<flow>>** stereotype and create a nested flow activity.
- *While activity* [loop] - Composite process with **Loop** implementation type and **While** loop type.
- *Wait activity* [generate event] - Process with **Generate event** implementation type and **Timer** type event.
- *Compensate activity* [generate event] - Process with **Generate event** implementation type and **Compensation** type event.
- *Throw activity* [generate event] - Process with **Generate event** implementation type and a **Fault** type event.
- *Receive activity* [execute operation] - Process with **Execute operation** implementation type and **Receive request** action type.
- *Reply activity* [execute operation] - Process with **Execute operation** implementation type and **Reply** action type.
- *Reply fault activity* [execute operation] - Process with **Execute operation** implementation type and **Reply fault** action type.
- *Invoke activity* [execute operation] - Process with **Execute operation** implementation type and **Invoke operation** action type.
- *Assign activity* - Process with **<<assign>>** stereotype.
- *Empty activity* - Process with no implementation type or stereotype.
- *If activity* - Decision (see Decisions (BPM) [page 43]) with a **<<switch>>** stereotype.
- *Pick activity* - Decision with a **<<pick>>** stereotype.
- *Exit activity* - End (see Starts and Ends (BPM) [page 41]) with a **<<terminate>>** stereotype.

8.5 Messages (BPEL)

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) [page 50]) with additional properties (see WS-BPEL 2.0 Object Properties [page 169] and BPEL4WS 1.1 Object Properties [page 172])
- *Parameters* - identify subdivisions of messages. PowerDesigner models messages as standard message parts (see Message Parts (BPM) [page 52]) .
- *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) [page 96]).
- *Properties* - refer to any parts of a variable. PowerDesigner models properties as standard variables (see Variables (BPM) [page 96]).
- *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 (BPM) [page 99]).
- *Correlations* - specify groups of properties that, taken together, serve to identify a message. PowerDesigner models correlations as standard correlation keys (see Correlation Keys (BPM) [page 97]).
- *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 (BPM) [page 88]). You can create an XSD document from the service provider property sheet or import or reverse engineer a WSDL to obtain one.

8.6 WS-BPEL 2.0 Object Properties

WS-BPEL 2.0 object property sheets contain additional properties on the *WS-BPEL* tab.

Table 94:

Name	Description
Isolated	[`scope`, `sequence`, `flow` and `if`] Provides concurrency control in governing access to shared variables, through a *serializable scope*, which must not be nested, and must be a leaf scope. Scripting name: `Isolated`
Validate	[`Assign`] Specifies that the assign activity validates all the variables being modified by the activity. Scripting name: `Validate`

Name	Description
Counter name/ Start counter expression / Final counter expression	[`forEach`] Specify the name of the implicit variable used to store the counter of the loop, together with its initial and final values, which are evaluated when the activity starts. During each repetition, the counter variable is implicitly declared in the activity's child scope. Scripting name: `CounterName, StartCounterExpression, FinalCounterExpression`
Parallel	[`forEach`] Specifies that the activity is parallel. By default, it is serial. Scripting name: `Parallel`
Successful branches only	[`forEach`] Specifies to count only those scopes that have completed successfully. The `<branches>` element of the forEach activity represents an unsigned-integer expression used to specify a completion condition. Scripting name: `SuccessfulBranchesOnly`
Correlations	[`invoke`] Specifies the correlations used by the invoke activity. You can choose between: • Multiple correlation - Disables the other correlation properties and displays the *Further Correlations* tab, on which you can specify any number of correlations keys. • First and second correlation patterns (each of which can have the values request, request-response, or response) and initiate input and output correlations (each of which can have the values yes, no, or join. Scripting name: `MultipleCorrelation, InCorrelationPattern, InCorrelationInitiate, OutCorrelationPattern, OutCorrelationInitiate`
Create instance	[receive and pick] Specifies the instanciation of the process. Scripting name: `CreateInstance`
Initiate correlation	[`receive`, `reply` and `reply fault`] Specifies the value of the initiation of the correlation, which can be `join`, `no`, `yes`. Scripting name: `InitiateCorrelation`
Exit on standard fault	[`scope`, `sequence`, `flow` and `if`] Specifies that the process must exit immediately. Scripting name: `ExitOnStandardFault`

Name	Description
Common properties	The properties common to all WSBPEL2.0 objects: • Expression language - Specifies the expression language used in expressions, which is by default: `urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0` • Join condition - Used to specify requirements about concurrent paths reaching an activity. The default value for XPath is the logical OR of the link status of all incoming links. • Suppress join failure - Specifies that 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. Scripting name: `expressionLanguage, joinCondition, suppressJoinFailure`

The following properties are available on the *Extended Attributes* tab:

Table 95:

Name	Description
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`
Imported WSDL content	[WSDL file] Specifies the content of the original reversed WSDL file. Scripting name: `ImportedWsdlContent`
Namespace prefix to use	[XSD document and WSDL file] Specifies the XML prefix used by the process to reference the schema definition or service provider. Default variable: %wsdlUsedPref% Scripting name: `PrefixForUse`

8.7 BPEL4WS 1.1 Object Properties

BPEL4WS 1.1 object property sheets contain additional properties on the *BPEL4WS* tab.

Table 96:

Name	Description
Create instance	[`receive`] Specifies the instanciation 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`
First correlation pattern	[`invoke`] When the first correlation is used by the invoke activity, you can choose between one of the following values: • `in` • `in-out` • `out` Scripting name: `InCorrelationPattern`
Imported WSDL content	[WSDL file] Specifies the content of the original reversed WSDL file. Scripting name: `ImportedWsdlContent`
Initiate correlation	[`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: • `true`, `false` Scripting name: `InitiateCorrelation`

Name	Description
Join condition	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. Scripting name: `joinCondition`
Multiple correlation	Specifies a BPEL Invoke, Receive or Reply using multiple correlation keys. Scripting name: `MultipleCorrelation`
Name	[partner link type] Specifies the name of the partner link type. Scripting name: `PartnerLinkTypeName`
Namespace prefix to use	[WSDL file] Specifies the XML prefix used by the process to reference the service provider. Default value: `%wsdlUsedPref%` Scripting name: `prefixForUse`
Organization unit role	[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. Scripting name: `OrganizationUnitRole`
Process role	[partner link type] Specifies the role played by the process in the collaboration. When undefined, the generated role is the name of the process. Scripting name: `ProcessRole`
Second correlation pattern	[invoke] When the second correlation is used by the invoke activity, you can choose between one of the following values: • `in` • `out-in` • `out` Scripting name: `OutCorrelationPattern`
Suppress join failure	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. Default value: `No` Scripting name: `suppressJoinFailure`

Name	Description
Variable access serializable	[`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. Default value: `No` Scripting name: `variableAccessSerializable`

8.8 Generating a BPEL Model from an Analysis Model

You can generate from an analysis to a BPEL model in order to model the implementation of the processes. For example, once the Analysis team has designed the analysis model, the model can be submitted to the Development team for implementation. You can propagate subsequent changes made to the source model by repeating the generation and selecting the *Update Existing Model* option.

Procedure

1. Select *Tools* > *Generate Business Process Model* (*Ctrl+Shift+B*) to open the BPM Generation Options dialog:
2. On the *General* tab, select a radio button to generate a new or update an existing model, and complete the appropriate options.

   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*.
3. [optional] Click the *Detail* tab and set any appropriate options. We recommend that you select the *Check model* option to check the model for errors and warnings before generation (see Checking a BPM [page 195]).
4. [optional] Click the *Target Models* tab and specify the target models for any generated shortcuts.
5. [optional] Click the *Selection* tab and select the objects to generate. By default, all objects are generated.
6. Click *OK* to begin the generation.

   The following transformations are executed to make the model compliant with BPEL and logged in the *Output* window:
   - Top-level processes - When a graph of activities is defined under a package or a model, a top-level process is created and the whole graph of activities is moved under it. A top-level process is created for each unrelated set of activities. 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.
   - Starts - If multiple starts appear in a diagram, they are merged into one.
   - Shortcuts - Can be used in analysis BPMs to reuse processes, but are replaced in the orchestration BPM with a copy of the target object. 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.

- Flow message formats - The association of message formats with flows are not supported, as the exchange of information is not managed by flows, and so message formats are detached from flows.
- Flow types - The analysis Timeout, Technical Error, and Business Error flow types are replaced with event objects with Timer or Fault stereotypes, and the events are associated with the flows to define event handlers.
- Data - Data objects are replaced with variables, and any data attachments to flows or message formats is lost.

8.9 Generating BPEL Code

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.

Procedure

1. Select *Language* > *Generate BPEL4WS 1.1 [or WS-BPEL 2.0] 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 Checking a BPM [page 195]).
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 the appropriate value for the **`Generate WSDL files`** generation option:.
   - *Local* - Generates the .WSDL file into a separate local file which is referenced into the .BPEL definition file via the [Import] clause.
   - *Import* - [default] Generates an [Import] clause into the .BPEL definition file.
   - *Embedded* - Generates a .WSDL file into the .BPEL definition file, and generates a .WSDL file for each service provider.
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 > Generated Files (Profile)*.
6. Click *OK* to begin generation.

   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.

> **Note**
>
> You can attach an extension file (.XEM) to your model to extend the generation process (see *Customizing and Extending PowerDesigner > Extension Files > Generated Files (Profile) > Generating Your Files in a Standard or Extended Generation*).

8.10 Reverse Engineering BPEL Languages

You can reverse engineer .BPEL files, WSDL files, and .XML files containing a BPEL definition into a BPM. The WSDL definitions contained in the .BPEL files are reversed into service providers.

Context

> **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.

Procedure

1. Select *File* > *Reverse Engineer* > *Process Language*, select a process language, and click *OK*.

   > **Note**
   >
   > Alternatively, in an existing BPM targeting a BPEL language, 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 muti-select files to reverser engineer using the *Ctrl* or *Shift* keys. All files will be reversed in the same BPM.
5. [optional] On the *Options* tab, select *Create XML Model* if you want to create an XML model (see *XML Modeling*) 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 detailed information about merging models, see *Core Features Guide* > *Modeling with PowerDesigner* > *Comparing and Merging Models*.

7. Click *OK* to close the dialog.

   The objects are added to your model.

9 Simulating a Business Process Model with SIMUL8

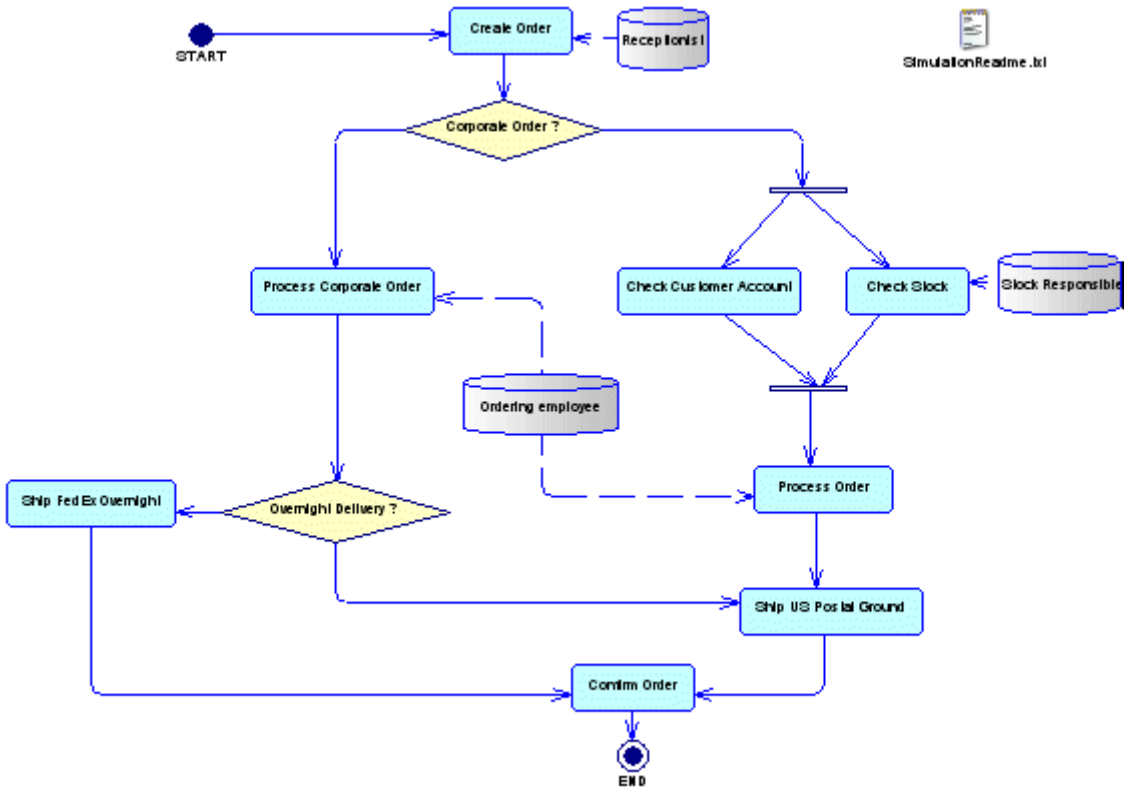
SIMUL8 is a flow simulation program that lets you view your process in action, showing how its control flow moves around the organization, revealing bottlenecks, over-utilized resources, or under-resourced elements. PowerDesigner supports the principal objects and parameters for SIMUL8 version 9.0 and higher.

> **Note**
>
> You can simulate any BPM, but for the best results, we recommend that you simulate Analysis language BPMs only (see Analysis BPM [page 18]).

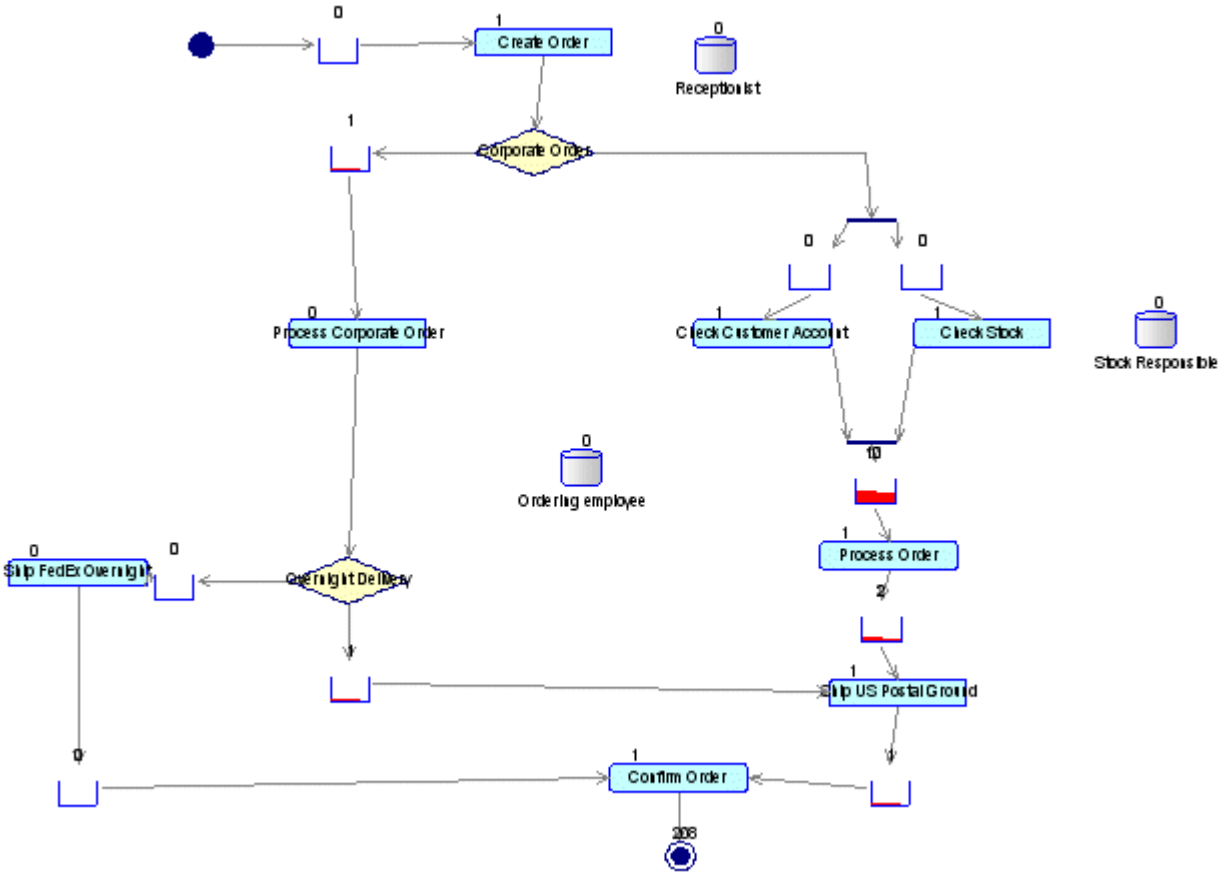
Simulation helps you to better understand the expected performance of your business processes before their implementation, by providing you with useful analysis metrics and assistance in business process optimization, and is most useful when:

- 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.

The following example shows the `Create Order` 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.

For detailed information about SIMUL8, see http://www.SIMUL8.com).

9.1 Modeling for Simulation

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

1. Create an analysis BPM with a business process diagram, and attach the SIMUL8 extension to the new model (see Creating a BPM [page 7]).
2. Populate your diagram by creating a choreography of objects, such as processes, resources, flows, decisions, etc. (see Business Process Diagrams (Analysis) [page 20]).
3. [optional] Review each object's default simulation properties (see Reviewing SIMUL8 Default Properties [page 182]), and edit them if appropriate.
4. Export your BPM diagram to SIMUL8, and then run a simulation (see Exporting a BPM to SIMUL8 [page 183]).

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 [page 184]). Repeat this step until satisfied with your results.
6. Re-import your simulated BPM to PowerDesigner to synchronize the SIMUL8 changes with your BPM, and, if necessary, create additional modeling objects (see Synchronizing SIMUL8 Changes Back to PowerDesigner [page 186]). Repeat steps 4 to 6 until you are satisfied with your BPM.
7. [optional] Generate code for BPEL to model the implementation of your processes (see Generating BPEL Code [page 175]).

BPM / SIMUL8 objects conversion

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

Table 97:

BPM object and properties	SIMUL8 object and properties
Diagram with Window color display preference	*Model* with Fill color property.
[No equivalent in BPM]	*Work item* - 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 Processes (BPM) [page 23])	*Work center* - specifies the place where the work is performed (see SIMUL8 Work Center Properties [page 187]).
Composite process (see Processes (BPM) [page 23])	*Component* - specifies a single object containing one or more existing standard objects or other components.
Implemented by process (see Processes (BPM) [page 23])	*Component* [if the process is implemented by a composite process] or *work center*.
Resource (see Resources (BPM) [page 66])	*Resource* - are required at work centers in order for the work center to work on a work item (see SIMUL8 Resource Properties [page 189]).
Start (see Starts and Ends (BPM) [page 41])	Can be either a : • *Work entry point* - Where work to be done appears in your simulation for the first time (see SIMUL8 Work Entry Point Properties [page 190]). • *Work center* with a Zero working time [if the start is contained in a composite process] see (SIMUL8 Work Center Properties [page 187]).
End (see Starts and Ends (BPM) [page 41])	Can be either a : • *Work exit point* - Where work that is complete leaves your simulation (see SIMUL8 Work Exit Point Properties [page 191]). • *Work center* with a Zero working time [if the end is contained in a composite process] (see SIMUL8 Work Center Properties [page 187]).

BPM object and properties	SIMUL8 object and properties
Flow (Flows (BPM) [page 47])	*Routing in/out* properties of a work center - specifies the path taken by each individual work item through the simulation (see SIMUL8 Route Properties [page 192]).
[No equivalent in BPM]	*Queue* - 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 SIMUL8 Route Properties [page 192]). A queue is generated for each link between SIMUL8 objects, except for work centers generated from decisions or synchronizations.
Resource flow (see Resource Flows (BPM) [page 67]) with Read access property	*Required resource property* of a work center - specifies a resource that must be available before a work center can start processing a work item (see SIMUL8 Required Resource Properties [page 189]).
Decision (see Decisions (BPM) [page 43])	*Work center* [without queue] (see SIMUL8 Work Center Properties [page 187]).
Synchronization (see Synchronizations (BPM) [page 45])	*Work center* [without queue] (see SIMUL8 Work Center Properties [page 187]).

> **Note**
>
> PowerDesigner free symbols are preserved in SIMUL8, but organization units, files, packages, message formats, parts and data are not supported.

9.1.1 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:

Table 98:

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

Domain	Property to review
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.

9.2 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.

9.2.1 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.

Context

Procedure

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 [page 184]).

   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.

Results

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

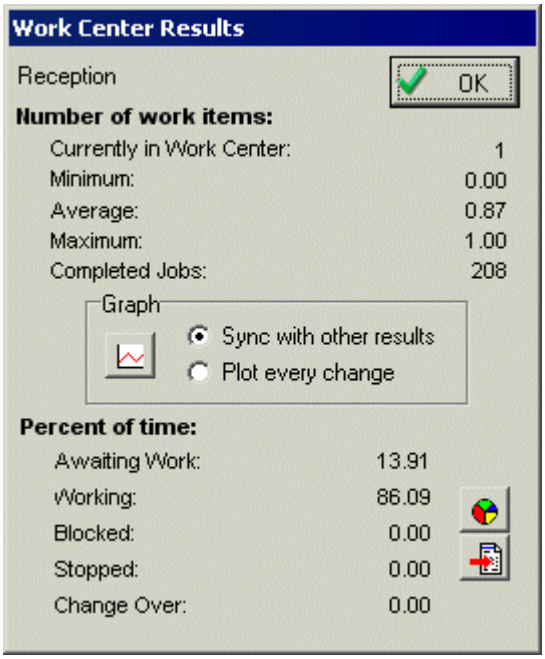
9.2.2 Analyzing Results and Fine-Tuning the Simulation

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.

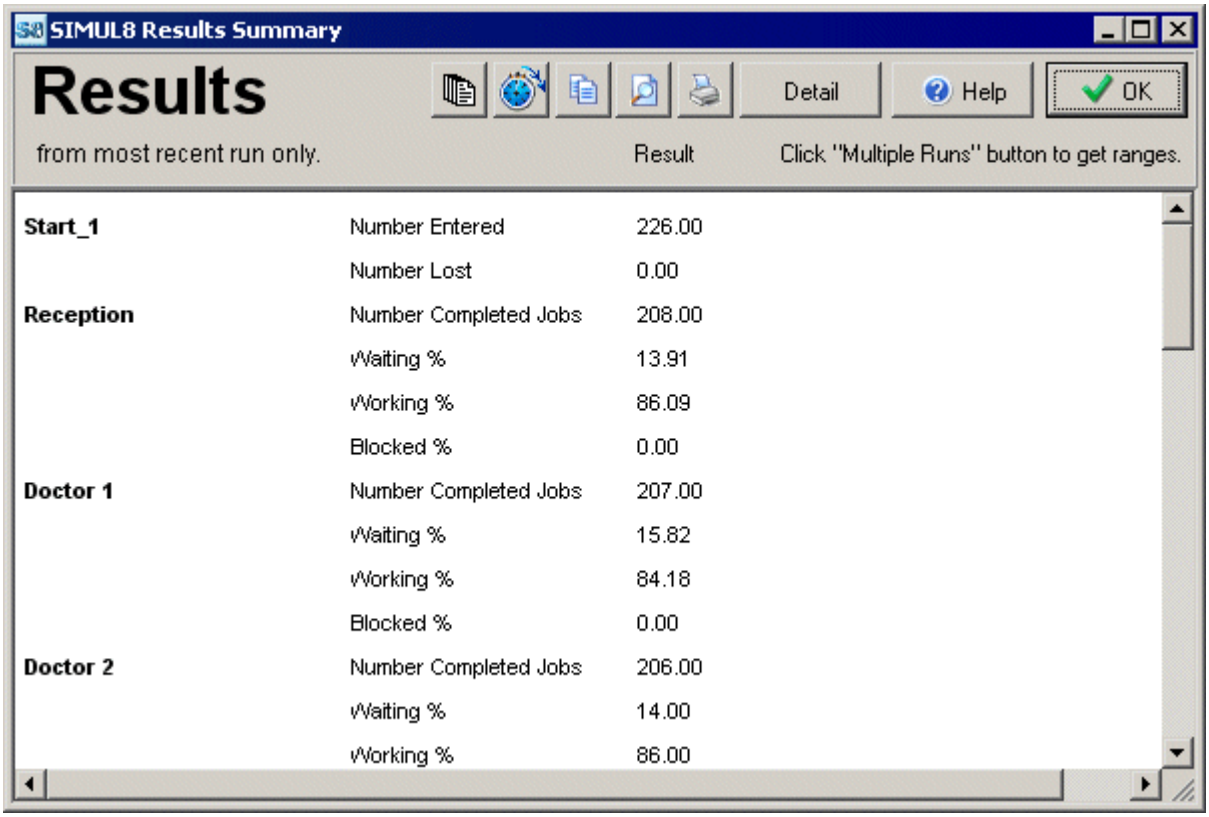
Results Analysis

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 [page 186]).

9.2.3 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.

Procedure

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* > *Modeling with PowerDesigner* > *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.

9.2.4 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.

Procedure

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.

9.3 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:

Table 99:

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`

Name	Description
Cost by time unit / Cost by unit	Specify the usage cost of the process by time unit and work unit. Default value: 0, 0 Scripting name: `FinanceCostByTimeUnit, FinanceCostByUnit`
Priority	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. Default value: 50% Scripting name: `ResourcePriority`
Release	[If unchecked] The resource must wait for the work item, if the process cannot send the work item to the next simulation object. Default value: `true` Scripting name: `ResourceRelease`
Distribution	Specifies a method for simulating the variations that occur in timing in the process: • `Average` [default] • `Exponential` • `Fixed` • `Normal` • `Uniform` Scripting name: `TimingDistribution`
Lower / Upper bound	Specify the lower and upper bounds 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, 11 Scripting name: `TimingBoundLower, TimingBoundUpper`
Standard deviation	Specifies the normal timing distribution type. For the average distribution type, the standard deviation value is set to: average value / 4. Default value: 0 Scripting name: `TimingStandardDeviation`

9.4 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:

Table 100:

Name	Description
Required resource	Specifies the way in which the resource is used by the work center: • `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. Scripting name: `ResourceRequire`
Minimum / Maximum number of resources	Specify the minimum and maximum numbers of this type of resource required at the process. Modify the minimum value if you need more than one unit of this resource to perform tasks at this process, and the maxiumum value, if the process can work faster with more resources. Default value: 1, 1 Scripting name: `ResourceMinNumber, ResourceMaxNumber`

9.5 SIMUL8 Resource Properties

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

The *Simulation* tab contains the following properties:

Table 101:

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

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: `FinanceCostByUnitByUnitTime`

9.6 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:

Table 102:

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: • `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 / Upper bound time	Specify the lower and upper bounds 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, 11 Scripting name: `InterArrivalTimeBoundLower`, `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`

9.7 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:

Table 103:

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`
Capital cost	Specifies the capital cost. Financial results can be viewed in SIMUL8, by selecting *Finance* > *Income Statement*. Default value: 0 Scripting name: `FinanceCapitalCost`

Name	Description
Revenue per unit	Specifies the revenue per unit. Default value: 0 Scripting name: `FinanceRevenuePerUnit`

9.8 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:

Table 104:

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`
Min wait time	Specifies the minimum time a work item must stay in the queue. Default value: 0 Scripting name: `QueueMinWaitTime`

Name	Description
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: `QueueFinanceCapitalCost`
Finance cost by time unit	Specifies the usage cost of the queue by unit and time unit. Default value: 0 Scripting name: `QueueFinanceCostByTimeUnit`

9.9 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:

Table 105:

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 propery 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`
Finance currency symbol	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*. Default value: $ Scripting name: `FinanceCurrencySymbol`

Name	Description
Finance overhead cost / revenue	Specify fixed costs and revenues. Non-object based costs and revenues will be included in financial results, which can be viewed in SIMUL8, by selecting *Finance* > *Income Statement*. Default value: 0, 0 Scripting name: `FinanceOverheadCost, FinanceOverheadRevenue`

10 Checking a BPM

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* > *Modeling with PowerDesigner* > *Objects* > *Checking Models*.

10.1 Package Checks

PowerDesigner provides default model checks to verify the validity of packages.

Table 106:

Check	Description and Correction
Existence of several data with same definition object	Several data should not be linked to the same definition object within the same namespace. • Manual correction: Link the data to different definition object from the data property sheet • Automatic correction: None

10.2 Process Checks

PowerDesigner provides default model checks to verify the validity of processes.

Table 107:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Missing incoming flow / Missing outgoing flow	Activities/processes must have at least one incoming and one outgoing flow. • Manual correction: Add any missing flows to the process • Automatic correction: None
Missing start / Missing end	Sub-processes/composite processes must have at least one start and at least one end. • Manual correction: Add a start and an end in the sub-process diagram • Automatic correction: None
Invalid implementation	Activities/processes cannot be implemented by an activity/process that is, itself, implemented. • Manual correction: Select a process which is not an implemented process • Automatic correction: None
Existence of several data with the same definition object	Several data should not be linked to the same definition object within the same namespace, as data can be created in a composite process. • Manual correction: Link the data to different definition object from the data property sheet • Automatic correction: None
Process with incoherent data accesses	The data attached to a flow should also be attached to the source and destination processes. • 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

Check	Description and Correction
Undefined data accesses	The data accesses of a process should have one of the following values: Create, Read, Update, Delete. • Manual correction: Add a data access for the data in the Data tab of the process property sheet • Automatic correction: None

10.3 Decision Checks

PowerDesigner provides default model checks to verify the validity of decisions.

Table 108:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Too few incoming or outgoing flows	Gateways/decisions must have more than one outgoing flow to represent a split (conditional branch) or more than one incoming flow to represent a merge. • Manual correction: Add any missing flows on the decision • Automatic correction: None

10.4 Synchronization Checks

PowerDesigner provides default model checks to verify the validity of synchronizations.

Table 109:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Too few incoming or outgoing flows	A synchronization must have more than one outgoing flow to represent a fork when a unique flow is split into several output flows executed in parallel or more than one incoming flow to represent a join when several input flows are joined and they wait until all flows reach the join before continuing as a unique output flow. • Manual correction: Add any missing flows to the synchronization • Automatic correction: None

10.5 Flow Checks

PowerDesigner provides default model checks to verify the validity of flows.

Table 110:

Check	Description and Correction
Missing source / Missing destination	Flows must have both a source and a destination object. • Manual correction: Assign a source or a destination to the flow • Automatic correction: None
Undefined message format	Message flows must either specify a message format or have their message format set to **<None>**. • Manual correction: Define the message format for the flow or delete it • Automatic correction: None

Check	Description and Correction
Inconsistent message format	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. • Manual correction: Add any missing message formats to the appropriate flows of the decomposed processes • Automatic correction: None
Invalid exception flow destination	A flow with Exception stereotype must target a process that is implemented by an operation and whose Action Type is Receive Request. • 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

10.6 Resource Checks

PowerDesigner provides default model checks to verify the validity of resources.

Table 111:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Unused resource	Data objects/resources must be linked to at least one activity/process. • Manual correction: Link the resource to a process • Automatic correction: None

10.7 Resource Flow Checks

PowerDesigner provides default model checks to verify the validity of resource flows.

Table 112:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Invalid source / Invalid destination	Data associations/resource flows can only go from (have as sources) and point to (have as destinations) activities/processes and data objects/resources. • Manual correction: Assign a process and a resource to the resource flow extremities • Automatic correction: None
Undefined access mode	A resource flow should have a defined access mode (Read, Create, Update or Delete). • Manual correction: Assign an access mode to the resource flow • Automatic correction: None

10.8 Organization Unit Checks

PowerDesigner provides default model checks to verify the validity of organization units.

Table 113:

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. • 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	[if glossary enabled] Names and codes must not contain synonyms of glossary terms. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Circular dependency through parent property	An organization unit cannot be the parent of itself or cannot have for parent one of its children. • Manual correction: Change the organization unit in the Parent box in the organization unit property sheet • Automatic correction: None

10.9 Start and End Checks

PowerDesigner provides default model checks to verify the validity of starts and ends.

Table 114:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Missing outgoing flow / Missing incoming flow	Starts must have at least one outgoing flow. Ends must have at least one incoming flow. • Manual correction: Create a flow from the start or a flow to the end. • Automatic correction: None

10.10 Message Format Checks

PowerDesigner provides default model checks to verify the validity of message formats.

Table 115:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Message format definition uniqueness	Message format definitions should be unique in the model. • Manual correction: Delete the duplicate message format definition • Automatic correction: None

10.11 Data Checks

PowerDesigner provides default model checks to verify the validity of data.

Table 116:

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. • 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	[if glossary enabled] Names and codes must not contain synonyms of glossary terms. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Unused data	The data you created is not used in the model. • Manual correction: Attach the data to an object in the model • Automatic correction: None

10.12 Service Provider and Interface Checks

PowerDesigner provides default model checks to verify the validity of service providers and interfaces.

Table 117:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Missing interfaces / Missing operations	Service providers must contain at least one interface. Interfaces must contain at least one operation. • Manual correction: Create a service interface in the *Interfaces* tab of the service provider property sheet or an operation in the *Operations* tab of the service interface property sheet. • Automatic correction: None.

10.13 Operation Checks

PowerDesigner provides default model checks to verify the validity of operations.

Table 118:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Missing input message / Missing output message	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 bot input and output messages. • 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

10.14 Variable Checks

PowerDesigner provides default model checks to verify the validity of variables.

Table 119:

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. • 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. • 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	Object names must be unique in the namespace. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Undefined data type	The variable should have a defined data type. • 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	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. • Manual correction: Move the variable under the common parent or duplicate it • Automatic correction: Moves the variable under the common ascendant
Data type coherence	A variable mapped to a message should be of the same type as the message. • 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

10.15 Data Transformation Checks

PowerDesigner provides default model checks to verify the validity of data transformations.

Table 120:

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. • 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. • 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	Object names must be unique in the namespace. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Empty transformation expression	The transformation expression should not be empty. • Manual correction: Define a transformation expression in the Transformation tab of the data transformation property sheet • Automatic correction: None
Empty assigned variable	The target variable of a transformation must not be undefined. • 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

10.16 Correlation Key Checks

PowerDesigner provides default model checks to verify the validity of correlation keys.

Table 121:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Missing variables	Correlation keys must contain at least one variable. • Manual correction: Attach a variable to the correlation key • Automatic correction: None

Check	Description and Correction
Correlation key used out of scope	A correlation key must be used within the scope of its definition. • Manual correction: Choose only correlation keys defined under the parent scope of the process • Automatic correction: Moves out of scope correlation keys to common ascendant and leaves a shortcut at the initial location
Unused correlation key	The correlation key should be used by an activity. • Manual correction: Use the correlation key object in a process implemented by an operation or delete the useless correlation key • Automatic correction: None

10.17 Event Checks

PowerDesigner provides default model checks to verify the validity of events.

Table 122:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Empty stereotype	An event should have a defined stereotype. • Manual correction: Define a stereotype in the Stereotype box of the event property sheet • Automatic correction: None

10.18 Choreography Task Checks

PowerDesigner provides default model checks to verify the validity of choreography tasks

Table 123:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Initiating or Responding participant missing	Each choreography task must have both an initiating and responding participant specified. • 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	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. • 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	Each choreography task must have an initiating message specified. • Manual correction: On the *General* tab of the choreography task property sheet, specify an appropriate initiating message. • Automatic correction: None

10.19 Conversation Node Checks

PowerDesigner provides default model checks to verify the validity of conversation nodes.

Table 124:

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. • 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. • 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. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.
Inconsistent participants with parent node	A sub-node must not be attached to participants that are not attached to its parent node. • 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	Each conversation node must have a correlation key specified. • Manual correction: Specify a correlation key in the conversation node property sheet. • Automatic correction: None

10.20 Communication Link Checks

PowerDesigner provides default model checks to verify the validity of communication links.

Table 125:

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. • 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. • 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	Object names must be unique in the namespace. • Manual correction: Modify the duplicate name or code. • Automatic correction: Appends a number to the duplicate name or code.

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).

www.sap.com/contactsap