

webMethods BPM Process Development Help

Version 10.7

October 2020

This document applies to webMethods Process Development 10.7 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2007-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: PD-OLH-107-20230908

Table of Contents

About this Guide.....	9
Deprecation of webMethods Broker.....	10
Document Conventions.....	10
Online Information and Support.....	11
Data Protection.....	11
1 Designer Workspace.....	13
Designer Workspace.....	14
2 Process Development Views.....	15
About Process Development Views.....	16
3 Process Development Preferences.....	33
About Process Development Preferences.....	34
Configuring Process Development Preferences.....	34
About Capabilities Preferences.....	35
Configuring the Process Development Mode.....	35
Configuring Appearance Preferences.....	36
Configuring Colors And Fonts Preferences.....	39
Configuring Build and Upload Preferences.....	40
Configuring Optimize Server Preferences.....	42
Configuring Process Audit Database Preferences.....	43
Creating a Custom Image Set for Activity Steps.....	45
4 Process Projects.....	47
About Process Projects.....	48
5 Processes.....	51
About Processes.....	52
About Process Development Modes.....	52
Legacy Processes and BPMN.....	53
Creating a Process.....	56
Configuring a Process.....	57
Updating Process Model Versions.....	57
Troubleshooting a Process.....	58
Basic Process Properties.....	58
Advanced Process Properties.....	61
About Synchronizing Process Runtime Settings with webMethods Monitor.....	66
Deleting a Process.....	67
Printing a Process.....	67
Saving a Process Image.....	68
Copying a Process.....	68

Using E-forms in a Process.....	69
6 Process Steps.....	71
Process Step Overview.....	72
About Step Labels.....	72
Moving Process Steps Using the Keyboard.....	73
Cutting or Copying Process Steps.....	73
Pasting Process Steps.....	74
Deleting a Process Step.....	75
Enabling a Step for Resubmission.....	76
Assigning Custom Step Ornaments.....	77
7 The Process Editor.....	79
Using the Process Editor.....	80
Using the Palette.....	80
Working with Speed Buttons.....	80
Resizing Objects on the Canvas.....	81
Moving Process Steps Using the Keyboard.....	82
Changing an Activity, Event, or Gateway Type.....	83
Using the Canvas Clipboard.....	84
Using Keyboard Shortcuts.....	85
Toolbar Buttons.....	86
8 Step Inputs and Outputs.....	89
About Inputs and Outputs.....	90
9 Rules.....	99
About Rules.....	100
About webMethods Business Rules.....	100
About Rule Tasks.....	102
Using webMethods Business Rules in Processes.....	102
About Process Actions.....	102
About Manual Decisions.....	103
Creating a Process Action.....	103
Configuring a Remote Integration Server.....	105
10 Process Engine Processing.....	107
About Process Execution.....	109
About Transitions.....	117
About Looping.....	128
About Process Logic.....	130
About Joins.....	131
About Process and Step Timeouts.....	138
Subprocess Concepts.....	145
Call Activity Concepts.....	147
Handling Process and Step Errors.....	156
Process Cancellation.....	159

Process Suspension.....	160
Process Debugging.....	160
About Document Correlation.....	161
Creating Correlation Services.....	163
Correlation Service Input and Output Variables.....	163
Specifying Correlations.....	165
About Document Merging.....	166
About Process Tracking.....	166
Process Logging Behavior.....	169
About Expression Operators.....	173
11 Process Engine Services.....	175
Process Engine Built-In Services Location.....	176
Summary of Elements in the WmPRT\pub Folder.....	176
12 Activity Steps.....	211
About Activities.....	212
About Task Types.....	214
About Abstract Tasks.....	221
About Service Tasks.....	222
About User Tasks.....	225
About Manual Tasks.....	228
About Rule Tasks.....	230
About Send Tasks.....	233
About Receive Tasks.....	237
About Subprocesses.....	242
About Call Activities.....	253
13 BPMN Event Steps.....	263
About BPMN Events.....	264
Adding an Event to a Process.....	265
Removing an Event from a Process.....	267
About Start Events.....	267
About Intermediate Events.....	275
About End Events.....	298
Basic Event Properties.....	310
14 EDA (Deprecated) Event Types.....	315
About EDA Event Types.....	316
About EDA Event Type Storage.....	317
Adding a Custom EDA Event Type to a Message Step.....	318
Adding an EDA Event Type to an Activity Step.....	320
Enabling and Disabling Predefined EDA Event Emission for a Process Model.....	321
About EDA Event Types and Integration Server Document Types.....	322
About EDA Event Types and Logged Fields.....	323
Importing the EDA Predefined Event Type Project.....	324
About EDA Predefined Process Event Types.....	325

15 Gateway Steps.....	339
About Gateways.....	340
About Gateway Types.....	341
Adding a Gateway.....	343
Configuring a Gateway.....	344
Changing the Gateway Type.....	344
Removing a Gateway.....	344
Basic Gateway Properties.....	345
Advanced Gateway Properties.....	347
16 Stages and Milestones.....	351
About Process Stages and Milestones.....	352
About Milestones.....	352
Working with Stages.....	353
17 Pools and Swimlanes.....	359
About Pools.....	360
About Swimlanes.....	362
18 Notes and Annotations.....	367
About Notes and Annotations.....	368
Working with Notes.....	368
Working with Annotations.....	369
19 KPIs.....	371
About KPIs.....	372
20 Process Documentation.....	379
About Process Documentation.....	380
Adding Documentation Fields.....	380
Assigning Documentation Field Values.....	380
Removing Documentation Fields.....	381
Generating Documentation Reports.....	381
Configuring Default Documentation Fields Preferences.....	382
21 Working with Escalation Processes.....	385
Escalation Process Overview.....	386
Importing Closed Loop Analytic Assets into Designer.....	386
Activating the WmClosedLoopAnalytics Package.....	386
Working with the Default Escalation Process.....	387
Working with the Default Escalation Task.....	388
Working with the Default webMethods Rule Projects.....	388
Mapping Data Fields in the Wrapper Services.....	388
Triggering a Test Escalation Process.....	390

22 Process Integration with ARIS	393
ARIS Process Integration.....	394
Configuring ARIS Server Preferences.....	397
About Importing ARIS Processes as XPDL Files.....	397
About Importing ARIS Solutions.....	402
Synchronizing ARIS Processes.....	404
23 Importing and Exporting Processes	405
About Importing and Exporting Processes.....	406
BPMN to XPDL Mapping.....	420
24 Building and Uploading Processes	433
About Building and Uploading a Process.....	434
Generating a Process.....	437
About Process Generation and Stage Status Display.....	440
About Mapping Services.....	441
Working with Process Versions.....	442
Working with Subscription Filters.....	444
Working with Triggers.....	446
Working with Protocols.....	448
About Enabling Processes.....	453
Setting Default Deployment Values for a Process Model.....	453
25 Debugging Processes	455
About Process Debugging.....	456
Creating a Process Debug Configuration.....	459
Debugging a Process.....	460
Modifying Input Data.....	470
Working with Breakpoints.....	472
26 Working with CentraSite Metadata for BPM and CAF Assets	477
About Metadata in CentraSite.....	478
Publishing Metadata for Processes and CAF Applications.....	481
Retracting Metadata for Processes and CAF Applications.....	482
About Searching for Asset Metadata.....	483
Performing a Basic Search for webMethods Assets.....	484
Performing an Advanced Search for webMethods Assets.....	485
Configuring Search Preferences.....	486
About the Search View.....	487
About Working with Saved Searches.....	490
Working with Saved Searches.....	491
27 Process Simulation	497
About Process Simulation.....	498

About this Guide

■ Deprecation of webMethods Broker	10
■ Document Conventions	10
■ Online Information and Support	11
■ Data Protection	11

This guide contains the *webMethods BPM Process Development Help* in PDF book format. The information in this guide is the same information that you can view via the Software AG Designer online help.

webMethods Process Development enables you to design, create, configure, build, upload, debug, and simulate complex processes.

Deprecation of webMethods Broker

webMethods Broker is deprecated for use with webMethods 10.2. If you are starting development using webMethods 10.2, you should use webMethods Universal Messaging instead of webMethods Broker. If you are upgrading to webMethods 10.2, you should consider migrating to Universal Messaging. If you choose to continue to use webMethods Broker, you will still be fully supported, but only until the announced end-of-life dates for webMethods Broker. For details, see <https://empower.softwareag.com/brokerendoflife/>

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Designer Workspace

■ Designer Workspace	14
----------------------------	----

Designer Workspace

When you start Designer the first time, you are prompted to select a workspace. You can accept the default location (`c:\Documents and Settings\\workspace90`), or choose a different location.

Stored in this directory are:

- Projects (process projects, task projects, etc.)
- Folders inside projects
- Processes and local process assets (the workspace index)
- The log (in `<workspace directory>/.metadata/.log`)
- Preferences settings

If you switch workspaces using **File > Switch Workspace** and choose a new directory, you will no longer see the same items as above. Each workspace contains its own set of projects, processes, preferences, and local metadata.

By default, Designer prompts you for a workspace every time you start it. You can choose to have it accept a default workspace and not prompt you. You can configure this option at startup by checking the **Use this as the default and do not ask again** box in the Workspace Launcher window, or at any other time by going to **Window > Preferences > General > Startup and Shutdown**.

For additional workspace information, see *webMethods BPM and CAF Workspace Metadata Help*. For Eclipse-related information, refer to Eclipse Help.

2 Process Development Views

- About Process Development Views 16

About Process Development Views

Process Development views are organized into perspectives. Each perspective contains a number of associated views, and each view contains a particular type of information about the process, or asset in the process. A view is not just for viewing information; some views allow advanced editing and configuration as well.

You can add views to or remove them from perspectives, and move views to different locations in Designer. **Window > Show View >** displays the current perspective's associated views. Designer provides access to a general Software AG category, for some shared views, as well as specific categories for Software AG Process Debug and, if installed, Software AG Process Simulation.

Tip:

Eclipse does not support multiple instances of a single view in the tree structure. Views shared between Process Development and UI Development are listed in the general Software AG category.

Reset a perspective to reload all of its associated views. See Eclipse's own help topic: **Workbench User Guide > Tasks > Working with perspectives > Resetting perspectives.**

The following table lists the available views in the Process Development and Process Debug perspectives:

Designer View	Description
 Properties view	See "About the Properties View" on page 17.
 Outline view	See "About the Outline View" on page 17.
 Navigator view	See "About the Navigator View" on page 19.
 Solutions view	See "About the Solutions View" on page 19.
 Registry Explorer view	See "About The Registry Explorer View" on page 480.
 Package Navigator view	See "About the Package Navigator View" on page 30.
 Saved Searches view	See "About Working with Saved Searches" on page 490.
 Search view	See "About the Search View" on page 487.
 Build Report view	See "About the Build Report View" on page 31.
 Problems view	See "About the Problems View" on page 31.
 Error Log view	See "About the Error Log View" on page 32.
 Trace view	See "About the Trace View" on page 463.
 Pipeline Data view	See "About the Pipeline Data View" on page 471.
 Breakpoints view	See "About the Breakpoints View" on page 473.

Designer View	Description
 ARIS Tasks view	See “ The ARIS Tasks View ” on page 396.
 Rules Explorer view	See “ About the Rules Explorer View ” on page 30.

About the Properties View

The Properties view displays information about the currently selected asset in Designer, including processes, steps, pools, transitions, annotations (notes), and process simulation resources (with the Process Simulation feature). Based on the type of asset selected, the Properties view is organized into pages that allow you to see and configure specific aspects of the asset.

When you select an asset in an editor or in a view, information about it appears in the Properties view.

Tip:

If the information in the Properties view does not display properly, click anywhere in the process editor, and then click the step again.

When you first open the Process Development perspective, the Business Analyst mode is enabled by default. For more information, see “[About Process Development Modes](#)” on page 52. In Business Analyst mode, the Properties view displays only basic properties. To see advanced properties, preferences, and functions, you must select the Process Developer mode, as described in “[Configuring the Process Development Mode](#)” on page 35.

If you close the Properties view and want to reopen it, click **Window > Show View >  Properties**.

About the Outline View

The Outline view is a standard Eclipse view. In Designer, it provides a tree view of the elements that are displayed in the process editor. Use it to locate elements on the process editor's canvas. This is especially helpful when the process editor contains many and/or a complex set of elements. When you select an element in the Outline view, it becomes the selected element on the canvas. As a result, other views that are specific to the selected element, such the Properties view, also display information for the selected element.

If you have the Process Simulation feature installed, you can also use the Outline view to manage processes in an open process simulation file. The Process Simulation feature is required to view process simulation files. See “[About Process Simulation](#)” on page 498.

When you first open the Process Development perspective, the Business Analyst mode is enabled by default. For more information, see “[About Process Development Modes](#)” on page 52. In Business Analyst mode, when you select a step in the Outline view, the Properties view displays only basic properties. To see advanced properties, preferences, and functions, you must select the Process Developer mode, as described in “[Configuring the Process Development Mode](#)” on page 35.

The Outline view is shown by default in the Process Development and Process Debug perspectives. If you close the Outline view and want to reopen it, click **Window > Show View >  Outline**.

Using the Outline View

The Outline view shows elements in the process that is currently displayed in the process editor, including their names, types, and IDs. If your process contains pools, Designer groups the elements by the pools.

> To use the Outline view

1. If the Outline view is not visible, in Designer: **Window > Show View >  Outline.**
2. Expand the nodes as needed to display the hierarchical view of the process.
3. You can perform the actions described in the following table in the Outline view.

To do this	Description
Hide or show the external pools in the Outline view	<p>To hide external pools and the steps within them, click ▾ View Menu (in the upper right corner) and select Filters > Hide External Pools. Designer places a check mark next to the Hide External Pools menu item to indicate external pools are currently hidden.</p> <p>To show external pools, repeat this step by clicking ▾ Menu again and selecting Filters > Hide External Pools again. Designer removes the check mark next to the Hide External Pools menu item.</p>
Hide or show the swimlanes in the Outline view	<p>To hide swimlanes, click ▾ View Menu (in the upper right corner) and select Filters > Hide Swim Lanes. Designer places a check mark next to the Hide Swim Lanes menu item to indicate swimlanes are currently hidden.</p> <p>To show swimlanes, repeat this step by clicking ▾ View Menu again and selecting Filters > Hide Swim Lanes again. Designer removes the check mark next to the Hide Swim Lanes menu item.</p>
Display an area of the process that is not on the visible part of the process editor's canvas	<p>You can update the display of the process on the canvas to show another area of the process. To do so, click  Canvas View. Designer replaces the tree view with a thumbnail of the entire process. When the process extends beyond the visible area of the Process Development canvas, Designer displays a blue box to show the area of the process that it is currently displaying on the canvas. Drag the blue box to cover the area of the process you want Designer to display on the canvas.</p> <p>To return to the tree view, click  Tree View.</p>
Locate a step, swimlane, or pool in the process editor	Expand nodes in the tree until you find the step, swimlane, or pool and click it. The selected item is highlighted in the process editor.

To do this	Description
Locate a step in the process editor that contains a KPI	Select a  KPI nested under a step and click it. The step that contains the KPI is highlighted in the process editor.
	Note: KPIs cannot be reused in multiple steps; each step has its own KPIs.

About the Navigator View

The Navigator view is a standard Eclipse view that provides a tree view of all the assets in your workspace. You can use the Navigator view to open files for editing or select assets for operations such as exporting. If you are using Team Development, use the Navigator view to share assets in your workspace with other team members.

Important:

There is a `.config` file associated with the `.process` file. It is recommended that you manage the process using the Solutions view, which automatically handles these two files together. See [“About the Solutions View” on page 19](#).

For more information about using Team Development, see [“About Team Development and Version Control” on page 29](#). For more information about the Navigator view, see the Eclipse documentation included in the *Software AG Designer Online Help*.

The Navigator view is shown by default in the Process Development and Process Debug perspectives. If you close the Navigator view and want to reopen it, click **Window > Show View >  Navigator**.

Note:

An alternative to viewing your workspace using the Navigator view is to use the Solutions view. In the Navigator view, the tree structure lists files that are in your workspace. The Solutions view provides a simpler view that shows only your webMethods assets. In the Solutions view, the tree structure lists only the assets (for example, processes, tasks, and user interfaces) and not the individual files that make up those assets. If you have the Process Simulation feature installed, the Solutions view also shows process simulations. The Process Simulation feature is required to view process simulation files. See [“About Process Simulation ” on page 498](#).

About the Solutions View

The Solutions view shows webMethods assets in your workspace; that is, it shows processes, user tasks, rules, and user interfaces

. Assets are grouped by solution, which is a logical grouping you can form to relate the processes (including those shared with ARIS), rules projects, user tasks,

and user interfaces that you use for a solution. This enables you to visually comprehend all the assets that make up one of your solutions.

Note:

If you have the Process Simulation feature installed, the Solutions view also shows process simulations. The Process Simulation feature is required to view process simulation files. See *webMethods BPM Process Simulation Help*.

The Solutions view always contains the  **Default Solution**, which contains *all* processes, user tasks, rules, and user interfaces in your workspace. You can create additional solutions that contain only the processes, user tasks, rules, and user interfaces that you associate with it. For example, you might create a "New_Orders" solution to group the assets that handle processing orders; this will allow you to easily comprehend the processes, user tasks, rules, and user interfaces that are used to process orders. You can associate a single resource with multiple solutions. For example, you might create a "Report_Problems" process and associate it with all of your solutions.

Note:

An alternative to viewing your workspace using the Solutions view is to use the Navigator view. In the Navigator view, the tree structure lists files that are in your workspace. An asset in the Solutions view can relate to multiple files in your workspace. For example, a process is comprised of a .process file and a .config file. It is recommended that you manage the process using the Solutions view, which automatically handles these two files together.

You can perform actions for each node in the Solutions view. The actions vary based on the type of node. Use a node's context (right-click) menu to view the actions you can perform.

You can adjust and update the Solutions view tree using the  **Collapse All** and  **Refresh** buttons on its toolbar. You can also  **Import** and  **Export** files in the Solutions view.

The Solutions view is shown by default in the Process Development and Process Debug perspectives. If you close the Solutions view and want to reopen it, click **Window > Show View >  Solutions**.

Working with Process Projects and Processes in the Solutions View

The Solutions view provides a quick way to view all process projects and processes in your workspace and to view how process projects relate to your solutions. You can also perform actions for process projects and processes in the Solutions view. The **Processes** node contains process projects and the processes within them. Use a node's right-click menu to view the actions you can perform.

> To view information about your process projects and processes in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**

The Solutions view is shown by default in the Process Development and Process Debug perspectives.

2. The table below lists the information about process projects and processes that you can view in the Solutions view:

To view	Take this action in the Solutions view
All the process projects and processes in your workspace	Expand the  Processes node under  Default Solution .
The solutions with which a process project is associated	Right-click the  process project, select  Associate With Solution , and view the list of solutions. The process project is associated with all solutions for which there is a check mark next to the name.
The processes in a process project	Expand the  process project node.

Working with ARIS Processes in the Solutions View

Using the **ARIS** right-click menu in the Solutions view, you can re-import or upload an ARIS process, as well as create and upload a process Documentation Report. The **Re-Import ARIS Process** selection invokes workflow-independent functionality to re-import the ARIS process from CentraSite into Designer. The **ARIS Synchronization** selection opens a new Synchronize ARIS Process window that displays only the options that are available for your process at its given state in your workflow.

➤ To work with shared ARIS processes in the Solutions view

- In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**. The Solutions view is shown by default in the Process Development and Process Debug perspectives.
- Right-click an ARIS process model and select one of the options:
 - **Upload Documentation Report**
 - **Re-Import ARIS Process**
 - **ARIS Synchronization**
- When uploading a process, select the **Action** you want to take in the Synchronize ARIS Process window:
 - **Request review for model** to request a model review
 - **Upload for update in ARIS** to push your changes to ARIS in response to an **Initiate update**
 - **Finish implementation** to mark a model as complete
 - **Delegate implementation** to delegate the implementation to someone
- Optionally, enter a **Description** and a **Message** to accompany the selected **Action**.

5. Click **OK** to start the upload, or click **Cancel** to cancel the upload.

Working with Rules Projects in the Solutions View

The Solutions view provides a quick way to view and manage the webMethods Business Rules business rules projects in your workspace. The **Rules** node contains only webMethods rules. Use a node's right-click menu to view the actions you can perform.

> To view information about webMethods Business Rules projects in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**. The Solutions view is shown by default in the Process Development and Process Debug perspectives.
2. To view the webMethods Business Rules projects in your workspace, expand the  **Rules** node under  **Default Solution**.
3. To open a webMethods Business Rules project in the Rules Explorer view, double-click it in the Solutions view.

Creating Solutions

> To create a solution

1. Open the New Solution window. To do so, perform one of the following sets of steps:
 - From the **File** menu:

In Designer: **File > New >  Solution**.

You can create a new solution this way in the Process Development and Process Debug perspectives.
 - From the **Solutions** view:
 - In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**.

The Solutions view is shown by default in the Process Development and Process Debug perspectives.
2. Right-click any solution node in the Solutions view and select  **New Solution**.
 - a. In the New Solution window, specify the name of the solution in the **Project name** field.
 - b. Ensure the **Use default location** check box is selected.

- c. Click **Finish**.
3. In the New Solution window, specify the name of the solution in the **Project name** field.
4. Ensure the **Use default location** check box is selected. Do not clear this check box.
5. Click **Finish**.

Deleting Solutions

To delete a solution, you must delete it from the Navigator view. However, you can use the Solutions view to identify the files you need to delete.

> To delete a solution

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**.
2. Right-click the solution you want to delete and select **Show Files**. Designer opens the Navigator view and highlights the file associated with the solution you selected in the Solutions view.

Note:

You cannot delete the  **Default Solution**.

3. In the Navigator view, right-click the selected folder and select  **Delete**.
4. In the Confirm Project Delete window, select one of the following:
 - **Also delete contents under <workspace_name>** to remove the solution from your workspace and also delete the files associated with the solution from your file system.
 - **Do not delete contents** to remove the solution from your workspace, but leave the files associated with the solution in your file system.
5. Click **Yes** in the Confirm Project Delete window.

Associating Assets with Solutions

To logically group processes, process simulations, user tasks, rules, and user interfaces, you *associate* assets with a solution that you have created:

- To associate processes or process simulations with a solution, you associate the process project that contains the processes or process simulations with a solution.

Note:

The Process Simulation feature must be installed in order to work with process simulations in the Solutions view. See *webMethods BPM Process Simulation Help*.

- To associate tasks or user interfaces with a solution, you associate the composite application that contains the tasks or user interfaces with a solution.

Note:

The Solutions view always displays the  **Default Solution**, which contains *all* processes, tasks, and user interfaces in your workspace. Create additional solutions to form logical groups of processes, process simulations, tasks, and user interfaces for each of your solutions. If you have the Process Simulation feature installed, the Solutions view also shows process simulations. The Process Simulation feature is required to view process simulation files. See *webMethods BPM Process Simulation Help*.

To associate a process project with a solution

- In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
- In the Solutions view, under the  **Processes** (or  **Simulations** folder, if you have the Process Simulation feature installed), right-click the  process project that you want to associate with a solution, select  **Associate With Solution**, then select the name of the solution with which to associate the selected item.

Designer places a check mark next to the solution's name in the menu to indicate that the asset is now associated with that solution. Additionally, Designer adds the tree structure for the process project to the selected solution.

Removing Associations from Solutions

A process project contains processes. If you no longer want a process project associated with a solution, you can remove the association.

Note:

Process projects can also contain process simulations. The Process Simulation feature must be installed in order to work with process simulations in the Solutions view. See *webMethods BPM Process Simulation Help*.

> To remove a process project association from a solution

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
2. In the Solutions view, under the  **Processes** (or  **Simulations** folder, if you have the Process Simulation feature installed), right-click the  process project with which you want to remove the association, select  **Associate With Solution**, then select the name of the solution from which to remove the association.

Designer removes the check mark next to the solution's name in the menu to indicate that the asset is no longer associated with that solution. Additionally, Designer removes the tree structure for the process project from the solution.

Renaming Process Assets

You can rename process projects, processes, user tasks, rules, and user interfaces using the Solutions view. If you have the Process Simulation feature installed, you can rename process simulations, too.

Important:

Changing a process-wide property, such as a process or package name, results in Designer prompting you to regenerate the process.

➤ **To rename a process asset**

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
2. Right-click the asset you want to rename and select **Rename....**
3. Edit the current asset's name and press ENTER.

Adding Assets to a Process from the Solutions View

➤ **To add assets from the Solutions View to a process**

You can drag assets from the Solutions view onto the process editor's canvas as appropriate. For example, if you are creating a process and want to add a user task that is already listed in the Solutions view, you can drag the task onto the editor; Designer creates a user task in the process, with the correct **Task Type ID**.

You can drag a process model onto the process editor to create a call activity step. If the process model contains a none start event, the step is configured with callable process behavior. If the process model does not contain a none start event, the step is configured with referenced process behavior, and its **Start Document** and **Return Document** are both populated. For more information, see [“About Call Activities” on page 253](#).

If you attempt to drag and drop an item that is not applicable for the editor, such as a process project, Designer does not add the item to the process.

Importing and Exporting Assets from the Solutions View

➤ **To import and export assets from the Solutions View**

You can  **Import** and  **Export** files in the Solutions view using the buttons in the view.

Creating Process Projects in the Solutions View

Note:

The very first process project you create in a workspace may not appear in the Solutions view until you refresh the view.

> To create a process project in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window >Show View >  Solutions**
2. Right-click any  **Processes** node, and select  **New Process Project**.
 - If you select the  **Processes** node in the  **Default Solution**, Designer adds the new process project to your workspace, but does not associate it with any solution that you created. After it is created, Designer displays the new process project in the  **Default Solution** in the Solutions view.
 - If you select the  **Processes** node under one of the solutions you created, Designer adds the new process project to your workspace and associates it with this solution. After it is created, Designer displays the new process project in this solution in the Solutions view.
3. In the New Process Project window in the **Project name** field, type a name for the process project.

The project name cannot have any of the following characters:

* | \ : " < > . / ?

4. If you want to create the process project in a different workspace from the one Eclipse is currently using:
 - a. Clear the **Use default location** check box.
 - b. Click **Browse** and identify the location where you want to create the new process project.
5. Click **Finish**.

Note:

You can also create a process project when you create a process. See [“Creating a Process” on page 56](#).

Working with Process Projects in the Solutions View

You can perform actions for process projects in the Solutions view.

> To work with process projects in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window >Show View >  Solutions**
2. The table below lists the actions you can perform for process projects in the Solutions view:

To	Take this action in the Solutions view
View the file in your workspace that corresponds to a process project	<p>Right-click a  process project and select  Show Files.</p> <p>Designer opens the  Navigator view and highlights the file associated with the process project.</p>
Associate a process project with a solution that you created	<p>Right-click a  process project and select  Associate With Solution, then select the name of the solution with which to associate the selected process project.</p> <p>Designer places a check mark next to the solution's name in the menu to indicate the process project is now associated with that solution.</p>
Remove the association of a process project with a solution	<p>Right-click a  process project and select  Associate With Solution, then select the name of the solution from which to remove the association.</p> <p>Designer removes the check mark next to the solution's name in the menu to indicate the process project is no longer associated with that solution.</p>
Add a new process to the process project	Right-click a  process project and select  New Process . Complete the New Process window.
Refresh the contents of the process project	Right-click a  process project and select  Refresh .

Creating Processes in the Solutions View

> To create a new process in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
2. Right-click any  process project node, and select  **New Process**.
3. In the New Process window, type the **Process Name**.

The process name cannot have any of the following characters:

* | : " < > . ?

Note:

Process names can contain spaces, but if they contain spaces at the end of the name, Eclipse will warn you that the name is invalid. Although Designer does not enforce this restriction, it is a good practice to avoid trailing spaces in process names.

4. Select the **Process Project** from the list

OR

Click **New** to create a new process project for your new process.

Note:

If you create a new process project, Designer opens the New Process Project window. Type a **Project name** and select the workspace in which to save it. For more about creating process projects, see [“Creating Process Projects” on page 48](#).

5. Click **Finish**.

Working with Processes in the Solutions View

You can perform actions for processes in the Solutions View.

➤ To work with processes in the Solutions view

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
2. The table below lists the actions you can perform for processes in the Solutions view:

To	Take this action in the Solutions view
Open a process in the process editor	Right-click a  process and select Open , or double-click the process.
View the file in your workspace that corresponds to a process	Right-click a  process and select  Show Files . Designer opens the  Navigator view and highlights the file associated with the process.
View dependencies of a process	Right-click a  process. Select Show Dependencies >  In Library or Show Dependencies >  In Workspace . Because Designer uses metadata to determine dependent assets, if you have set your Designer preferences to disable workspace indexing, the Search view might not list all dependencies.
View references of a process	Right-click a  process. Select Show References >  In Library or Show References >  In Workspace .
	Note: Because Designer uses metadata to determine dependent assets, if you have set your Designer preferences to disable workspace indexing, the Search view might not list all dependencies.

To	Take this action in the Solutions view
Refresh the contents of a process	Right-click a  process and select  Refresh .
Call a another process from a step in the current open process in the process editor	Select a  process and drag it onto the canvas. This creates a call activity step with the same name as the process dropped there. If the process model contains a none start event, the step is configured with callable process behavior. If the process model does not contain a none start event, the step is configured with referenced process behavior. For more information, see “About Call Activities” on page 253 .
Publish process metadata to CentraSite	Right-click a  process and select Publish .
Delete a process	Right-click a  process and select  Delete . CTRL+click to select multiple processes, then right-click and select  Delete .

About Team Development and Version Control

Eclipse supports team development through the use of a version control system (VCS). These are sometimes also referred to as source code control systems. Some examples are Concurrent Versions System (CVS), Perforce, and CentraSite.

You can do the following:

- Obtain assets from a VCS and put them into your workspace from the Navigator view.
- Copy assets from your workspace into a VCS. You also do this from the Navigator view; however, from the Solutions view, you can easily locate the files associated with assets you want to copy.

For information about how to interact with a VCS using the Navigator view, see documentation provided with your VCS that describes the Eclipse plug-in to your VCS.

Important:

There is a `.config` file associated with the `.process` file. It is recommended that you manage the process using the Solutions view, which automatically handles these two files together. See [“About the Solutions View” on page 19](#).

Locating the Files Associated with Assets for Version Control

➤ To locate the files associated with the assets that you want to copy to your VCS

1. In Designer, if the Solutions view is not visible, open it: **Window > Show View >  Solutions**
2. Right-click the asset you want to copy to your VCS, and select **Show Files**.

Designer opens the Navigator view and highlights the file(s) associated with the asset you selected in the Solutions view. This indicates the files you should add to your VCS for the selected asset.

About the Rules Explorer View

You can use the Rules Explorer view to locate, manage, and edit your webMethods Business Rules and their components. The Rules Explorer view is displayed by default in the Rules Development perspective, and you can open it in any perspective by clicking **Window > Show View > Other Software AG >  Rules Explorer**. For more information about working with rules, see *webMethods BPM Rules Development Help*.

About the Package Navigator View

The Package Navigator view is shown by default in the Process Development perspective (in both Business Analyst and Process Developer modes). You use the Package Navigator view to access assets on one or more Integration Servers. From here, you can drag documents and services onto the process editor for use in your process. You can drop documents and services onto existing steps, or directly onto the process editor's canvas.

Note:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

When you select an asset in the Package Navigator view, Designer displays information about it in the Properties view.

When you drag a document from the Package Navigator onto the process editor, Designer automatically creates a receive step in the process. If you have enabled the preference to **Automatically add inputs / outputs to steps created using drag and drop (Window > Preferences > Software AG > Process Development > Appearance)**, Designer sets the output of the receive step as well.

If you drag a second document from the Package Navigator onto an existing receive step that already has an assigned output, Designer retains the previously set output and adds the second output from the second document.

If you drag an asynchronous publish or asynchronous send adapter notification document type from the Package Navigator onto the process editor, Designer automatically creates a receive step in the process.

You cannot drag a synchronous publish and wait adapter notification document type from the Package Navigator directly onto the process editor. To create a synchronous request, create a receive step and then drag and drop the synchronous publish and wait request document onto the step. Similarly, to create a synchronous reply, create a reply step and then drag and drop the synchronous publish and wait reply document onto the step. A synchronous publish and wait adapter notification requires both the request and the reply.

If you close the Package Navigator view and want to reopen it, click **Window > Show View >  Package Navigator**. For more information about the Package Navigator view, see *webMethods Service Development Help*.

Displaying Generated Flow Services in the Package Navigator View

By default, Designer does not display generated flow services in the Package Navigator view. You can view them by editing the Package Navigator filter.

➤ To display generated flow services in the Package Navigator view

- Click  **Filter contents of Navigator**.
- Clear the **Hide generated flow services** check box in the Choose Elements to Display window.

You can also set a preference for this behavior in **Window > Preferences > Software AG > Service Development > Package Navigator**. See "Mapping Data in Flow Services", "Service Properties", and "Package Navigator Preferences" in *webMethods Service Development Help*.

About the Build Report View

The Build Report view opens when you build a process, and shows the progress and details of the build process, including errors and warnings. Click a line in the Build Report view to select the corresponding step in the process editor. You can also copy selected lines or all the text from the Build Report view, and clear the contents from the view.

If you set the Build and Upload Preferences to display stack traces, they are included in the Build Report view as well, up to the maximum number of lines specified.

If you close the Build Report view and want to reopen it, go to **Window > Show View > Other >  Build Report**.

About the Problems View

The Problems view is a default Eclipse view that displays system-generated errors, warnings, or information associated with a resource. For example, if a receive step in your process does not have a subscription document specified, the error is automatically logged in this view. The Problems view also displays errors and warnings generated by Simulation (in the Process Simulation perspective).

The Problems view opens by default in the Process Development perspective. If you close the Problems view and want to reopen it, or if you want to open it in another perspective, go to **Window > Show View >  Problems**.

About the Error Log View

The Error Log View provides a tabular view of the *<workspace directory>/metadata/.log* file. It also provides functionality to import, export, delete, and restore the log file. You can also clear the viewer without removing the data.

The Error Log view is available in all perspectives from **Window > Show View > Other > General >  Error Log**.

3 Process Development Preferences

- About Process Development Preferences 34
- Configuring Process Development Preferences 34
- About Capabilities Preferences 35
- Configuring the Process Development Mode 35
- Configuring Appearance Preferences 36
- Configuring Colors And Fonts Preferences 39
- Configuring Build and Upload Preferences 40
- Configuring Optimize Server Preferences 42
- Configuring Process Audit Database Preferences 43
- Creating a Custom Image Set for Activity Steps 45

About Process Development Preferences

Most Designer preferences are located in **Window > Preferences > Software AG**:

- Preferences specific to Process Development are located under **Process Development**.
- Capabilities preferences are located in **Window > Preferences > General > Capabilities**.
- Search preferences are located in **Window > Preferences > General > Search**.

When you first open the Process Development perspective, the Business Analyst mode is enabled by default. For more information, see [“About Process Development Modes” on page 52](#). In Business Analyst mode, only basic preferences appear in the Preferences window. To see and work with advanced preferences, you must select the Process Developer mode, as described in [“Configuring the Process Development Mode” on page 35](#).

Configuring Process Development Preferences

➤ To configure Process Development preferences

1. In Designer: **Window > Preferences**
2. Click to expand the **General** entry and set preferences for each of the following sections.
 - Capabilities Preferences; for more information, see [“About Capabilities Preferences” on page 35](#).
 - Search Preferences; for more information, see:
 - [“Search Preferences” in *webMethods BPM and CAF Workspace Metadata Help*](#)
 - [“Configuring Search Preferences” on page 486](#)
3. Click to expand the **Software AG** entry and set preferences for each of the following sections:
 - [“Connecting to CentraSite” in CentraSite Help](#).
 - [“Integration Server Preferences” in *webMethods Service Development Help*](#)
 - [“Workspace Index Preferences” in *webMethods BPM and CAF Workspace Metadata Help*](#)
4. Click to expand the **Software AG > Process Development** entry and set preferences for each of the following sections.
 - [“Configuring Appearance Preferences” on page 36](#)
 - [“Configuring Colors And Fonts Preferences” on page 39](#)
 - [“Configuring ARIS Server Preferences” on page 397](#)
 - [“Configuring Build and Upload Preferences” on page 40](#)

- [“Configuring Default Documentation Fields Preferences” on page 382](#)
 - [“Configuring Optimize Server Preferences” on page 42](#)
 - [“Configuring Process Audit Database Preferences” on page 43](#)
 - [“Configuring Process Debug Preferences” on page 458](#)
5. Do one of the following:
- Click **Apply** to save your changes and keep the Preferences window open.
 - Click **OK** to save your changes and close the Preferences window.
 - Click **Restore Defaults to apply default preferences on a page**.
 - Click **Cancel** to close the Preferences window without saving your changes.

Tip:

You can use the  **Forward** and  **Back** buttons in the Preferences window to navigate through Preferences pages you have visited.

For information on process simulation preferences, see “Process Simulation Preferences” in the *webMethods BPM Process Simulation Help*.

About Capabilities Preferences

You can configure Eclipse capabilities preferences to add or remove functionality. For example, you can disable the Eclipse Plug-in Development capability if you choose. For more information about Eclipse capabilities, see the Eclipse documentation installed as online help in Designer.

Software AG Designer provides two different modes of operation: *Business Analyst mode* and *Process Developer mode*. For more information about these modes, see [“About Process Development Modes” on page 52](#).

Typically, you select the mode you want to work with by clicking the tool bar buttons dedicated to these modes. However, you can also set the mode as a preference, as described in [“Configuring the Process Development Mode” on page 35](#).

Configuring the Process Development Mode

You can select the Process Development mode with the Business Analyst and Process Developer buttons on the Designer tool bar, or by changing the Process Developer capability preference.

➤ To enable or disable the Process Developer Capability

1. In Designer, do either of the following:
 - On the Designer toolbar, click  **Process Developer** if you want to enable advanced options, or click  **Business Analyst** if you want only basic options.

- In Designer:
 1. Go to **Window > Preferences > General > Capabilities**.
 2. Click **Advanced**.
 3. Expand the **Process Developer** folder.
 4. Clear the **Advanced Process Development** check box to enable Business Analyst mode, or select the check box to enable Process Developer mode.
 5. Click **OK**, and then click **OK** again.

The **Process Developer** check box at **Window > Preferences > General > Capabilities** always displays your current mode as selected on the tool bar.

Configuring Appearance Preferences

You can configure the default behavior of many process aspects on the Appearance page in the Software AG Preferences window. **Window > Preferences > Software AG > Process Development > Appearance**. You can:

- Set the process editor to show or hide inputs/outputs and tooltips by default.
- Set the way activities and events are configured when you create them by drag and drop actions, and whether inputs and outputs update automatically.
- Set auto-layout and pool orientations, transition line shape and text type, and whether you want Designer to automatically switch perspectives for you when you open different types of processes.

➤ To configure Process Development Appearance preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Appearance**
2. Configure the following Process Development Appearance preferences on the Appearance page as described in the table below:

Preference	Description
Show inputs and outputs by default	Select the check box to enable. Disabled by default.
Show Tool Tips	Clear the check box to disable. Enabled by default.
Automatically add inputs / outputs	Clear the check box to disable. Enabled by default.
Replace existing inputs/outputs with	Select the check box to enable. Disabled by default. Available only when Automatically add inputs/outputs is enabled.

Preference	Description
automatically added inputs/outputs	
Automatically update step names when adding documents/services via drag-and-drop	<p>Clear the check box to disable. Enabled by default. When enabled, this preference adds the document or service name to the step label after a document or service is dropped onto the step.</p> <ul style="list-style-type: none"> ■ When IS document types are being dragged and dropped, this preference is subject to the Show event and gateway labels by default preference. When the Show event and gateway labels by default preference is disabled, step names will not be updated when a document is dropped onto the step. ■ Service names always follow the behavior defined by this preference, and are not subject to the Show event and gateway labels by default preference.
Show event and gateway labels by default	<p>Clear the check box to disable. Enabled by default. When this preference is disabled, event and gateway steps are created without a label. This preference also affects the Automatically update step names when adding documents/services via drag-and-drop preference, see that preference description for more information.</p>
Automatically update task business data with the task inputs/outputs	<p>Click Always, Never, or Prompt. Default is Always.</p>
Default step label location	<p>Click Below step to place task, call activity, and collapsed subprocess labels underneath the step. Click On step to place the label inside a task, call activity, and collapsed subprocess step. Default is Below step. Does not apply to event or gateway steps. A toolbar button is also available, see “Changing Task Label Placement” on page 217.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: This setting applies only to label placement for process models created <i>after</i> you apply the setting. If you open or import an existing model, the labels in that model will retain their position, and you must change the label placement with the toolbar button, as described in “Changing Task Label Placement” on page 217.</p> </div>
Default pool orientation	<p>Click Horizontal or Vertical. Default is Horizontal.</p>
Default auto layout orientation	<p>Click Horizontal or Vertical. Default is Horizontal.</p>

Preference	Description
Default transition line shape	Click Curved , Straight , or Custom . Default is Straight .
Transition line text	<p>Click to specify the display priority of the information you want Designer to show on the transition line. Click Condition Expression, Transition Description, or None. Default is Condition Expression. For example:</p> <ul style="list-style-type: none"> ■ If Condition Expression is selected, and a line has both a condition defined and a transition description, the condition expression is displayed on the line. ■ If Condition Expression is selected and a line has no condition defined but does have a transition description, the transition description is displayed on the line. ■ If Transition Description is selected, and a line has both a condition defined and a transition description, the transition description text is displayed on the line. ■ If Transition Description is selected, and a line has a condition defined but no transition description, the condition expression is displayed on the line. <p>If you click None, no text is displayed on the transition line.</p>
Custom task images	Click the Browse button to select the directory containing your custom images for use as activity step ornaments. Default is <code>..\Designer\resources\.</code> For more information, see “ Creating a Custom Image Set for Activity Steps ” on page 45.
Switch to the Process Development perspective when opening a process file	Click Always , Never , or Prompt . Default is Prompt .

3. Do one of the following:

- Click **Apply** to save your changes and keep the Preferences window open.
- Click **OK** to save your changes and close the Preferences window.
- Click **Restore Defaults to apply default preferences on a page**.
- Click **Cancel** to close the Preferences window without saving your changes.

Configuring Colors And Fonts Preferences

On the Colors and Fonts page in the Software AG Preferences window (**Window > Preferences > Software AG > Process Development > Appearance > Colors and Fonts**), you can set the default colors for swimlanes, swimlane labels, pool labels, subprocesses, and notes. You can also set the default font styles, including font name, size, weight, angle, and color, for process step labels, transition labels, notes, and annotations.

➤ To configure Process Development Colors And Fonts preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Appearance > Colors and Fonts**.
2. Configure the following Process Development Colors and Fonts preferences on the Colors And Fonts page as described in the table below:

Preference	Description
Default colors	Click a color button to specify a color for each of the following Swimlanes, Swimlane Labels, Pool Labels, Subprocesses, and Notes . No color association can be set for annotations.
Default step label font style	<p>Select a font from the list. Default is Arial.</p> <p>Select a font size from the list. Default is 8.</p> <p>Click the  Bold button to apply bold formatting to your default font.</p> <p>Click the  Italic button to apply <i>italic</i> formatting to your default font.</p> <p>Click the Font Color button to choose the default font color from a palette. Default is black.</p>
Default transition label font style	<p>Select a font from the list. Default is Arial.</p> <p>Select a font size from the list. Default is 8.</p> <p>Click the  Bold button to apply bold formatting to your default font.</p> <p>Click the  Italic button to apply <i>italic</i> formatting to your default font.</p> <p>Click the Font Color button to apply a specific font color from a palette. Default is black.</p>

Preference	Description
Default annotation/note font style	Select a font from the list. Default is Arial.
	Select a font size from the list. Default is 8.
	Click the  Bold button to apply bold formatting to your default font.
	Click the  Italic button to apply <i>italic</i> formatting to your default font.
	Click the Font Color button to apply a specific font color from a palette. Default is black.

3. Do one of the following:

- Click **Apply** to save your changes and keep the Preferences window open.
- Click **OK** to save your changes and close the Preferences window.
- Click **Restore Defaults to apply default preferences on a page**.
- Click **Cancel** to close the Preferences window without saving your changes.

Configuring Build and Upload Preferences

You can configure the behavior applied when you build and upload a process model to Integration Server.

➤ To configure Build and Upload preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Build and Upload**.
2. You can configure the Build and Upload preferences as described in the following table:

Preference	Description
Save before building/uploading process	Click Always , Never , or Prompt . Default is Prompt . Note: By default, when you add a step to a process model and then click Edit Data Mapping , the step in the model is saved if Save before building/uploading process is enabled. If it is not enabled, Designer prompts you to save the step.

Preference	Description
Automatically build/upload referenced process with parent process	Click Always , Never , or Prompt . Default is Always .
Automatically enable process for execution after build/upload	Click Always , Never , or Prompt . Default is Prompt .
Abort build/upload when trigger threads are actively processing documents	Click Always or Never . Default is Never .
Automatically deploy task to Task Engine	Click Always , Never , or Prompt . Default is Prompt .
Automatically upload KPIs on build	Click Always , Never , or Prompt . Default is Prompt .
Prompt to upload KPIs	For  Upload KPIs menu and toolbar button actions. Does not apply to uploading KPIs with a build. Click Always or Never . Default is Always .
Display stack traces in Build Report when encountering exceptions	Select the check box to enable. Enabled by default.
Stack Trace Line Limit	Maximum number of stack trace lines included in the Build Report view when encountering exceptions. This value is used only when Display stack traces in Build Report when encountering exceptions is enabled. The default value is 10.

3. Do one of the following:

- Click **Apply** to save your changes and keep the Preferences window open.
- Click **OK** to save your changes and close the Preferences window.
- Click **Restore Defaults to apply default preferences on a page**.

- Click **Cancel** to close the Preferences window without saving your changes.

Configuring Optimize Server Preferences

To upload KPIs, you must specify an Optimize Server.

In addition to Broker (deprecated) messaging, Designer supports Software AG Universal Messaging, which means you can use any JMS provider. The host and port information you specify in Process Development Preferences is used to create a URL.

- When you specify a **Universal Messaging** server configuration, the resulting URL starts with “nsp://”, and the default value is `nsp://localhost:9000`.
- When you specify a **Broker** (deprecated) server configuration, the resulting URL starts with “broker://”, and the default value is `broker://localhost:6849/Broker #1`.

You can change the Optimize Server configuration at any time.

> To configure Optimize Server preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Optimize Server**
2. The Optimize Server connection uses JMS messaging. The default selection is **Universal Messaging (UM)**. Click the **Broker (deprecated)** option to use it instead.
3. Configure the following Optimize Server preferences for your selected connection. Appropriate fields appear depending on whether you select the Broker (deprecated) or Universal Messaging option.

The table below describes the Optimize Server preferences.

Preference	Description
UM Host	Universal Messaging host name. Default value is <code>localhost</code> .
UM Port	Universal Messaging host port. Default value is <code>9000</code> .
Broker Host (deprecated)	Broker (deprecated) host name. Default value is <code>localhost</code> .
Broker Port (deprecated)	Broker (deprecated) host port. Default value is <code>6849</code> .
Broker Name (deprecated)	Broker (deprecated) name. Default value is <code>Broker #1</code> .
Timeout (seconds)	Enter the number of seconds Designer should wait for a connection to the Optimize Server. A value of -1 means never time out. The default value is <code>60</code> .
Test Connection	Click to test the connection to the defined Optimize Server.

4. Do one of the following:
 - Click **Apply** to save your changes and keep the Preferences window open.
 - Click **OK** to save your changes and close the Preferences window.
 - Click **Restore Defaults to apply default preferences on a page**.
 - Click **Cancel** to close the Preferences window without saving your changes.

Configuring Process Audit Database Preferences

Note:

When you use NTLM to connect to the Process Audit database, the connection attempt may fail with the error "NTLM (type-2) Authentication was requested but the required DDJDBCAuth04.dll was not found". By default, the Eclipse framework does not include the Type 2 drivers in the path that NTLM connections require. To correct this, you can copy the required DLL from the common/bin folder to the folder containing the Eclipse executable (such as <Software AG installation path>/eclipse/v36). Another way to correct this is to add the path to the eclipse.ini file. For example: -Djava.library.path=C:\SoftwareAG\common\bin

➤ To configure Process Audit Database preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Process Audit Database**
2. You can configure the Process Audit Database preferences as described in the following table:

Preference	Description
Database Connectivity	Click one of the following: <ul style="list-style-type: none"> ■ Use database parameters ■ Use Integration Server JDBC pool parameters (default)
RDBMS	For use with the database parameters selection in the Database Connectivity preference. Choose the type of RDBMS connection from the list: Oracle , SQL Server , DB2 , or Sybase .
URL	For use with the database parameters selection in the Database Connectivity preference. Enter URL information for database connection. Sample URL formats are displayed above the RDBMS field.
Database user	For use with the database parameters selection in the Database Connectivity preference. Enter the database user name.

Preference	Description
Password	For use with the database parameters selection in the Database Connectivity preference. Enter the database password.
Select a running Integration Server that has the Process Audit pool configured	For use with the Integration Server JDBC pool parameters selection in the Database Connectivity preference. Choose an Integration Server from the list. To add a server, open the Integration Server preferences page. Tip: When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.
Test Connection	Click the button to test the connection to the server using the authentication credentials and parameters supplied.

3. Do one of the following:

- Click **Apply** to save your changes and keep the Preferences window open.
- Click **OK** to save your changes and close the Preferences window.
- Click **Restore Defaults to apply default preferences on a page.**
- Click **Cancel** to close the Preferences window without saving your changes.

Connecting to Process Audit Database Using Kerberos Authentication

To connect directly to a Process Audit Database that uses an SQLServer with DataDirect drivers, you use Kerberos authentication. To use Kerberos authentication you must first set-up the operating system to use Kerberos authentication. For more information on required Kerberos setup, see the Kerberos documentation.

> To configure Designer to use authenticate with a SQLServer using Kerberos

1. Locate and copy the paths to the two configuration files required for Kerberos authentication:

- JDBCdriverLogin.conf
- krb5.conf

For example, on a Windows system the paths would look like this:

C:\SoftwareAG\common\lib\ext\JDBCdriverLogin.conf

2. Navigate to *Software AG_directory* \Designer\eclipse\configuration and open the config.ini file.
3. In the config.ini file add the paths to the JDBCLogin.conf and krb5.conf files using the following syntax:
 - `java.security.auth.login.config=Software AG_directory\common\lib\ext\JDBCLogin.conf`
 - `java.security.krb5.conf=Software AG_directory\common\lib\ext\krb5.conf`
4. Save your changes to the config.ini file.
5. In Designer, go to Window>Preferences>Software AG>Process Development>Process Audit Database and select **Use database parameters**.
6. Click **OK**.
7. Add the Kerberos authentication mode to the JDBC URL, for example, `jdbc:wm:DBSERVER:1433;databaseName=dbname;Authenticate=kerberos`

Note:

Do not enter userName and password, because the authentication will be based on the Windows user credentials.

Creating a Custom Image Set for Activity Steps

When you specify an ornament for an activity step as described in [“Assigning Custom Step Ornaments” on page 77](#), you can choose from a set of system images available from within Designer. You can also create one or more jar files containing a custom image set and add them to your Designer installation to make them available as ornaments.

➤ To create a custom image set

1. Create or locate the image files you want to include in your custom image set.

You can use images in GIF, SVG, JPG, JPEG, and PNG formats. Designer resizes images to 21 x 21 pixels, if they are not already this size. For best image results, use images that conform to the 21 x 21 pixel size.

Note:

The following two steps instruct you to create a jar file containing your custom image files. This is to enable you to organize your image files by tab when they appear in the Custom Image selection dialog box. If you have no need to organize your custom files, you can place them all in a single custom image directory and omit the jar file creation. In this case, you can proceed directly to step 4. The directory name is used as the tab name.

2. Compress the image files into a jar file. Ensure that all image files are at the root level of the jar file. If you want to organize the image files, create multiple jar files, each with an individual

set of image files. Each jar file appears as a separate tab in the Custom Image selection dialog box. The name you apply to the jar file is displayed as the tab name. For example, if you create a jar file named “custom_images.jar”, the tab name is also “custom_images.jar.”

3. Place the jar file in the custom image directory (that is, the directory you want to use for custom image files). This can be any directory in your file system.
4. In Designer: **Window > Preferences > Software AG > Process Development > Appearance**
5. In the Custom task images area, click the **Browse** button and select your custom image directory.
6. Click **OK**.

4 Process Projects

■ About Process Projects	48
--------------------------------	----

About Process Projects

Process projects serve as the parent structures for Designer processes and their assets. A single process project can contain one or more processes. You can create a process project without a process in it, but a process must belong to a process project, and cannot exist outside of it.

Creating Process Projects

Note:

The very first process project you create in a workspace may not appear in the Solutions view until you refresh the view.

➤ **To create a process project**

1. In Designer: **File >New >**  **Process Project**.
2. In the New Process Project window, type the project name.
3. If you want to create the process project in a different workspace from the one Eclipse is currently using, do the following:
 - a. Clear the **Use default location** check box.
 - b. Click **Browse** and identify the location where you want to create the new process project.
4. Click **Finish**.

Note:

The new process project is automatically associated with the default solution. You can also create new process projects in the Solutions view. This allows you to specify a solution with which you want the new process project to be associated. See [“Creating Process Projects in the Solutions View” on page 25](#).

Note:

You can also create a process project when you create a process. See [“Creating a Process” on page 56](#).

Deleting Process Projects

➤ **To delete a process project**

1. If the Navigator view is not showing, click **Window >Show View >**  **Navigator**.
2. Right-click the process project you want to delete, and select **Delete**.

3. In the Confirm Project Delete window, choose whether to delete the rest of the contents of the process project directory or not. The default selection is **Do not delete contents**.
4. Click **Yes** to confirm your selection and delete the process project.

5 Processes

■ About Processes	52
■ About Process Development Modes	52
■ Legacy Processes and BPMN	53
■ Creating a Process	56
■ Configuring a Process	57
■ Updating Process Model Versions	57
■ Troubleshooting a Process	58
■ Basic Process Properties	58
■ Advanced Process Properties	61
■ About Synchronizing Process Runtime Settings with webMethods Monitor	66
■ Deleting a Process	67
■ Printing a Process	67
■ Saving a Process Image	68
■ Copying a Process	68
■ Using E-forms in a Process	69

About Processes

A process is the top-level asset in a process project. It contains steps and logic; and it is the asset that is ultimately built and executed in the Process Engine and uploaded to the Process Audit Database for analysis. Processes can be started in many different ways, including using rules and services.

A process can invoke a rule, of course, but a rule can also invoke a process. Designer 8.2 and later contains new functionality to support BPMN 2.0, including new ways to represent your processes in a more visual and logical language. Activities are things that are done. Events are things that happen. Those are the first concepts you need to understand. See [“About Activities” on page 212](#) and [“About BPMN Events” on page 264](#).

Important:

There is a `.config` file associated with the `.process` file. It is recommended that you manage the process using the Solutions view, which automatically handles these two files together. See [“About the Solutions View” on page 19](#).

BPMN processes can be designated as *callable*. A BPMN *callable process* has a defined input / output signature, known as a *Global Process Specification*, and can be called by a call activity step. Call activities can also call webMethods referenced processes. For more information, see [“Call Activity Concepts” on page 147](#).

You can open older Designer processes created with Designer 8.1 and earlier with later versions of Designer. Some changes are applied to ensure compatibility with BPMN. For a reference of those changes, see [“Legacy Processes and BPMN” on page 53](#).

About Process Development Modes

The Process Development perspective supports two modes of operation, the *Business Analyst* mode, which provides a limited amount of information and capability, and the *Process Developer* mode, which provides full information and capability.

When you first open the Process Development perspective, the Business Analyst mode is enabled by default. In Business Analyst mode, the Process Development perspective displays only basic properties in the Properties view, basic preferences in the Preferences window, and basic functions on the tool bar. To see advanced properties, preferences, and functions, you must select the Process Developer mode, as described in [“Configuring the Process Development Mode” on page 35](#).

The Business Analyst mode provides a streamlined set of features and functions that enable the business analyst to create business process models without the distraction of the more advanced process development functionality. If you are a process developer, you will most likely want to work exclusively in Process Developer mode. You can move back and forth between the two modes at any time and as often as you want.

Throughout this documentation, the properties that appear in the Properties view in the Process Developer mode are referred to as *advanced properties*, and the properties displayed in the Business Analyst mode are referred to as *basic properties*.

Legacy Processes and BPMN

With the introduction of Software AG Designer 8.2, the Process Development perspective began a transition to BPMN 2.0 notation and behavior. If you have business process models that were created with an earlier version, you can import those processes (referred to as *legacy processes*) into Designer 8.2 and later. However, because of the move to BPMN 2.0, some conversion is applied to steps imported from legacy Designer processes into BPMN Designer, as described in the following table:

Legacy Designer Step	Converted to BPMN Step in Designer
Empty step	Abstract task “About Abstract Tasks” on page 221
Error transition from a subprocess	Boundary intermediate error event (non-interrupting) “About Boundary Intermediate Error Events” on page 291
Error transition from any step other than a subprocess	Boundary intermediate error event (interrupting) “About Boundary Intermediate Error Events” on page 291
Gateway step	Complex gateway “ About Gateways” on page 340
IS service step	Service task with type set to IS Service “About Service Tasks” on page 222
Join timeout transition (with timeout set) at a legacy join step	Join timeout transition This is true only for a timeout transition defined at a legacy join step. All other timeout transitions are converted into Boundary Timer events. “About Joins” on page 131
Publish step	Send task “About Send Tasks” on page 233
Receive step - starting receive	Message start event Note: Available Protocols: Publishable Subscription, JMS “About Message Start Events” on page 269

Legacy Designer Step	Converted to BPMN Step in Designer
Receive step - intermediate receive (that is, it does not start a process)	<p data-bbox="747 247 1385 289">“About Receive Steps” on page 438</p> <p data-bbox="747 310 1385 352">Receive task</p> <div data-bbox="747 367 1385 493" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="747 367 1385 409">Note: Available protocols: Publishable Subscription, JMS, Simple Service</p> </div>
Receive step and reply step (Simple Service for synchronous reply protocol on the receive step)	<p data-bbox="747 508 1385 550">“About Receive Tasks” on page 237</p> <p data-bbox="747 567 1385 609">“About Receive Steps” on page 438</p> <p data-bbox="747 625 1385 667">Receive task and send task</p> <div data-bbox="747 682 1385 787" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="747 682 1385 724">Note: Simple Service Protocol</p> </div>
Referenced process step	<p data-bbox="747 793 1385 835">“About Send Tasks” on page 233</p> <p data-bbox="747 850 1385 892">“About Receive Tasks” on page 237</p> <p data-bbox="747 907 1385 949">“About Receive Steps” on page 438</p> <p data-bbox="747 966 1385 1039">Call activity with type webMethods process (deprecated)</p> <p data-bbox="747 1060 1385 1134">“About Call Activities” on page 253 and “About BPMN Callable Processes” on page 148</p>
Reply step	<p data-bbox="747 1155 1385 1197">Send task</p> <p data-bbox="747 1213 1385 1255">“About Send Tasks” on page 233</p>
Rule step	<p data-bbox="747 1270 1385 1344">Business rule task with type set to webMethods Business Rule (default)</p> <p data-bbox="747 1360 1385 1407">“About Rule Tasks” on page 230</p>
Subprocess step	See “About Subprocess Types” on page 146 .
Task step	<p data-bbox="747 1480 1385 1522">User task</p> <p data-bbox="747 1539 1385 1581">“About User Tasks” on page 225</p>
Terminate step	<p data-bbox="747 1600 1385 1642">End terminate event</p> <p data-bbox="747 1659 1385 1701">“About Terminate End Events” on page 307</p>
Timeout transition (with timeout set)	<p data-bbox="747 1717 1385 1759">Boundary intermediate timer event</p> <p data-bbox="747 1776 1385 1885">The interrupting behavior of boundary timer events is described in “About Interrupting Behavior for Boundary Intermediate Events” on</p>

Legacy Designer Step	Converted to BPMN Step in Designer
	<p>page 288. A boundary intermediate timer event outgoing transition label shows Execution time > xxx .</p> <p>For more information, see “About Boundary Intermediate Timer Events” on page 291.</p>
Transition of type Step Iterations Exceeded	<p>Labeled conditional transition with no decorator.</p> <p>Label shows Iterations > xxx .</p>
Transition of type Unsatisfied Join	<p>Labeled conditional transition with no decorator.</p> <p>Label shows Unsatisfied Join.</p>
Web service step	<p>Service task with type set to Web Service</p> <p>“About Service Tasks” on page 222</p>

About BPMN Process Steps

With the introduction of Software AG Designer 8.2 and later versions, the Process Development perspective includes a number of step types that support the BPMN model.

The following table describes the BPMN Designer steps.

BPMN Designer Step	Description
None Event (Start, Intermediate Throw, End)	<p>See:</p> <ul style="list-style-type: none"> ■ “About None Start Events” on page 268 ■ “About None Intermediate Events” on page 276 ■ “About None End Events” on page 299
Signal Event (Start, Intermediate -- Throw or Catch, Boundary Interrupting and Boundary Non-Interrupting, End)	<p>See:</p> <ul style="list-style-type: none"> ■ “About Signal Start Events” on page 272 ■ “About Signal Intermediate Events” on page 283 ■ “About Signal End Events” on page 305
Error Event (Intermediate Boundary Interrupting and Non-Interrupting, End)	<p>See:</p> <ul style="list-style-type: none"> ■ “About Boundary Intermediate Error Events” on page 291

BPMN Designer Step	Description
Message Event (Intermediate, Boundary Interrupting and Non-Interrupting, End)	<ul style="list-style-type: none"> ■ “About Error End Events” on page 300 <p>See:</p> <ul style="list-style-type: none"> ■ “About Message Intermediate Events” on page 278 ■ “About Boundary Message Intermediate Events” on page 291 ■ “About Message End Events” on page 302 <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Message Intermediate Event has input and output transitions. This is the distinction between it and our former notion of Intermediate Receive. Intermediate Receives with no input transition migrate to Receive Tasks.</p> </div>
Manual Task	“About Manual Tasks” on page 228
Gateway -- Inclusive, Exclusive, Parallel	“ About Gateways” on page 340
Subprocess	See “About Subprocess Types” on page 146

Creating a Process

➤ To create a process

1. In Designer: **File > New >  Process.**
2. In the New Process window, type the **Process Name**.
3. Select the **Process Project** from the list OR click **New** to create a new process project for your new process.
 - If you create a new process project, Designer displays the New Process Project window. Type a **Project name** and select the workspace in which to save it. For more about creating process projects, see [“Creating Process Projects” on page 48](#).

Note:

Process names can contain spaces, but if they contain spaces at the end of the name, Eclipse will warn you that the name is invalid. Although Designer does not enforce this restriction, it is a good practice to avoid trailing spaces in process names.

Important:

Do not name your process using the prefix `wm`, whether in lowercase, uppercase, or a combination thereof. Integration Server names its system packages using those letters as the prefix, and using the same prefix in process names could cause problems later.

4. Click **Finish**.

Tip:

You can also create new processes in the Solutions view. See [“Creating Processes in the Solutions View” on page 27](#).

Configuring a Process

Each process has a number of basic and advanced properties that you can configure, in addition to the properties you can set for steps, transition lines, pools, swimlanes, and other process components.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

> To configure a process

1. In the Process Development perspective, locate the process you want to work with in the Solutions view and double click to open it in the process editor.
2. Click the process editor canvas to select the open process.
3. In the Properties view, do one or both of the following:
 - Configure the basic properties for the process, as described in [“Basic Process Properties” on page 58](#).
 - Configure the advanced properties for the process, as described in [“Advanced Process Properties” on page 61](#).
4. Save the process.

Updating Process Model Versions

In Software AG Designer, you can modify an existing process model and create a new version of the model. You can then build and upload the new version, and then enable it for use. In doing so, you can update any or all of the currently running instances of that model so that they start using the newer version. For more information, see these topics:

- [“Working with Process Versions” on page 442](#).
- [“Updating a Process Instance to a New Model Version”](#) in the PDF publication *webMethods Monitor User’s Guide*.

- “Enabling and Disabling Process Model Versions” in the PDF publication *webMethods Monitor User’s Guide*.

Troubleshooting a Process

When you develop a new process model, there may be times when the process does not execute as expected, and you want to find out why. Similarly, an existing process model may suddenly begin to behave in an unexpected manner, and you want to know why. The Process Engine provides you with the ability to gather troubleshooting information in the following ways:

- You can collect a package of comprehensive process information for troubleshooting purposes. For more information, see the chapter “Collecting Process Troubleshooting Information” in the PDF publication *Administering webMethods Process Engine*.
- You can collect logging information, as described in “Process Logging Behavior” on page 169.

You can use this information for your own troubleshooting efforts, and you may be asked to provide either or both of the above to Software AG Global Support if you have opened a support issue.

In addition, you can take advantage of other means of examining process instance runtime results in *webMethods Monitor*. With proper logging settings, you can examine the process instance pipeline, modify the pipeline, and resubmit a process instance. The Process Details page provides a great deal of information about the process instance, including a graphical representation of the process annotated with step statuses. For more information, see the “Process Monitoring” chapter of the PDF publication *webMethods Monitor User’s Guide*.

You can also use the Process Simulation feature in Software AG Designer to test the behavior of your model with data and conditions that the process would normally work with when it is implemented in the run-time environment. For more information, see the *webMethods BPM Process Simulation Help*.

Basic Process Properties

The following table describes the basic process properties.

Properties Page	Property	Description
General	Process Name	The name of the process model. Important: If you change a process-wide property, such as renaming a process or a package, Designer prompts you to regenerate the process.
	Process ID	A system-generated process identifier. Not editable.

Properties Page	Property	Description
		The Process ID consists of the process project name, any intermediate folders, and the process name, each separated by a slash (/), if all characters are simple ASCII characters. If any non-ASCII characters are used, Designer uses an encoding scheme to render all characters in the Process ID in ASCII.
	Version	The current version of the process. The initial process version is 1. See “Working with Process Versions” on page 442.
	Created By	The user name of the creator of the process. Not editable.
	Description	Your descriptive information about the process, for documentation purposes only. Text in process descriptions is searchable.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable. See “About Process Documentation” on page 380.
	Documentation Field Value	Value for the assigned Documentation Field .
KPIs	KPIs for <Process Name>	Step-level KPI (Key Performance Indicator) definitions. For more information about the fields on this page and how to work with them, see “About KPIs” on page 372.
	KPI Properties	Information used to describe a KPI: Name, Description, Unit of Measure, Associated Field, Aggregation Type , and Dimensions . See “About Step-Level KPIs” on page 375.
	Name	Name of the KPI. Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names on the target server. However, if you duplicate a KPI name in a process, generation of the process produces a warning.
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.

Properties Page	Property	Description
	Associated Field	<p>Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used.</p> <p>Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.</p>
	Aggregation Type	Aggregation type can be set to SUM , AVERAGE , or LAST VALUE
	Dimensions	<p>Use this area to add, delete, and manage data dimensions for a selected KPI. You can associate a field from the step output with one or more dimensions.</p> <p>Note: Dimension names must be unique on a given Optimize server. Designer cannot, at design time, detect all the dimension names on the target server. However, if you duplicate a dimension label in a process, generation of the process produces a warning.</p> <p>For more information, see “About KPIs” on page 372.</p>
Stages	Stages	A list of the stages created within this process model. For more information, see “Working with Stages” on page 353 .
	Add Stage	Click to add a stage.
	Delete Stage	Click to delete a selected stage.
	Stage Details	Configuration information about the selected stage.
	Name	Name of the stage (read-only).
	Description	Description (optional) of the stage.

Properties Page	Property	Description
	Start Milestone	Selected start milestone for the stage.
	End Milestone	Selected end milestone for the stage.
	Condition	Specified condition to define a stage breach.
	Stop Tracking On Breach	Stops stage processing for all remaining stages in the process instance when a stage breach occurs in this stage, and only one stage breached EDA (deprecated) event is emitted. Remaining stages are not tracked and will be shown as Incomplete in Monitor. The check box is cleared by default.
ARIS Model (for ARIS processes only)	ARIS Download Client	Open the model in ARIS.
	ARIS Process Documentation	View documentation of the ARIS process.
	ARIS Connect	Open the model in ARIS Connect.

Advanced Process Properties

The following table describes the advanced process properties.

Properties Page	Property	Description
Error	Error Handler Task	<p>The task step that the Process Engine executes when an error occurs during the execution of a process instance. Also known as a process-wide error step.</p> <ul style="list-style-type: none"> ■ If a step with a boundary intermediate error event fails, the transition for that event is taken in response to the failure. ■ If a step with no boundary intermediate error event fails, the process executes the logic defined in the error handler task. ■ If a step with no boundary intermediate error event fails and there is no designated error handler task, the process fails.
Cancel	Cancel Handler Task	The task step that the Process Engine executes, along with any downstream transitions, when

Properties Page	Property	Description
		<p>its status is changed to Cancel by webMethods Monitor or a service invocation. Also known as a process-wide cancel step.</p> <p>Note: Designer supports debugging the Cancel Handler Task. When using the Trace view in the Process Debugging perspective, you can manually force the cancellation of a process. See “About the Trace View” on page 463.</p>
Timeout	Timeout Handler Task	<p>The task step that the Process Engine executes when the Maximum Process Execution Time is reached.</p> <p>Note: Designer supports debugging the Timeout Handler Task. When using the Trace view in the Process Debugging perspective, you can manually force a process-wide timeout. See “About the Trace View” on page 463.</p>
	Maximum Process Execution Time	<p>Defines the maximum length of time a process can execute. The Timeout Handler Task step executes when this setting is reached. If no timeout handler task is defined, the process instance fails. Also known as a process-wide timeout.</p> <p>The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value based on a business calendar in My webMethods Server. You can specify the number of days, hours, and minutes. <p>For more information, see “Defining a Process Timeout” on page 139.</p>
Run Time	Generated Package Name	<p>By default, the name of the generated Integration Server package is set to the name of the project. You can edit the package name. Only ASCII characters are allowed.</p> <p>Important: If you change a process-wide property, such as renaming a process or a package,</p>

Properties Page	Property	Description
	Quality of Service	<p data-bbox="886 264 1414 331">Designer prompts you to regenerate the process.</p> <p data-bbox="886 365 1468 537">This section provides settings that affect process run-time behavior. For complete information about working with these fields, see “Setting Quality of Service for a Process” on page 111.</p> <ul style="list-style-type: none"> <li data-bbox="886 564 1487 701">■ Optimize Locally: Execute adjacent steps on the same Integration Server without publishing transition documents. Enabled by default. <li data-bbox="886 728 1487 898">■ Express Pipeline: Send a reduced data set between the steps in a process. Enabled by default. Can significantly improve performance with large pipelines (1 MB or more). <li data-bbox="886 926 1487 1520">■ Volatile Transition Documents: Send process transition documents in volatile mode. Enabled by default. <ul style="list-style-type: none"> <li data-bbox="935 1079 1487 1318">■ Select Volatile Transition Documents to specify that Universal Messaging or Broker (deprecated) stores process transition documents and referenced process start documents in the specified message provider RAM. <li data-bbox="935 1346 1487 1520">■ Clear this check box to specify that Universal Messaging or Broker (deprecated) stores process transition documents and referenced process start documents on the local hard disk drive. <li data-bbox="886 1547 1487 1650">■ Volatile Tracking: Store process tracking information in memory only. Disabled by default. <p data-bbox="935 1682 1487 1869">Note: If the Process Engine is running in a clustered environment, volatile tracking cannot be used, and this setting is ignored.</p>

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ Select Volatile Tracking to specify that the Process Engine stores process status in RAM. ■ Clear this check box to specify that the Process Engine stores process status in the Process Engine database component. ■ Minimum Logging Level: Specifies the minimum audit logging threshold that can be set for this process in Monitor (that is, during run time). The default is 5 - Process and all events, activities, and looped activities. Available values are: <ul style="list-style-type: none"> ■ 1 - None ■ 2 - Errors only ■ 3 - Process only ■ 4 - Process and start events ■ 5 - Process and all events, activities, and looped activities (default) <p>For more information, see “About Process Model Data Logging” in the PDF publication <i>webMethods Monitor User’s Guide</i>.</p>
	<p>Runtime Editable Properties</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: All settings in this section can also be modified in Monitor. When you build and upload the process, the settings you define here will replace the existing run-time settings in the Process Audit database.</p> </div>	<ul style="list-style-type: none"> ■ Synchronize: Click this button to load the settings for the properties in this section from the Process Audit database. For more information, see “About Synchronizing Process Runtime Settings with webMethods Monitor ” on page 66. ■ Emit Process-specific Predefined EDA Events: <ul style="list-style-type: none"> ■ Select All/Clear All: Click to select or clear all EDA (deprecated) events. ■ Select or clear a predefined event type check box to specify whether or not the predefined event is to be emitted for this model. For more information, see “Enabling and Disabling Predefined

Properties Page	Property	Description
		<p>EDA Event Emission for a Process Model” on page 321.</p> <ul style="list-style-type: none"> ■ Steps Enabled for Resubmission <ul style="list-style-type: none"> ■ Select or clear a step check box to specify whether or not the step is enabled for resubmission. For more information, see “Enabling a Step for Resubmission” on page 76.
	Deprecated properties	<p>Enable deprecated error handling behavior (not recommended): This option is not recommended except in very isolated cases. For more information, see “About Step Failure Behavior” on page 157.</p>
	RosettaNet	<ul style="list-style-type: none"> ■ Synchronous: Select this setting to generate an internal process flag called “synch” with a value of true. The Process Engine then passes this value to the RosettaNet Conversation Manager private data structure used to communicate with RosettaNet. Not selected by default.
Global Process Specification		<p>Use this page to define the inputs and outputs for a callable process. A global process specification, along with a start none event, is used when a callable process is invoked by a call activity step.</p> <p>The input specification for a callable process determines the data that flows into the start none event in the callable process when it is triggered by a call activity. The output specification for a callable process determines the data that flows out of the callable process and back to the call activity that triggered it. The entire pipeline is automatically sent back to the call activity.</p> <p>Adding, editing, and removing inputs and outputs for a global process is done the same way as it is for steps. Refer to “Create Inputs and Outputs” on page 91 and “Remove Inputs and Outputs” on page 92.</p> <p>For more information, see “Defining a Global Process Specification” on page 97.</p>

Properties Page	Property	Description
Default Deployment Values	Process will be execution-enabled by default during deployment by Deployer	Select this check box to automatically enable the process for execution when deployed by Deployer.
	Process will be analysis-enabled by default during deployment by Deployer	Select this check box to automatically enable the process for tracking and analysis when deployed by Deployer.
		<p>Note: This feature is associated with webMethods Optimize for Process. Analysis-enabled processes can be tracked by Business Activity Monitoring by the Optimize Analytics engine. For more information, refer to <i>webMethods Optimize User's Guide</i>.</p>

About Synchronizing Process Runtime Settings with webMethods Monitor

You can create, modify, and delete stage, EDA (deprecated) event, and step enablement runtime settings in two locations:

- In Software AG Designer, on the **Stages** page in the Property view, and on the **Runtime** page in the Property view.
- In webMethods Monitor, as part of the Business Process administration functionality. For more information, see the chapter “Working with Process Models” in the PDF publication, *webMethods Monitor User's Guide*.

In both cases, any changes to these runtime settings in a process model can be saved to the Process Audit database. The saved changes overwrite whatever previous setting information was present in the database. As a best practice, you should ensure that your runtime settings are always synchronized between the two locations.

For Designer, the following conditions apply:

- When you open a process in Designer, it retrieves the runtime settings saved with the model in the local workspace.
- When you click the **Synchronize** button on the **Stages** page in the Property view, or in the **Runtime Editable Properties** section of the **Runtime** page in the Property view, Designer retrieves the runtime settings from the database and applies them to the process model. Saving the process model saves the settings to the local workspace.
- When you build and upload a process in Designer, the runtime settings in the model are written to the Process Audit database, overwriting whatever settings are stored there and applying

those changes to the runtime environment. In addition, a warning message is recorded in the build and upload report.

Therefore:

- If you want your process model in Designer to display the current database runtime stage settings, you must click the **Synchronize** button on the **Stages** page of the Property view. For more information, see [“Synchronizing Stage Settings” on page 356](#).
- If you want your process model in Designer to display the current runtime EDA (deprecated) event emission and step submission enablement values, you must click the **Synchronize** button on the **Runtime Editable Properties** section of the **Runtime** page in the Property view. For more information, see [“Enabling and Disabling Predefined EDA Event Emission for a Process Model” on page 321](#).
- If you want your process model runtime settings in Designer to be written to the database, click  to build and upload the process model. This overwrites whatever settings are stored in the database, and the settings are applied to the runtime environment.

To help ensure that you are working with the latest settings, you are advised to click the appropriate **Synchronize** button immediately before you modify and save these settings.

Important:

Deleting a step that is contained in a process stage without first synchronizing the stage settings with the database can lead to the process being out of sync with edits performed in Monitor, resulting in unexpected behavior. Always click the stage settings **Synchronize** button immediately before you delete any steps in Designer that are contained in a stage.

Deleting a Process

➤ To delete a process

1. In Designer, if the Solutions view is not visible, display it: **Window > Show View >  Solutions**.
2. Right-click a  process and select ** Delete**.

To delete multiple processes at once, CTRL+click to select, then right-click and select ** Delete**.

Printing a Process

➤ To print a process

1. Click the process editor's canvas to select a process.
2. Click **File > Page Setup**.
3. In the Page Setup window, configure the following:

- In the Orientation section, select **Portrait** or **Landscape**. The default is **Landscape**.
 - In the Scaling section, select **Adjust to:** and then select a zoom percentage or select **Fit to page(s):** and select the number of pages across which to print the process. The default is **Fit to page(s): 1 across**.
 - In the Margin (inches) section, set the Left, Right, Top, and Bottom margins. The defaults for each of these is 0.0.
4. Click **OK** to save your settings.
 5. If you want to preview your process before printing, select **File > Preview**.
 6. Click **File > Print**.

Saving a Process Image

> To save a process image

1. Click the process editor's canvas to select a process.
2. Click **File > Save as Image**.
3. In the Save Canvas as Image window, navigate to the location where you want to save the image file, and type a **File name**, including the extension.

Valid extensions are SVG, JPG, JPEG, and PNG.
4. Click **OK** to save the image file.
5. An image of the process is also saved when you generate an HTML Documentation Report.

Copying a Process

You can copy an open process and save it with a new process name. The source process retains its mapping services, while the new target process has no mappings.

Important:

The **Save As** menu option creates a copy of the process only. It does not create a copy of all the underlying (and generated) services.

> To copy a process and save it with a new name

1. Click the process editor's canvas to select a process.
2. Click **File > Save As**

3. In the Save As window, do the following:
 - a. Enter or select the process project in which to save the process. The process project must exist; you cannot create a new process project in this window.
 - b. Enter the file name (.process name) of the new process. The field is pre-populated with the file name of the source process.
4. Click **OK** to save the process file.

Using E-forms in a Process

Designer supports the use of Microsoft InfoPath and Adobe LiveCycle e-forms in processes. Both receive tasks that start processes and those that do not start processes can use e-forms. Message and signal start events can use e-forms to start processes. Message and signal intermediate events -- boundary and top-level -- can also use e-forms.

Important:

User tasks can also use e-forms. E-form support in a user task must be added in the task editor, even if the process is e-form driven. For information about e-form support to a task, see "Adding E-form Support to a Task" in *webMethods BPM Task Development Help*. For more information about using e-forms, see the PDF publication *Implementing E-form Support for BPM*.

E-forms are created from templates; those templates contain information required by the process using the e-form. When an e-form is used in a process, it is an *instance* of that e-form. E-form templates and e-form instances both need to be accessible to the process, and it is important that each has a separate location.

> To use an e-form in a process

In Designer, do one of the following:

- Drag and drop an existing e-form sourced IS document from the Package Navigator view onto the process editor to create a message start event pre-configured to use an e-form. The **E-form (For E-form Triggered Processes)** check box is automatically selected, and the **E-form Template Name** field is automatically populated. You must select the **E-form Content Repository** location.

OR

- Drag and drop an existing e-form sourced IS document from the Package Navigator view onto an existing receive task, message / signal start event, or message / signal intermediate event (including boundary events) to configure it to use an e-form. The **E-form (For E-form Triggered Processes)** check box is automatically selected, and the **E-form Template Name** field is automatically populated. You must select the **E-form Content Repository** location.

OR

- Manually configure an existing receive task, message / signal start event, or message / signal intermediate event (including boundary events) to configure it to use an e-form. You must select the **E-form (For E-form Triggered Processes)** check box. The **E-form Template Name** field is automatically populated when you select an e-form sourced IS document as the step's **Receive Document**. You must select the **E-form Content Repository** location.

Configuring E-forms in a Process

> To manually configure an e-form

1. On the Implementation page in the Properties view of a receive task, message / signal start event, or message / signal intermediate event (including boundary events): click **Browse** to select an e-form sourced IS document as the step's **Receive Document** or **New** to create a new e-form sourced IS document and select it as the step's **Receive Document**.

Important:

You must use **Browse** or **New** to select the e-form sourced IS document as the step's **Receive Document**. If you type the **Receive Document** name, the e-form fields are not enabled.

Important:

Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task or start / catching message or signal event.

2. Select the **E-form (For E-form Triggered Processes)** check box to enable e-form support in the step.
3. Select the **E-form Content Repository** location. See [“Selecting an E-form Content Repository” on page 70](#).

Selecting an E-form Content Repository

> To select an E-form Content Repository

- On the Implementation page in the Properties view of a receive task or message start event, click **Browse...** to select a configured content repository as the e-form's **E-form Content Repository**.

Software AG Designer's e-form solution works with My webMethods Server as the e-form content repository, through the JSR-170 interface.

For more information about e-form content repositories, see the PDF publication *Implementing E-form Support for BPM*.

6 Process Steps

■ Process Step Overview	72
■ About Step Labels	72
■ Moving Process Steps Using the Keyboard	73
■ Cutting or Copying Process Steps	73
■ Pasting Process Steps	74
■ Deleting a Process Step	75
■ Enabling a Step for Resubmission	76
■ Assigning Custom Step Ornaments	77

Process Step Overview

You create steps on the process editor's canvas by dragging a step type from the Palette view to the canvas and connecting the steps with transitions to create a process. Steps are categorized by what they do, specified in their properties, and also by their function in the process.

If you are a business analyst, you can add steps in the Business Analyst mode to model the process with a minimum of technical configuration for each step. Business model developers who work in the Process Developer mode have access to the advanced properties for each step to enable full technical configuration. For more information, see [“About Process Development Modes” on page 52](#).

For more information about the types of steps that are available, see these topics:

- [“Activity Steps” on page 211](#)
- [“BPMN Event Steps” on page 263](#)
- [“Gateway Steps” on page 339](#)

For more information about labels and transitions, see:

- [“About Step Labels” on page 72](#)
- [“About Transitions” on page 117](#)

About Step Labels

Software AG Designer enables you to apply a label to each step in your process model. It is possible for step labels to be empty, and for the same label value to be used more than once in the same process model. This enables a closer integration with ARIS process models and enables models to be more accurately imported from other modeling tools in XPDL format.

Note:

In version 8.2.x and earlier, each step was required to have a unique step name as indicated by the label.

You can add, remove, or modify a label value at any time by clicking the step to select it, and then editing the step **Label** field on the **General** page in the Properties view.

In those locations outside of the process design canvas where step labels are referred to, the following rules apply:

- If a step has no label but generates a flow service, the step ID is used in place of the step label. You can view the step ID value by hovering the cursor over as step, or by selecting the step and clicking the **General** tab in the Properties view.
- If there are duplicate steps of the same name, each step generates its flow service with a value of "<step label>_<stepID>".

Task, call activity, and subprocess steps are always created with a default label, for example, “Task1,” with subsequent steps numbered incrementally. Event and gateway steps can be created

with or without a label. When applied, the label uses the same format, for example, Gateway1, Message Event1.

Note:

If you delete a step with a default name, that name is not reused in the process model. For example, if you add steps named Task1 and Task2 and then delete Task2, the next step is named Task3.

Label behavior is governed by the following preferences:

- The **Default step label location** preference determines the default placement of task, call activity, and collapsed subprocess step labels (on the step or below the step). In addition, you can set the position of the step labels in a process with the **Position label on step/below step** button  on the tool bar. For more information, see [“Changing Task Label Placement” on page 217](#).
- The **Automatically update step names when adding documents/services via drag and drop** preference determines whether step labels change after a drag and drop action.
- The **Show event and gateway labels by default** preference determines whether or not a label is created when an event or gateway step is added to the process.

For more information about these preferences, see [“Configuring Appearance Preferences” on page 36](#).

Moving Process Steps Using the Keyboard

The ability to move process steps is native Eclipse behavior. Use the following Eclipse procedure to fine-tune the position of process steps in your model.

➤ To move process steps using the keyboard

1. Click to select a single process step in the process editor. To select two or more steps, CTRL+click, SHIFT+click, or use the marquee tool on the palette to draw a marquee around the steps you want.
2. Press the period (.) key on the keyboard.
3. Press the up, down, left, and right arrow keys on the keyboard to move the selected steps.
4. To cancel the move, click an empty spot on the process editor's canvas.
5. To complete the move, press the Enter key or the period (.) key.

Cutting or Copying Process Steps

When you cut or copy a step, all step properties are cut or copied as well, including its label, icon, step type, description, documentation fields, logged fields, and inputs and outputs. Outgoing transitions and transition conditions from the step are included if the target step is selected for cut or copy at the same time.

Important:

If the target step is not selected, outgoing transitions and transition conditions are lost when you paste the step. Transition lines must be cut or copied with their source and target steps; you cannot cut or copy and paste them separately.

> To cut or copy one or more process steps on the process editor's canvas

1. Select a single step or note

OR

CTRL+click, SHIFT+click, or use the cursor to draw a marquee around steps and/or notes to select multiple steps or notes.

2. CTRL+C, **Edit > Copy**, or right-click and select **Copy**

OR

CTRL+X, **Edit > Cut**, or right-click and select **Cut**.

Note:

You can undo a cut with CTRL+Z, **Edit > Undo Cut**, or right-click and select **Undo Cut**.

You can also “redo” a cut that you undo with CTRL+Y, **Edit > Redo Cut**, or right-click and select **Redo Cut**.

Pasting Process Steps

When you paste a process step, some properties of the cut or copied assets are modified:

- Step IDs are recalculated to use a new, unique ID in the target process
- Transition IDs are recalculated to use a new, unique ID in the target process
- Transition references are updated to the new source and target Step IDs
- If the target process has a data item of the same name as one of the pasted steps, the pasted step updates its references to use the existing data item
- If the target process has a data item of the same name as the pasted assets, but with different type or description information, Designer renames the pasted data item, by appending a suffix of “_<number>”. So a pasted “myInput” becomes “myInput_1”.

> To paste one or more process steps on the process editor's canvas

Do one of the following:

- Right-click and select **Paste** to paste to a specific location on the canvas.
- CTRL+V or **Edit > Paste** to paste to an unspecified location on the canvas.

When you paste process steps, Designer automatically selects them so you can easily relocate them on the process editor's canvas if you want to.

Relative positions of multiple steps is preserved when you paste. If the target process editor's canvas has an asset selected, steps and/or notes are pasted relative to the selected asset. If there is nothing selected on the target canvas, steps and/or notes are pasted relative to the top left of the canvas. If you paste a step or a note very close to the edge of a pool, swimlane, or expanded subprocess, the container expands to accommodate the pasted item.

Note:

You can undo a paste with CTRL+Z, **Edit > Undo Paste**, or right-click and select **Undo Paste**. You can also redo a paste that you undo with CTRL+Y, **Edit > Redo Paste**, or right-click and select **Redo Paste**.

Deleting a Process Step

You can delete a process step at any time. When you delete the step, all of its incoming and outgoing transition lines are deleted, along with all other step information. Designer provides the standard Eclipse Undo/Redo functionality, so you can undo your deletion, even after you save the process. You cannot undo the deletion after you close your current Designer session.

Important:

When you delete a step that contains a start milestone or an end milestone marker for a process stage, *that process stage is deleted along with the step* (you are prompted for confirmation before the stage is deleted). If the step contains multiple milestone markers, all of the associated stages are deleted. For more information, see [“Stages and Milestones” on page 351](#).

Important:

Deleting a step that is contained in a process stage without first synchronizing the stage settings with the database can lead to the process being out of sync with edits performed in Monitor, resulting in unexpected behavior. Always click the stage settings **Synchronize** button immediately before you delete any steps in Designer that are contained in a stage. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor” on page 66](#).

» To delete one or more process steps on the process editor's canvas

Do one of the following:

1. Open the process model you want to work with in the process editor.
2. Select the step or steps you want to delete. Hold down the CTRL key while you click to multi-select.
3. Do any of the following:
 - Right-click and click **Delete**.
 - Press the DELETE key on your keyboard.

- Click **Edit > Delete** in the main menu.

4. Save the process.

Enabling a Step for Resubmission

webMethods Monitor enables you to resubmit a process instance that has failed or completed, provided you have first enabled at least one step in the process model for resubmission *before the process instance began executing*. This setting has no effect on process instances that are currently running or that have already stopped running.

In addition to enabling steps for resubmission in Software AG Designer as described below, you can also enable steps for resubmission in Monitor.

- For more information about enabling and disabling steps for resubmission in Monitor, see the chapter “Working with Process Models” in the PDF publication *webMethods Monitor User’s Guide*.
- For more information about step resubmission in general, see “About Resubmitting Process Instances and Process Steps” in the PDF publication *webMethods Monitor User’s Guide*.

Important:

Before you make modifications to step resubmission settings, be aware of the interaction of these settings between Designer and webMethods Monitor. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor”](#) on page 66.

Note:

If you want to be able to resubmit process instances from Monitor at certain steps, you must set the process model logging level to a level that will log the input pipelines for those steps. For more information, see “About Process Model Data Logging” in the PDF publication *webMethods Monitor User’s Guide*.

➤ To enable or disable a step for resubmission in Designer

1. Open the process model you want to work with in the process editor.
2. Click anywhere in the process canvas to select the entire process.
3. In the Properties view, click the **Runtime Settings** tab, and then in **Steps Enabled for Resubmission** area of the **Runtime Editable Properties** section:
 - Click **Synchronize** to retrieve the current runtime settings from the Process Audit database and apply them to EDA (deprecated) events and step enablement areas. A confirmation dialog box appears to inform you either that the settings are identical and no changes are needed, or that the settings are different, in which case you can choose to synchronize the settings or cancel.
 - Select the corresponding check box for a step you want to enable for resubmission.

- Clear the corresponding check box for a step to disable resubmission capability.

Note:

When a step is enabled for resubmission, the pipeline data for that step is saved. Extensive use of resubmittal enablement (for example, enabling all steps for all process models for resubmittal) may result in a reduction in performance.

4. Save the process model.

You must build and upload the process model to apply the new settings to the runtime environment. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor ” on page 66.](#)

Assigning Custom Step Ornaments

You can assign custom activity step images known as *ornaments* to replace the small images Designer displays by default. You can also remove an existing ornament image. You can select from a set of system images Designer provides, or you can use your own images, as described in [“ Creating a Custom Image Set for Activity Steps” on page 45.](#)

You can assign an ornament to any of the activity step types, with the exception of subprocesses. You cannot assign an ornament to event and gateway steps.

➤ To assign an activity step ornament

1. In the process editor, right-click the activity step.
2. Click **Choose Image**.
3. Do one of the following:
 - If you want to remove an existing ornament, click **Restore Defaults**.
 - If you want to apply a new ornament and have not added any custom images, click one of the images available on the **System** tab.
 - If you have added custom image files to your Designer installation as described in [“ Creating a Custom Image Set for Activity Steps” on page 45,](#) click the tab for the jar file or directory that contains the custom image files and click one of the available images.
 - Click **Browse** to locate an image in any folder in your file system.
4. Click **OK**.

7 The Process Editor

■ Using the Process Editor	80
■ Using the Palette	80
■ Working with Speed Buttons	80
■ Resizing Objects on the Canvas	81
■ Moving Process Steps Using the Keyboard	82
■ Changing an Activity, Event, or Gateway Type	83
■ Using the Canvas Clipboard	84
■ Using Keyboard Shortcuts	85
■ Toolbar Buttons	86

Using the Process Editor

When you develop processes in Designer, you use an Eclipse editor. Process Development editor is also known as the process editor. It is also sometimes referred to as the *canvas*. The word *canvas* is also sometimes used to refer to empty space in the process editor.

You can scroll the visible canvas in the process editor; you can also modify the way Designer displays items in the process editor with buttons on the toolbar.

Using the Palette

The process development palette contains most of the key process components you need to build a business process model, including:

- Task activities
- Call activity
- Subprocess activity
- Events
- Gateways
- Transitions
- Notes and annotations

Note:

The palette does not contain an entry for all possible boundary events. For more information, see [“Adding a Boundary Intermediate Event” on page 289](#).

➤ **To add a palette object to the process editor's canvas**

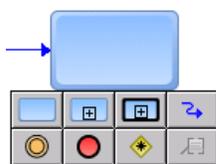
1. On the process editor palette, click the object you want to add to your process.
2. Click the process editor canvas where you want to place the new object.

Tip:

To add multiple instances of an object from the palette, hold down the CTRL key when you place the first instance on the canvas. You can then add an additional object of that type with each successive click as long as the CTRL key is held down.

Working with Speed Buttons

The Process Editor provides you with an array of speed buttons that appear when you hover the cursor over an activity, subprocess, event, or gateway step, for example:

**Note:**

The speed button toolbar will not appear when a step is near the left-hand edge of the process canvas. Move the step toward the center of the canvas to view the toolbar.

These speed buttons provide fast and convenient access to a number of process design activities, including the ability to:

- Create a new activity, event, or gateway with a transition line from the associated step. For more information, see [“About Activities” on page 212](#), [“About BPMN Events” on page 264](#), and [“About Gateways” on page 340](#).
- Draw transition lines from the associated step. For more information about transitions, see [“About Transitions” on page 117](#).
- Add a text annotation. For more information about annotations, see [“About Notes and Annotations” on page 368](#).

Note:

You cannot create a start event from the speed buttons. The purpose of the speed buttons is to be able to add a process object such as an activity, event, or gateway, with an automatic transition from the source object. Because start events have no incoming transitions, they are not available as a speed button.

The buttons are fixed in position and order. However, the process model object for each button changes to always show the last process object selected from the palette. For example, if you add an inclusive gateway to the canvas from the palette, the next time you display the speed buttons, the gateway button will be set to inclusive gateway.

If the object displayed is not the one you want, you can place it on the canvas anyway and then right-click the object and click **Change [Object] Type** to specify the object type you want.

In addition to buttons for creating an activity, subprocess, call activity, intermediate or end event, and gateway, buttons are available to create a text annotation or a transition line.

Resizing Objects on the Canvas

You can resize the following objects on the process editor canvas:

- Activities, events, and gateways. For more information, see [“Resizing Activities, Events, and Gateways” on page 82](#).
- Pools. For more information, see [“Resizing Pools” on page 361](#).
- Swimlanes. For more information, see [“Renaming Swimlanes” on page 364](#).
- Notes and annotations. For more information, see [“Working with Notes” on page 368](#).

In addition, you can also manipulate transition lines. For more information, see [“About Transitions” on page 117](#).

Resizing Activities, Events, and Gateways

➤ To resize an activity, event or gateway on the process editor canvas

1. Select the object you want to resize on the process editor canvas. You can also multi-select (CTRL+click) two or more objects. Do not select any transition lines.
2. Do one of the following:
 - Click to select one of the eight available object handles and drag the handle to resize the object to a custom size (non-proportional).
 - Select one of the four object corner handles, hold down the SHIFT key, and drag the handle to resize the object proportionately.
3. To return an object to its default size, do one of the following:
 - Right-click the object and click **Reset Default Size** in the context menu.
 - Right-click the object and click **Reset Default Size** on the **General** page in the Properties view.
 - Multi-select (CTRL+click) two or more resized objects, right-click, and click **Reset Default Size**. Do not select any transition lines, or the **Reset Default Size** command will not be available.
4. Save the process.

Moving Process Steps Using the Keyboard

The ability to move a process step with the keyboard is native Eclipse functionality. That is, the functionality is not contributed by the Process Development feature. The procedure is presented here for convenience.

➤ To move process steps using the keyboard

1. Click a single process step in the process editor to select it.
2. Press the period (.) key on the keyboard.
3. Press the up, down, left, and right arrow keys on the keyboard to move the selected steps to its new position, then do one of the following:
 - To cancel the move, click an empty location on the process editor's canvas.

- To complete the move, press the Enter key or the period (.) key.

Tip:

After you select a step, the first press of the period (.) key selects the step for moving. For each subsequent press of the period (.) key, the focus will move through each of the resizing handles on the step.

Changing an Activity, Event, or Gateway Type

As you develop a process model on the process editor canvas, you might find that a task activity, event, or gateway you have added to the process is not the correct type. You can always delete the task activity, event, or gateway and add the desired type in its place, but Designer enables you to change the task activity, event, or gateway type from within the task activity, event, or gateway itself.

Note:

There are limitations to the changes you can make from within an boundary intermediate event. For example, a non-interrupting intermediate boundary error event is not supported on a service task activity type. Therefore, the non-interrupting intermediate boundary error event is not present on the list of available choices in the right-click menu, nor can you configure the interrupting behavior on the **General** page of the Properties view. For more information, see [“Available Change Type Selections” on page 83](#).

➤ To change a task activity, event, or gateway type

1. Locate the task activity, event, or gateway on the canvas and do one of the following:

Note:

For additional information and procedures for changing task activities, see [“Changing Task Types” on page 215](#). For the supported boundary event types, see [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

- Right-click the activity, event, or gateway and click **Change [Task, Event, or Gateway] Type**. Click the desired task, event, or gateway type in the resulting selection menu(s).
- Click the task activity, event, or gateway. On the **General** page of the Properties view, select the desired task or event in the **Type** list. Select the **Allow [boundary event type] to interrupt step** check box if you want interrupting behavior (not available for all boundary event types).

2. Save the process model.

Available Change Type Selections

The following change types are available for each of the listed activity, event, and gateway types:

- **Subprocesses.**

To change a subprocess to a different type, you must delete the subprocess and add the new step manually. However, you can change a BPMN subprocess (the default) to a webMethods subprocess. For more information, see [“About Subprocess Types” on page 146](#).

- **Call Activities.**

To change a call activity to a different type, you must delete the call activity and add the new step manually. However, you can change the invocation behavior of a call activity step between callable process behavior and webMethods referenced subprocess behavior. For more information, see [“Call Activity Concepts” on page 147](#).

- **Task activities.** You can select from:

Abstract, Service, Receive, Send, User, Rule, or Manual. To change to any other step type, you must delete the task and add the new step manually.

- **Start events.** You can select from:

None, Message, or Signal. To change to any other step type, you must delete the start event and add the new step manually.

- **Intermediate events.** You can select from:

None, Message, or Signal. To change to any other step type, you must delete the intermediate event and add the new step manually.

- **End events.** You can select from:

None, Message, Signal, Error, or Terminate. To change to any other step type, you must delete the end event and add the new step manually.

- **Gateways.** You can select from:

Complex, Inclusive, Exclusive, or Parallel. To change to any other step type, you must delete the gateway and add the new step manually.

- **Boundary Intermediate Events.**

In BPMN 2.0, boundary intermediate events can be interrupting or non-interrupting. Designer does not support both behaviors for all boundary intermediate events. Only the supported behavior is available in the **Change Event Type** selection menu. To change to any other step type, you must delete the boundary intermediate event and add the new step manually.

The available change types for boundary intermediate events vary depending on the placement of the boundary intermediate event. For specific information about the supported interrupting behavior, see [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

Using the Canvas Clipboard

You can use the process editor canvas clipboard to cut and copy process steps and paste them in the same process or a different one.

You can cut and copy and paste all process steps, including notes and annotations. You cannot cut or copy and paste transitions. Use the Transitions tool in the process editor palette to add a transition.

Using Keyboard Shortcuts

You can use the keyboard shortcuts listed in the table below, to navigate and perform actions in the process editor.

Keyboard	Function
Period (.)	Move a process step. For more information, see “Moving Process Steps Using the Keyboard” on page 82.
ALT+U	 Upload for Analysis Only
ALT+K	 Upload KPIs
ALT+G	 Build and upload for execution Requires Process Developer mode. See “Configuring the Process Development Mode” on page 35.
ALT+F11	 Debug Selected Process Requires Process Developer mode. See “Configuring the Process Development Mode” on page 35.
ALT+ left arrow	 Back
ALT+ right arrow	 Forward
ALT+S	 Simulate Process Note: Process simulation requires the Process Simulation feature. See “About Process Simulation ” on page 498.
CTRL+F8	Displays a list of perspectives from which to select. Tip: Keep the CTRL key pressed down after you press F8 to keep the list visible. After you press an arrow key to navigate the list, you can stop holding down CTRL. Press Enter to open the selected perspective. To appear in the list, a perspective must have already been used at least once.

Toolbar Buttons

Most Designer toolbar buttons are always available. Some, however, are available in some perspectives but not in others, and some are available based on the editor that has focus. Process Development views can have their own toolbars and buttons as well.

The following table lists and describes the toolbar buttons.

Toolbar Button	Description
	Search
	Back
	Forward
	Canvas Zoom Default is 100%.
	Auto-layout the process Default (Horizontal or Vertical) is set in Preferences. See “Configuring Appearance Preferences” on page 36 .
	Upload for Analysis Only
	Upload KPIs
	Build and upload for execution Requires Process Developer mode. See “Configuring the Process Development Mode” on page 35 .
	Position label on step/below step Controls the placement of step labels on the canvas. The default setting is to display a label beneath the associated object on the canvas. Click this button to change the current label location. For more information, see “About Step Labels” on page 72 .
	Show/hide inputs and outputs
	Debug Selected Process Requires Process Developer mode. See “Configuring the Process Development Mode” on page 35 .
	Refresh Documents Click to refresh all Integration Server (IS) documents used in the process currently displayed in the process editor.

Toolbar Button	Description
	Requires Process Developer mode. See “Configuring the Process Development Mode” on page 35.
	Business Analyst Mode Show only basic properties, preferences, and functions. See “Configuring the Process Development Mode” on page 35.
	Process Developer Mode Show advanced properties, preferences, and functions. See “Configuring the Process Development Mode” on page 35.
	Simulate Process If you have Process Simulation installed, you can click to create a process simulation from an open process in the process editor. For more information on process simulation, see <i>webMethods BPM Process Simulation Help</i> .

8 Step Inputs and Outputs

■ About Inputs and Outputs	90
----------------------------------	----

About Inputs and Outputs

Each step in a process has information that flows into and out of it. Information flowing into a step is called *input*, and information flowing out of a step is called *output*. A process itself can also have inputs and outputs, such as when calling a process from a call activity step.

Process data assigned in Designer to flow in and out of steps needs to be mapped to *physical data* that the underlying services require in order for the process to execute.

Step inputs and outputs are used to define flow signatures, branching and looping logic in the process, data logging for examination at run time, and KPIs. See [“About Inputs and Outputs” on page 90](#) and [“About KPIs” on page 372](#).

Step inputs and outputs are used to generate the signatures for the generated services that implement the process. If the underlying implementation of the step requires different physical data than this process data, the data must be mapped in the generated flows.

Process data follows a pipeline model, where all data that is input to a step must be output upstream in the process from that step.

Data can therefore enter the process in two ways:

- In a receive step, a subscription document can trigger or join the process, and output data for that step and into the pipeline
- In an activity step, the step can output new process data into the pipeline

While you can add new inputs to any step, the process will not be valid (e.g., ready to be built) until all step inputs are first selected as outputs of an upstream receive or activity step.

Designer can automatically map inputs and outputs in the following circumstances:

- Step A is linked to step B, and the output of step A has the same name as the input of step B
- An activity step input name is the same as the document or service input name
- A Receive Task output document is the same type as its incoming document type

In all but these cases, you must manually map step input and output data. On the Inputs / Outputs page of the Properties view of the step whose data you want to map, select **Edit Data Mapping**. Alternatively, right-click the step and select **Edit Data Mapping** from the context menu.

Note:

By default, when you add a step to a process model and then **Edit Data Mapping**, the step in the model is saved if the **Save before building/uploading processes** Build and Upload preference is enabled. If it is not enabled, Designer prompts you to save the step. See [“Configuring Build and Upload Preferences” on page 40](#).

When you edit the data mapping for a service, Designer saves the process definition. It then creates a generated mapping service on the Integration Server that uses the step inputs and outputs as the new service inputs and outputs. It also puts the invoked service (specified in the **Service** property) inside this mapping service, and adds a flow map step. Designer then opens the Pipeline

view to allow you to map the step inputs and outputs to the invoked service inputs and outputs. See "What Does the Pipeline View Contain?" in *webMethods Service Development Help*.

By default, Designer does not display generated flow services in the Package Navigator view. You can view them by editing the Package Navigator filter. Click  **Filter contents of Navigator** and clear the **Hide generated flow services** check box in the Choose Elements to Display window. You can also set a preference for this behavior in **Window > Preferences > Software AG > Service Development > Package Navigator**. See "Package Navigator Preferences" in *webMethods Service Development Help*.

After a mapping service has been created, you can right-click its step and select **Edit Mapping Service** to open it in the flow service editor and update the generated service signature to match the step's current inputs and outputs. Another method is to select **Edit Data Mapping** on the Inputs/ Outputs page of the Properties view of the step whose data you want to map.

Note:

Mapping services are not intended to contain business logic. The mapping service editor does not prevent you from adding other flows inside a mapping service. However, all reusable business logic should be placed in the invoked service, not in the mapping service.

When you build and upload a process, Designer automatically updates the mapping services of all steps to include the current step signatures, and generates mapping services for all steps that do not yet have them defined.

For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.

Show and Hide Inputs and Outputs

You can configure whether to show or hide step inputs and outputs by default in the Preferences window, and you can also toggle the show/hide behavior using a button on the tool bar.

> To show and hide step inputs and outputs

1. To set the default behavior for showing inputs and outputs, click **Window > Preferences > Software AG > Process Development > Appearance**, and select or clear the **Show inputs and outputs on steps by default** check box.
2. To show or hide step inputs and outputs in the open process, select or clear the **Show: Inputs/Outputs** check box on the process editor's toolbar.

Create Inputs and Outputs

Inputs and outputs are created in the same way, but they have different requirements due to their roles in a process. Outputs from steps create pipeline data, and are available to select as inputs to steps that are downstream in the pipeline. Inputs to all steps must exist upstream in the pipeline. If this is not the case, the issue is reported in the  Problems view.

To edit the data mapping of fields inside a document or service, select **Edit Data Mapping** on the Inputs / Outputs page of the Properties view of the document or service whose data you want to map. Alternatively, right-click the step and select **Edit Data Mapping** from the context menu.

For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.

> To create an input or output

1. Select a step in the process editor.
2. On the Inputs / Outputs page in the Properties view, click  **Create new input** in the Inputs section or  **Create new output** in the Outputs section.

Important:

All inputs must exist upstream in the pipeline. If you create a new input that does not yet exist upstream, you must create an output to feed the new input before completing the process.

3. Create a new input or output, or select an input from upstream in the pipeline:
 - If you create a new input or output, configure the **Name**, **Type**, and **Description**, and select the **List** check box if the input is an array. If you select a **Document Reference**, select the document from the Choose Document window.

Tip:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

- If you create an input based on an existing output from upstream in the pipeline, you do not need to configure the **Name**, **Type**, or **Description**. Designer populates the values automatically when you select the existing output.

Important:

Unnamed inputs and outputs are not saved.

Note:

Text entered in the **Description** field is included in the HTML Documentation Report.

Remove Inputs and Outputs

> To remove a step input or output

1. Select a step in the process editor.

- On the Inputs / Outputs page in the Properties view, click  **Remove input from step** in the Inputs section or  **Remove output from step** in the Outputs section.

Important:

If the inputs or outputs of a step are changed such that they do not match what is displayed in the process editor, the process will not refresh its inputs and outputs automatically. You must refresh them by editing the inputs or outputs on the step's Inputs / Outputs page in the Properties view. Remove the old inputs or outputs from the step, and use the  **Auto-populate based on service signature** button to assign the new inputs or outputs.

Auto-Populate Inputs and Outputs

You can automatically populate the inputs and outputs of a step from its underlying IS service, Web service, task, or rule. This underlying information is known as the *service signature*.

Auto-populating step inputs and outputs allows Designer to do the data mapping of the step inputs and outputs. If you do not auto-populate with the service signature, you must manually map the data to the appropriate service. Click the **Edit Data Mapping** link on the **Inputs/Outputs** page in the Properties view, or right-click a step and click **Edit Data Mapping**.

Note:

Most steps have a single **Edit Data Mapping** right-click menu option. Call activity steps and task steps have two mapping options in their context menus: **Edit Input Data Mapping** and **Edit Output Data Mapping**. Empty steps do not have data to map, so they have no data mapping capability.

For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.

➤ To auto-populate a step input or output

- Select a step in the process editor.
- On the **Inputs / Outputs** page in the Properties view, click  **Auto-populate inputs based on service signature** in the Inputs section or  **Auto-populate outputs based on service signature** in the Outputs section.

Note:

Text entered in the **Description** field is included in the HTML Documentation Report.

Log Inputs and Outputs

In the Process Development perspective and in the Process Debugging perspective, you can select fields from input and output documents for logging. You can also create aliases for the logged fields, which makes locating them in webMethods Monitor easier.

Note:

These activities require the Process Developer mode. For more information, see [“Configuring the Process Development Mode” on page 35](#) and [“About Process Development Modes” on page 52](#).

Input and output field logging is part of the Process Engine audit logging mechanism. However, these fields are not subject to the **Minimum Logging Level** setting of the process model, but are specified on the process model’s **Logged Fields** page in the Properties view at design-time, as described below.

Logged fields can be viewed on the Process Instance Detail page in webMethods Monitor, and can also be used as KPIs in webMethods Optimize.

Note:

Before you can select input and output document fields for logging, you must first define step inputs and outputs.

➤ **To select a step input or output document field for logging**

1. Select a step in the process editor for which you have defined inputs and outputs.
2. On the **Logged Fields** page in the Properties view, click  **Expand** to expand the **Inputs** and **Outputs** trees to display the fields available in the documents.
3. Select the check boxes that correspond to the document fields you want to log.
4. If you want to define an alias for a document field, type an **Alias** name.

The alias defaults to the name and path of the selected field, but it can be modified to any alias for viewing in webMethods Monitor.

Note:

You can create the same alias for more than one field on a step, but this is not recommended, as it will make monitoring the fields at run time difficult.

Note:

When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the **Logged Fields** page in the Properties view. Designer does not, however, automatically remove an output field from the **Logged Fields** page in the Properties view if you remove it from a KPI or dimension.

Input and Output Types

The following step input / output types are available when configuring a step input / output. To select a list, choose the Input / Output type and then select the **List** check box.

The table below describes the input and output types.

Input / Output Type	Description
 Boolean	True or false.
 Boolean list	A one-dimensional boolean array.
 Byte	Signed integer. The value must be greater than or equal to -128 but less than or equal to 127.
 Byte list	A one-dimensional byte array.
 Char	A single unicode character.
 Char list	A one-dimensional character array.
 Date	Date and time.
 Date list	A one-dimensional date array.
 Double	Double-precision floating point number.
 Double list	A one-dimensional double array.
 Float	Standard-precision floating point number.
 Float list	A one-dimensional float array.
 Integer	Signed integer. The value must be greater than or equal to -2147483647 but less than or equal to 2147483647.
 Integer list	A one-dimensional integer array.
 Long	Signed integer. The value must be greater than or equal to -9223372036854775808 but less than or equal to 9223372036854775807.
 Long list	A one-dimensional long array.
 Short	Signed integer. The value must be greater than or equal to -32768 but less than or equal to 32767.
 Short list	A one-dimensional short array.
 Object	A data type that does not fall into any of the data types described in this table, and is not declared to be one of the basic Java classes supported natively by Integration Server.
 Object list	A one-dimensional object array.
 Document Reference	Select an Integration Server document type in the Choose Document window. The document you select becomes the input or output type.

Tip:

Input / Output Type	Description
	When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.
 String	A string of characters.
 String list	A one-dimensional string array.
 Unknown	An input or output of unknown type.
 Unknown list	A one-dimensional unknown array.

Specifying Referenced Process Start and Return Documents

When you configure a call activity step to statically invoke a webMethods referenced process, you must also specify start and return documents. Start documents are sent to the referenced process, and return documents are returned from the referenced process to the original process. Start and return documents are not needed when calling a BPMN callable process. A BPMN callable process requires a Global Process Specification in order to be called by a call activity step. See [“Defining a Global Process Specification” on page 97](#).

Important:

Although still available, the ability to invoke a referenced process from a call activity step is deprecated. You are advised to implement all of your steps that invoke another process with a call activity step that invokes a callable process.

As with other steps, the inputs of a call activity step configured to invoke a referenced process can be any available pipeline process data from upstream in the process. Such a call activity step must also publish the specified receive document to invoke another process.

If the call activity step input document(s) matches the selected start document(s) (and has values for all the required fields in the receive document), then no mapping is required after the process is built. If the input document does not match the start document, then you must map the data manually. Click the **Edit Data Mapping** links on the Inputs / Outputs page in the Properties view, or right-click a step and select **Edit Input Data Mapping** or **Edit Output Data Mapping**.

For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.

Similarly, the outputs of a call activity step can be any process data you choose to include. But if the outputs do not match the selected return document(s), they will require mapping after you build the process.

Available start documents are the sum of the subscription documents for all of the start event steps in the referenced process that can start new instances of the process. Available return documents are the sum of all the publication documents in the referenced process.

➤ To specify referenced process start and return documents

1. Select a call activity step in the process editor. The step must be configured to statically invoke a webMethods reference process. For more information, see [“Call Activity Concepts” on page 147](#).
2. In the **Available Input Documents** section on the **Start / Return Documents** page in the Properties view, select the referenced process subscription documents that should be used to invoke the referenced process (CTRL + click to select multiple documents), and click **Add** to move them to the **Inputs** section.
3. In the **Available Output Documents** section on the **Start / Return Documents** page in the Properties view, select the referenced process publish documents you want returned to the parent process (CTRL + click to select multiple documents), and click **Add** to move them to the **Outputs** section.
4. Save the process.

Defining a Global Process Specification

When you configure a call activity step to invoke a BPMN callable process, you also define a global process specification in the process you call. This includes inputs to and outputs from the callable process, allowing you to access process data.

The inputs of a callable process can be any available pipeline process data from previous steps in the process. Similarly, the outputs can be any process data you choose to include. The call activity step passes the entire pipeline to a start none event in the callable process, and the callable process automatically returns its resulting pipeline to the parent.

Designer automatically uses the defined global process specification (inputs and outputs) to populate the call activity step inputs and outputs. This happens when you drag and drop the child process onto the process editor canvas, or when you select the process on the **Implementation** page in the Properties view of the call activity step.

Tip:

Click the Auto-populate button  in both sections on the Inputs / Outputs page in the Properties view of the call activity step to update the inputs and outputs of the call activity to match the defined global process specification.

➤ To define a global process specification

1. In the process editor, open the process you want to configure as a callable process.
2. Click anywhere in the design canvas to select the process.

3. On the **Global Process Specification** page in the Properties view, specify the documents that should be used to invoke the callable process in the **Input Specification for Callable Process** section. For more information, see [“Create Inputs and Outputs” on page 91](#) and [“Remove Inputs and Outputs” on page 92](#).
4. In the **Output Specification for Callable Process** section specify the documents you want returned to the call activity that called the process (you may need to scroll to the right to see this section). For more information, see [“Create Inputs and Outputs” on page 91](#) and [“Remove Inputs and Outputs” on page 92](#).
5. Save the process.

9 Rules

■ About Rules	100
■ About webMethods Business Rules	100
■ About Rule Tasks	102
■ Using webMethods Business Rules in Processes	102
■ About Process Actions	102
■ About Manual Decisions	103
■ Creating a Process Action	103
■ Configuring a Remote Integration Server	105

About Rules

In Designer processes, you can use rules created with webMethods Business Rules. For detailed information on webMethods Business Rules and how they work, see *webMethods BPM Rules Development Help*.

About webMethods Business Rules

A webMethods Business Rule is a decision-making tree capable of complex behavior. Designing a rule is very straightforward. *Rule Entity* is the term used to describe the components. Specifically, those components are decision tables, rule sets, event rules, and rule actions. Each plays a specific role in the decision-making tree.

- Rule: a rule contains a decision table or a rule set with multiple tables.
- Decision table: a decision table can contain rule actions (Data, Service, and Process).
- Rule action: there are three types: Data actions, Service actions, and Process actions.
- Process action: something you do to a process. There are lots of options: start, join, suspend, cancel, fail, resume, call a task (manual decision or task action). These actions are constructed in the Rules Explorer view and can then be dragged from there into decision tables, which can then be used in rules.

Processes can be started in many different ways, including using rules and services. A process can invoke a rule, obviously, but a rule can also invoke a process.

For more on webMethods Business Rules and how they work, see *webMethods BPM Rules Development Help*. For detailed information about how to use them in processes, see [“Using webMethods Business Rules in Processes” on page 102](#).

About Decision Entities

In webMethods Business Rules, a *decision entity* can refer to a decision table, a rule set, an event rule, or a rule action. Decision entities are used to make decisions about the ways in which rules are applied.

For more information about working with decision entities, see *webMethods BPM Rules Development Help*.

About Decision Tables

Decision tables contain logic used to determine how a rule behaves. A rule may use a single decision table or multiple decision tables that work in concert. Decision tables can contain rule actions.

A decision table that contains a manual decision point (task action) needs to be configured in a particular way:

- The decision table must be process-aware. When creating the decision table, select the **Process Aware** check box.
- TaskData must be mapped.
- The task's Inputs/Outputs must match the decision table's Inputs/Outputs.

Note:

The decision table contains an extra parameter called ProcessData that is automatically generated when the decision table is created. This is mapped separately to the process, not to the task.

- The decision table's ProcessData must be mapped to the process.

Important:

Do not overwrite or drop ProcessData as it contains necessary runtime data. ProcessData mappings must be only outgoing.

For more information about working with rule actions and decision tables, see *webMethods BPM Rules Development Help*.

About Rule Sets

When multiple decision tables work together, they form a rule set.

For more information about working with rule sets, see *webMethods BPM Rules Development Help*.

About Event Rules

An event rule is a decision entity. It specifies the results triggered by an event that occurs during rule execution. If you use an event rule in a rule task, there must be a JMS trigger on the corresponding Integration Server.

For more information about working with event rules, see *webMethods BPM Rules Development Help*.

About Rule Actions

Rules can call a service and run that service: a service action. They can call a process and tell it to invoke (start) itself, or, if it is already running, to suspend itself/cancel itself/fail itself, OR, if it is suspended, to resume itself. Rules can also call a user task within a process in order that a human can make a decision as to the way in which the rule is ultimately applied. This is a unique process action known as a manual decision, sometimes referred to as a task action.

Rule actions are constructed in the Rules Explorer view and can then be dragged from there into decision tables, which can then be used in rules.

For more information about service actions, see *webMethods BPM Rules Development Help*. For more information about process actions, see [“About Process Actions” on page 102](#).

Important:

In order to use a rule action in a process, you need to configure your **Minimum Logging Level** for the process to **5 - Process and all steps**. Designer uses this data to identify the requirements of the rule action. See [“Configuring a Process” on page 57](#).

About Rule Tasks

Designer uses webMethods Business Rules in rule tasks. What you do with your rule is up to you.

Using webMethods Business Rules in Processes

By default, business rules are executed on the same Integration Server as the Process Engine, calling the Rules Engine on the same host. Business rules can also be executed on a remote Integration Server, which has a Rules Engine installed. For more information about executing business rules on a remote Integration Server, see *Administering webMethods Process Engine* and the *webMethods BPM Rules Development Help*.

➤ To use a webMethods Business Rule in a process

1. In Designer, create a rule task on the process editor’s canvas.
2. Configure the rule task to use a webMethods Business Rule as described in [“Configuring Rule Tasks” on page 230](#).

Tip:

You can also drag and drop a webMethods Business Rule from the Rules Explorer view in the Rules perspective. For more information about the Rules perspective, see the *webMethods BPM Rules Development Help*.

About Process Actions

A process action is a specialty rule action that affects an entire process. Actions you can apply to a process include the following:

- Start a new process instance
- Join a running process instance
- Resume one or more suspended process instance(s)
- Invoke a user task (requiring a manual (human) decision)

Note:

Manual process decisions are also known as task actions. See [“About Manual Decisions” on page 103](#).

About Manual Decisions

Manual decisions, sometimes referred to as task actions, are a special type of process action that uses a rule to instantiate a task. When the user completes the task, the Task Engine calls the Rules Engine with the result and the original call context.

Note:

Manual decisions have nothing to do with manual tasks.

A decision table that contains a manual decision point (task action) must be configured in as follows:

- When you create the decision table, you must select the **Process Aware** check box.
- All TaskData must be mapped.
- The task's Inputs/Outputs must match the decision table's Inputs/Outputs.

Note:

The decision table contains an extra parameter called ProcessData that is automatically generated when the decision table is created. This is mapped separately to the process, not to the task.

- The decision table's ProcessData must be mapped to the process.

For more information about working with rule actions and decision tables, see *webMethods BPM Rules Development Help*.

Creating a Process Action

➤ **To create a process action**

1. In the Rules Explorer view, right-click and then click **New > Action** in the context menu.
2. On the Process Action Type page, click the type of process action you want the rule to invoke.
3. To complete the process action, refer to the following topics:
 - [“Starting a New Process Instance” on page 103](#)
 - [“Joining a Running Process Instance” on page 104](#)
 - [“Invoking a User Task” on page 105](#)

Starting a New Process Instance

➤ **To create an action to start a new process instance**

1. Select the **Start a new process instance** option as described in [“Starting a New Process Instance” on page 103](#).
2. Click **Next**.
3. On the New Process Action page, select one or more process model names to start a new instance of those models.
4. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.
5. On the Document Type Selection page, select the IS document type to use as input to the process instance. Click **Next**.
6. On the Process Action Default Values page, specify any default data field values you want to include. These values are overridden when data is provided from an associated process. Click **Next**.
7. On the Process Action Return Value page, select a return value check box as required.
8. Click **Finish**.

Joining a Running Process Instance

➤ To create an action to join a running process instance

1. Select the **Join a running process instance** option as described in [“Starting a New Process Instance” on page 103](#).
2. Click **Next**.
3. On the New Process Action page, select one or more process model names to join a running instance of those models.
4. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.
5. On the Document Type Selection page, select the IS document type to use as input when joining the process instance. Click **Next**.
6. On the Process Action Default Values page, specify any default data field values you want to include. These values are overridden when data is provided from an associated process. Click **Next**.
7. On the Process Action Return Value page, select a return value check box as required.

8. Click **Finish**.

Invoking a User Task

> To create an action to start a new user task instance

1. Select the **Manual decision** option as described in [“Starting a New Process Instance” on page 103](#).
2. Click **Next**.
3. On the Select Task page, select the task type you want to start with the action.
4. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.
5. On the Process Action Default Values page, specify any default data field values you want to include. These values are overridden when data is provided from an associated process. Click **Next**.
6. On the Process Action Return Value page, select a return value check box as required.
7. Click **Finish**.

Configuring a Remote Integration Server

By default, business rules are executed on the Integration Server the Process Engine is installed on and call the Rules Engine on the same host. If required, business rules can be executed on another, remote Integration Server which has a Rules Engine installed. This remote Integration Server can be configured in two ways:

- On Integration Server Administrator, you can create a remote server alias pointing to the remote Integration Server. The alias must be named **businessrulesruntime**, and you must provide the necessary details as described in *webMethods Integration Server Administrator's Guide, Adding an Alias for a Remote Integration Server*.
- On Microservices Runtime, you can configure the remote Rules Engine using configuration variables templates. For more information on working with these templates, see *Microservices Runtime Guide, Using Configuration Variables Templates with Microservices Runtime*. In the template, you must define the remote Rules Engine as follows:

```
remoteserver.businessrulesruntime.host=someotherhost
remoteserver.businessrulesruntime.port=5678
remoteserver.businessrulesruntime.user=Administrator
remoteserver.businessrulesruntime.password={AES}MbYmPezV4NS3Ta31QMtyUQ\=\=
```


10 Process Engine Processing

■ About Process Execution	109
■ About Transitions	117
■ About Looping	128
■ About Process Logic	130
■ About Joins	131
■ About Process and Step Timeouts	138
■ Subprocess Concepts	145
■ Call Activity Concepts	147
■ Handling Process and Step Errors	156
■ Process Cancellation	159
■ Process Suspension	160
■ Process Debugging	160
■ About Document Correlation	161
■ Creating Correlation Services	163
■ Correlation Service Input and Output Variables	163
■ Specifying Correlations	165
■ About Document Merging	166
■ About Process Tracking	166
■ Process Logging Behavior	169

■ About Expression Operators 173

About Process Execution

This section explains how a process is triggered and how the servers run the process steps.

Important:

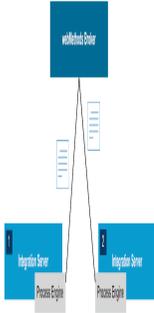
The explanation assumes that the default quality of service settings for the Process Engine are applied; the default settings are designed for best performance. For more information about the default quality of service settings for processes, see [“Setting Quality of Service for a Process” on page 111](#). For more information about tuning the Process Engine, see "Overview of Tuning" in *Administering webMethods Process Engine*.

1. At startup:
 - a. When Universal Messaging or Broker (deprecated) is present, all Integration Servers synchronize document types with the message provider at startup, ensuring that all subscribed document types exist on the message provider as well.
 - b. Each Integration Server reads all packages that were generated by Designer and loads its process description file into memory.
2. An external application publishes a document to which the process trigger subscribes. This document is placed on the trigger's client queue.
3. The trigger on the server that runs the start step retrieves the document and passes it to the Process Engine service that handles external input documents. The Process Engine reads the Process Audit Log database component to find out if the process model is enabled and what the current model version is.
 - a. If the process is not enabled, the Process Engine service exits and the trigger sends an acknowledgment for the external document and discards the document without starting a process.
 - b. If the process is enabled, the Process Engine loads the document content into server memory, and the server runs the start step.
4. Step execution is handled as follows:
 - a. If step 2 is designed to run on the same server as step 1, the server passes the pipeline from step 1 to step 2 and runs step 2.



- b. If step 2 is designed to run on a different server than step 1, step 1 produces a process transition document:

- a. The Process Engine publishes the process transition document.
- b. The Process Engine for the server that runs step 2 loads the document content into server memory and runs step 2. If webMethods Broker (deprecated) is installed, the data flow would be:



- c. If step 2 is the start step for a referenced process, step 1 produces a referenced process start document instead of a process transition document.
 - d. The Process Engine publishes the referenced process start document. The trigger on the server that runs the referenced process start step retrieves the document and passes it to the Process Engine. If the Process Engine is configured to optimize referenced process execution, the referenced process is invoked directly rather than by a publishing a referenced process start document.
 - e. The Process Engine loads the document content into server memory. The server runs the referenced process start step.
5. The process continues to execute steps in this manner until the last step is complete.

For steps with AND joins or COMPLEX joins containing AND logic, the server waits until the Process Engine has received all required external documents, process transition documents, and referenced process documents to satisfy the join, and then runs the step.

Referenced processes use a referenced process done document (failure) or a referenced process return document (success), as appropriate, to transition back to the next process step. The trigger on the server that runs that next process step retrieves the document and passes it to the Process Engine. If the referenced process completed successfully, the Process Engine loads the document content into server memory so the server can run the new process step.

If the entire process runs on the same server and does not branch into multiple execution paths or contain referenced processes that are not optimized for local execution, when the process is complete, the trigger sends an acknowledgment for the external document that was the input to the start step.

Process Execution Table

The following table lists the Process Engine database component tables in which the Process Engine stores process execution data, assuming you are not storing these data in server memory.

Table	Contents
PRTINSTANCEITER	Each time a new iteration of a process starts running, the Process Engine writes a row containing the process iteration count.
PRTPROCESS	For each process, the Process Engine writes a row containing the process status and step pipeline. The Process Engine updates the process status after a step or webMethods Monitor user changes the status, and updates the pipeline after each step.
PRTQUEUECONTENT	After each step that produces a process transition document or referenced process document, the Process Engine writes a row containing the contents of the document, minus the pipeline.
PRTSTEPSTATE	For each step, the Process Engine writes a row containing the step status and the external documents, process transition documents, or referenced process documents the Process Engine has received so far, excluding the pipelines.
PRTSTEPSUSPENSION	For each step a webMethods Monitor user suspends, the Process Engine writes a row containing the step ID.
PRTTIMERENTRY	For each timeout configured for a process, the Process Engine writes a row containing the date and time indicating when the affected step or process will time out.
	<p>Note: When host or port is changed, the SERVERID column in the PRTTIMERENTRY table must be updated manually.</p>
PRTUIDPIDXREF	This table provides persistent storage of document identifiers that Process Engine uses to detect duplicate published input or transition documents. The key for this table is the document UUID. The Process Engine inserts entries into this table when it receives an input or transition document and the user has enabled the duplicate event detection feature.
WMPRTSTORE	After each step, the Process Engine updates a row containing the process thread count.
WMPRTXREF	For each process that uses a correlation ID, the Process Engine writes a row containing the correlation ID. If the process also uses a Trading Networks conversation ID, the row also contains that ID.

Setting Quality of Service for a Process

You can define quality of service settings when you design a process. These settings are made in Software AG Designer and determine how a process executes at run time, enabling you to select a balance of performance, reliability, visibility, and control.

Note:

You must be in Process Developer mode to view and work with this setting. For more information, see [“About Process Development Modes” on page 52.](#)

➤ **To specify the quality of service settings for a process in Designer**

1. In the Process Developer perspective, open the process you want to work with.
2. In the Properties view, click **Run Time**.
3. Configure the quality of service settings as described in the tables below:

Quality of Service Description Setting

- | | |
|-------------------------|--|
| Optimize Locally | <p>Execute adjacent steps on the same Integration Server without publishing transition documents. Enabled by default.</p> <ul style="list-style-type: none"> ■ Select Optimize Locally when you want to use a pipeline to pass data from step to step on the local server, and publish a process transition document <i>only</i> when there is a transition to a step running on another server, or if a process splits into more than one branch. ■ Clear this check box to always publish a process transition document when transitioning to any step, no matter where it is located. No pipeline is used. |
|-------------------------|--|

You can select **Optimize Locally** to decrease document message traffic and improve performance. However, if a step fails, the process can recover automatically only at the step that published the most recent process transition document, and that step might not be the step of failure. For example:

Suppose process step 1 runs on Server A and process steps 2, 3, and 4 run on Server B. When you are optimizing locally and step 3 fails, the most recently published process transition document was produced by step 1, because the Process Engine did not publish a process transition document for step 2 or 3. The process, therefore, recovers automatically at step 1 (that is, step 2 will be run again).

When you do not optimize locally, every step publishes a process transition document, so a process can automatically recover at the step it completed last. In this case, the process recovers at step 3.

The biggest risk of optimizing locally is duplication of work. For example, you might not want to risk duplicating work for processes that store, synchronize, or correlate data. For processes that do less critical work, the performance benefits might outweigh the risks.

Quality of Service Description Setting

When a referenced process is invoked and **Optimize Locally** is selected, the Process Engine will attempt to locally invoke a referenced process, with the following conditions applied:

- The referenced process exists on the same Process Engine node as the parent step.
- The referenced process has no subscription filter.
- If the Integration Server thread usage is below the threshold, communication is done with a direct service invocation on a new Integration Server thread. Otherwise, communication is through the publishing of a document that is handled on the appropriate model trigger.

Note:

The above thread behavior applies to all types of child invocations, including static and dynamic reference processes, and static and dynamic callable processes.

Subscription filters are enforced at the trigger level. If there is a filter on a referenced process, the Process Engine will ignore the **Optimize Locally** setting and publish the transition document.

When **Optimize Locally** is selected and data is returned from the referenced process to the parent, the parent step must be running on the same Process Engine node as the referenced process for successful data transfer.

Express Pipeline Send a reduced data set between the steps in a process. Enabled by default.

Using the express pipeline can significantly improve performance when pipelines are large (1 MB or more).

This setting applies to both the pipeline and transition document methods of transferring data and is independent of the **Optimize Locally** setting.

- Select **Express Pipeline** to specify that you want to pass a reduced (*express*) data set from step to step.
- Clear this check box to specify that the complete data set is passed from step to step.

Note:

When a process is resubmitted, the complete data set is passed, regardless of this setting.

When you use the complete data set, the server passes all data from step to step, regardless of whether outputs are used by downstream steps.

With the Express Pipeline option enabled, the server reads the list of inputs in the process description file and passes a reduced data set that contains *only those outputs explicitly specified in the process model version as inputs to following steps*. All other data is discarded.

Do *not* use the Express Pipeline setting for:

- Processes that include steps with services that add values to the pipeline data; the server will not include the added values because they are not explicitly specified as inputs to downstream steps.
- Processes that contain a process-wide error handler step; the process-wide error handler step is not recognized as a downstream step and will therefore not receive the necessary input data.

Important: Designer *never* implements Express Pipeline for dynamic referenced process/call activity steps, regardless of the **Express Pipeline** option setting.

The purpose of Express Pipeline is to protect explicitly-defined pipe elements, such as step inputs and outputs, from being removed at run-time. Early in your development cycle, however, you may not yet have added any inputs/outputs to the steps in your model, and thus your model is incomplete and not thoroughly designed. In such cases, when you generate (build and upload) your model, then the Express Pipeline setting displayed on the WmPRT home page will not match the value of the Express Pipeline check box as set in Designer.

Important:

When the design is not yet complete, that is you have not yet added any inputs/outputs to the steps, then Express Pipeline in the WmPRT home page will always be displayed as "No". However, during runtime Process Engine will always correctly use the Express Pipeline value as set in Designer.

Volatile Transition Documents Send process transition information in volatile mode. Enabled by default. This setting applies to Universal Messaging and webMethods Broker (deprecated) transition documents and referenced process start documents, and also to JMS transition and referenced process start messages.

- Select **Volatile Transition Documents** to specify the following:
 - **For Subscription (Publishable Documents) protocol:** Process transition documents and referenced process start documents are stored in memory.

- **For JMS (Triggered Processes protocol):** Process transition messages and referenced process start messages are delivered to the JMS `deliveryMode` interface as `NON-PERSISTENT`.
- Clear this check box to specify the following:
 - **For Subscription (Publishable Documents) protocol:** Process transition documents and referenced process start documents are stored on the local hard disk drive.
 - **For JMS (Triggered Processes protocol):** Process transition messages and referenced process start messages are delivered to the JMS `deliveryMode` interface as `PERSISTENT`.

Enabling **Volatile Transition Documents** can significantly improve performance when documents are large (2 MB or larger). However, if the Universal Messaging, webMethods Broker (deprecated) or JMS server fails while a step is running, the process cannot automatically recover and completion cannot be guaranteed, and the document or message will be lost. If you are logging step pipelines to the Process Audit Log database component, you can manually recover the process by resubmitting steps through webMethods Monitor.

About Message Provider Behavior:

When the message provider receives a process transition document or referenced process document, it places the document in the process trigger's client queue and also stores it either in memory or on disk.

When the trigger retrieves the document, if the document was stored in memory it is immediately acknowledged and deleted from the message provider. If the document was stored on disk on the message provider, the document is acknowledged and deleted by the message provider after the process has published the next process transition document or the process completes.

Volatile Tracking Store process tracking information in memory only. Disabled by default.

Note:

If the Process Engine is running in a clustered environment, volatile tracking cannot be used, and this setting is ignored.

- Select **Volatile Tracking** to specify that the Process Engine stores process status in memory.
- Clear this check box to specify that the Process Engine stores process status in the Process Engine database component.

The Process Engine stores process status while a step that requires it is running. Process status is comprised of content from:

- External documents and process transition documents
- Referenced process documents
- Process and step status
- Process iteration count and correlation IDs
- Step and process timeouts

Using volatile tracking can significantly improve performance. However, if you use volatile tracking and a server fails while running a step, process status will be lost.

If you are logging process step status to the Process Audit Log database component, the step iteration count will be inaccurate in webMethods Monitor, making it harder to address the negative effects of server failure and to determine how much work has been duplicated.

If you choose to store process status in the Process Engine database component, you must configure the database component. For instructions, see "Configuring and Monitoring the Process Engine" in the PDF publication *Administering webMethods Process Engine*. For more information about the Process Engine database component, see *Installing Software AG Products*.

Minimum Logging Level Sets the minimum audit logging threshold for this process at run time. Set to **5 - Process and all events, activities, and looped activities** by default.

At generation time, the **Minimum Logging Level** is set in webMethods Monitor based on this value. On subsequent generations, if the **Minimum Logging Level** is increased in Designer, the level in Monitor is also increased. If the **Minimum Logging Level** in Designer is lowered in subsequent generations, the level in Monitor is not lowered. You must explicitly lower the audit logging level in Monitor.

If a user attempts to set a new process audit logging level in Monitor, the user will not be able to specify a logging level that is numerically lower than the value you specify here. For example, if you specify a level of **2-Errors Only** here, the user will not be able to specify a logging level of **1-None** in webMethods Monitor; the user's choices are limited to audit logging levels 2, 3, 4, and 5.

If you are sending process data to webMethods Optimize to run historical data fit distributions in Process Simulation, you must set the minimum audit logging level to **5-Process and all events, activities, and looped activities**. If you need only process-level logging and step errors to be sent to Optimize, then logging level 3 is sufficient.

- For more information about process audit logging, including descriptions of logging levels, see the PDF publication *webMethods Monitor User's Guide*.

Select one of the following values from the drop-down list:

1-None

2-Errors only

3-Process only

4-Process and start events

5-Process and all events, activities, and looped activities

Note:

Logging level **6 - Process and all events, activities, and looped activities** was available in version 8.2 but has been removed in later versions and its functionality moved into logging level 5.

Parallel Execution (Step Locking)

Parallel execution, also known as step locking, allows the step to be executed by multiple threads. Parallel execution can be applied to activities (tasks, call activities (including BPMN callable activities and webMethods referenced processes, and subprocesses) and gateways. The **Allow parallel execution** check box is on the **Implementation** page in the Properties view.

Note:

You cannot apply parallel execution to events.

About Transitions

Transition lines between steps serve two purposes:

- As graphical objects in a process model, they provide a visual indicator of the flow of work and data through the process. A transition line can be modified to change its route, color, line format, and the font characteristics of the associated text. For more information, see [“Configuring Custom Transition Line Appearance” on page 126](#).
- As process objects, they provide behavior control over the flow of work and data. For more information about the behavior of transitions, see [“About Transition Type Behavior” on page 118](#).

In BPMN 2.0, transitions are referred to as *sequence flows* (inbound or outbound). Throughout this documentation, the term *transition* is used, and unless noted otherwise, information about transitions also applies to sequence flows. To create a transition:

- You add a transition to a process as described in [“Creating a Transition” on page 124](#).

- You configure the transition behavior to your requirements as described in [“Configuring Transition Behavior” on page 125](#).

In addition to transitioning to a following step in the process or subprocess, a transition can also connect to an earlier step in the process, or even back to the input of the same step. This is referred to as *transition looping* (also referred to as *sequence flow looping* in BPMN). For more information, see [“About Looping” on page 128](#).

Other transition line actions include:

- You can reposition the transition label of a transition line. If the line is relocated for any reason, the label returns to its default position on the center of the line.
- You can hover the cursor over a transition to view a summary of transition properties. A default transition (taken when all others evaluate to false) is depicted by a transition line with a backslash symbol across it.
- You can set default preferences for the transition line type used in all process models (straight, curved, or custom), as well as label text priority. For more information, see [“Configuring Appearance Preferences” on page 36](#).
- You can set default preferences for the transition label used in all process models. For more information, see [“Configuring Appearance Preferences” on page 36](#).
- You can change the appearance of an individual transition line. For more information, see [“Configuring Custom Transition Line Appearance” on page 126](#).

About Transition Type Behavior

A transition connects one step to another within a process or a subprocess. The transition behavior described in these topics applies to both steps and subprocesses. For more information about creating and configuring transitions, see [“Creating a Transition” on page 124](#) and [“Configuring Transition Behavior” on page 125](#).

A transition routes information from one step to another within a process model. You can define the logical behavior of the transition by configuring it as one of the available transition types (some steps do not support all transition types). For information about the available transition types, see [“About Transition Types” on page 118](#).

You can create more than one transition for a step or subprocess; for example, you can configure a step with an If Condition transition, a Default transition, and a Step Iterations Exceeded transition, each connecting to a different step or subprocess, to account for different step outcomes. For more information about configuring step transitions, see [“Configuring Transition Behavior” on page 125](#).

Important:

If you import a process model created with Designer version 8.1 or earlier, timeout transitions are replaced with BPMN 2.0 interrupting or non-interrupting boundary timer events. Error transitions are also no longer available in version 8.2 or later and are replaced with BPMN 2.0 boundary intermediate error events when you import process models from earlier versions.

About Transition Types

You can create step transitions between any two steps in the process editor, including within subprocesses.

Note:

Transitions using fields of type String that contain numerical values at run time are compared numerically.

Designer supports the described in the following table step transition types (some steps do not support all transition types):

Designer Transition Type	Description
If Condition	Use this transition type to specify a requirement condition for the transition. This transition compares a data field in the process pipeline to a specified value, using a logic operator (for example, = or >). The transition is taken when the IF condition is matched. For more information, see “About Expression Operators” on page 173 and “About Transition Looping” on page 129 .
Default	This transition type supports no conditional behavior and is always taken when it is the only transition out of a step. If other conditional transitions exist, this transition is taken if none of the other step transition conditions are matched. Legacy Designer Else transitions migrate to Default transitions.
Join Timeout	This transition is configured on the Joins page in the Properties view of a step. The Joins page is available only when a step has two or more input transitions. Join timeout transitions are supported for AND and COMPLEX join types only. This transition is taken when a specified join timeout period is exceeded. The join timeout can be based on: <ul style="list-style-type: none"> ■ A static value that you specify. ■ A value derived from a field that is found in the incoming pipeline. ■ A value derived from business calendar in My webMethods Server. The outgoing transition label displays the text Join wait time > xxx . For process models imported from Designer versions earlier than version 8.2, timeout transitions configured with a timeout migrate to a join timeout transition.
Step Iterations Exceeded	This transition is taken when a step is invoked more times than the limit specified by the Maximum Iterations for <Step Name> step property. You use this type of transition to terminate a loop created with transition looping. When activated, the Step Iterations Exceeded transition will

Designer Transition Type	Description
	override all other transitions (whether they are true or not). For more information, see “About Transition Looping” on page 129 .
Unsatisfied Join	This transition is taken when the corresponding has a join expression and that expression has failed at run time (that is, the join conditions of the step are not satisfied). As of version 9.7, an unsatisfied join is not escalated unless it has an unsatisfied join handler.

About Step Transition Behavior

Process model transitions are executed in the Process Engine. The Process Engine carries out step transitions in a defined order based on the design of the model. Generally:

- Step-level exceptions (such as errors, timeouts, and maximum iteration counts) are first. For more information about step exceptions, see [“About Transition Exceptions” on page 122](#).
- These are followed by transitions, both with and without conditions.
- If the Process Engine does not encounter any of these, it moves on to the process-level timeout.

In the course of executing the steps in a process model, the Process Engine employs a set of guidelines known as *Standard Exception Handling*. The Process Engine follows the Standard Exception Handling order when it encounters no other specific guidance at the step level. For more information, see [“About Transitions and Standard Exception Handling” on page 123](#).

The Process Engine executes step transitions as follows.

The following table explains what happens before the step is executed:

If	The Process Engine
There is a join expression on a step and that expression is not satisfied at run time.	Takes the Unsatisfied Join transition, if defined. When an Unsatisfied Join transition is not defined, the process executes Standard Exception Handling. See “About Transitions and Standard Exception Handling” on page 123 .
The step has exceeded its Step Retry Count .	Takes the Step Iterations Exceeded transition. When a Step Iterations Exceeded transition is not defined, the process Executes Standard Exception Handling. See “About Transitions and Standard Exception Handling” on page 123 .

The following table explains what happens while the step is executing:

If	The Process Engine
The step encounters an error.	Executes Standard Exception Handling. See “About Transitions and Standard Exception Handling” on page 123.
The step has an intermediate boundary interrupting timer event, and its Timer Condition is met before the step completes.	Takes the step’s Timer Condition transition. See “Configuring a Boundary Intermediate Event” on page 290.

The following table explains what happens after the step has executed:

If	The Process Engine
There are transitions without conditions.	Takes them.
There are "if" conditions that are satisfied.	Takes them.
There are "if" conditions that are not satisfied <i>and</i> there is an "else" transition defined.	Takes the "else" transition.

The following table explains what happens at any point:

If	The Process Engine
A join has a timeout value specified <i>and</i> one of the required paths never reaches the step.	Takes the first of the following transition types it encounters: <ol style="list-style-type: none"> 1. Step timeout transition. 2. Process-level timeout. 3. Standard Exception Handling. See “About Transitions and Standard Exception Handling” on page 123.
A process-level timeout is defined <i>and</i> it expires at any point during the process.	Executes the process-level Timeout Handler Step .
No process-level Timeout Handler Step is defined.	Executes the process-level Error Handler Step .

If**The Process Engine**

No process-level **Error Handler Step** is defined. Sets the process instance to **Failed**.

About Transition Exceptions

The following transition types can generate exceptions:

- Join Timeout transition
- Step Iterations Exceeded transition
- Unsatisfied Join transition
- A transition from a Boundary Timer Event

Any transition that generates an exception contains a top-level document named `ExceptionTransition`. This document can contain up to four fields:

- **ExceptionType**. Contains one of the following values indicating the type of exception that caused the error: `StepTimeout`, `JoinTimeout`, `RetriesExceeded`, or `UnsatisfiedJoin`. `ProcessTimeout`, `StepError`, and `Cancel` values are also supported for those exceptions.
- **SourceStep**. Contains the step ID of the step if there is one. No value is provided for exceptions of type `ProcessTimeout` or `Cancel`.
- **SourceStepIteration**. The step iteration count of the step that encountered the error.
- **ErrorMessage**. Contains the error message when the exception type is `StepError`.

The information in this document enables you to create specialized logic for handling step and process errors. For more information, see “[pub.prt.ExceptionTransitionInfo](#)” on page 193.

Process timeouts and process cancellations also generate an exception specific to each condition. Although these transitions are not represented by a graphical transition line in a process model, the exceptions they generate are passed to the process timeout handler and process cancel handler steps, respectively.

About Importing Models with Subprocesses with Compensating Error Transitions

The 8.0 release of Designer provided the ability to create *compensating error transitions*, which enabled the process designer to draw an error transition from a `webMethods` subprocess. This transition featured a **Compensating** check box on the Properties panel of the `webMethods` subprocess. When this **Compensating** check box was selected and an error occurred in the subprocess, the transition passed the pipeline *as it existed at the start of the step that failed*, enabling corrective (or compensating) action to be applied.

In version 8.2, the compensating error transition was replaced by the BPMN intermediate boundary error event. When you import a process model created with a version of Designer prior to 8.2, any subprocesses that model that have a compensating error transition are converted to a `webMethods` subprocess with an intermediate boundary error event.

In addition, backward compatibility is provided for these imported compensating error transitions by offering a **Restore Starting Pipeline** check box for an intermediate boundary error event on a webMethods subprocess. The **Restore Starting Pipeline** check box provides the same behavior as the **Compensating** check box in the older version. That is, when this check box is selected, Designer saves the input pipeline upon entering the subprocess. If a subprocess error occurs, the boundary error event passes the contents of the starting pipeline through its transition to a subsequent step.

Note:

The **Restore Starting Pipeline** check box is available *only* on boundary error events placed on a webMethods subprocess, which is now deprecated. The check box is not available on a BPMN subprocess.

Process models created with version 8.2 and later still provide a **Compensating** check box on various activity types. However, this check box *has no behavior impact on the model*, and serves only to apply the BPMN 2.0 icon for a compensating step to the selected activity. This is purely notational.

For more information about the deprecated webMethods subprocess and BPMN subprocesses, see [“About Subprocesses” on page 242](#).

About Transitions and Standard Exception Handling

When executing steps in a process model, the Process Engine handles exceptions based on settings in the process model and its components. When doing so, it follows a standard set of guidelines known as *Standard Exception Handling*. This is sometimes referred to as “default” exception handling.

In Standard Exception Handling, the Process Engine selects a transition to follow based on these conditions, as described in the table below:

If	The Process Engine
The step has a boundary intermediate error event.	Takes the boundary intermediate error event transition.
The step is a subprocess.	Takes the subprocess error transition.
The step is a subprocess	Takes the parent subprocess error transition.
AND	
The subprocess is nested inside another subprocess.	
An Error Handler Step is defined.	Executes the Error Handler Step .
No Error Handler Step is defined.	Sets the instance to Failed .

About Transition Line Shape

You can specify the shape of a transition line as straight, curved, or custom. By default, transition lines are created with a straight shape.

Note:

The behavior and properties described here for transition lines also apply to annotation link lines. See [“About Notes and Annotations” on page 368](#) for more information.

You can specify whether Designer draws curved, straight, or custom transition lines by default. For more information, see [“Configuring Appearance Preferences” on page 36](#). You can also change the line type for an individual transition line or for an entire process. For more information, see [“Configuring Custom Transition Line Appearance” on page 126](#).

The following transition line shapes are available:

- **Straight.** These transition lines feature horizontal or vertical line segments only, and 90-degree corners for all changes of direction. When selected, these lines also feature movement handles, enabling you to move line segments horizontally and vertically.
- **Curved.** These transition lines feature no sharp corners and define a smooth and continuous path between steps. They cannot be re-routed by moving.
- **Custom.** These transition lines feature straight line segments, divided by *inflection points*. You can click and drag a point on the line to move the point. When you drop the inflection point, new points are created on the line segments surrounding it, enabling you to create as many segments as you need. This is useful for routing transition lines around other objects on the process editor's canvas.

Creating a Transition

> To create a transition

1. In Designer, open the process you want to work with.
2. In the process editor, do one of the following:
 - Hover the cursor over the step you want the transition to start from. When the speed buttons appear, click the transition speed button .
 - Right-click the step or subprocess you want the transition to start from, and then click **Add Transition**.
 - Click **Transition** in the palette and then click near an attachment point in the step or subprocess you want the transition to start from.
3. Move the cursor to draw a line to the step you want to connect to. Place the cursor over an attachment point on the target step, and when the step or subprocess is highlighted, click to attach the transition.

Each step and subprocess initially has four basic points for attaching a transition line, and you can attach the transition line to any of them. For activity steps, after the first attachment, you can select the source or target end of the transition line and reattach it anywhere on the outside of an activity step or subprocess.

Event steps and gateways steps retain the four attachment points at all times. You cannot attach a transition line to any other point on an event or gateway step.

Note:

If the destination is not supported (for example, if it would create an illegal loop) you will not be allowed to make the connection. In this case, start with step 2 again.

Tip:

To redraw the transition line to take the shortest path between the source and target steps, right-click the transition line and click **Reset Connection Path**.

You can now configure the transition behavior in the **Transitions** page in the Properties view of the step, or on the **Condition** page in the Properties view of the transition, as described in [“Configuring Transition Behavior” on page 125](#).

4. Save the process.

Configuring Transition Behavior

This topic describes how to define the *behavior* of the transition by defining the transition type and conditions (if available).

If you want to change the *appearance* of the transition, such as configuring the line color, and the label text font, size, and color, see [“Configuring Custom Transition Line Appearance” on page 126](#).

➤ To configure transition behavior

1. In the process editor, do one of the following:
 - Select a step with an output transition, and then select the **Transitions** page in the Properties view. If the step has multiple transitions, you may need to scroll down to find the transition you want to work with.
 - Right-click the transition line, click **Show Properties**, and then click the **Condition** page in the Properties view.

Note:

Some steps do not support certain transition types. A warning message appears if you select a transition type that is not supported by the step.

2. In the **Transition Type** list, select the type of transition you want. For more information about the available types, see [“About Transition Types” on page 118](#).

- Optionally, type descriptive information about the selected condition in the **Description** field. This text appears in the process canvas depending on the **Transition Text** setting in the Appearance preferences. For more information, see [“Configuring Appearance Preferences” on page 36](#).
- Configure the transition as follows :

Note:

For additional information about these choices, see [“About Transition Types” on page 118](#).

The following table explains what to do to configure the transition.

For this type	Do this
Default	No further behavior configuration is possible.
Unsatisfied Join	No further behavior configuration is possible.
If Condition	<p>Click  Add to add information that defines the If Condition. You can add multiple condition lines by specifying the AND/OR operator and clicking Add again.</p> <ul style="list-style-type: none"> ■ Field Name. Click this table cell and select an available document field from the list. The If Condition is based on the value of this field. ■ Operator. Click this table cell and select an operator to use to compare the Field Name value and the Comparison Value/Field value. See “About Expression Operators” on page 173 for more information. ■ Comparison Value/Field. Click this table cell and select an available document from the list and select a comparison field in that document, or type a value that you want to compare with the value selected for Field Name. ■ AND/OR. Use the AND and OR operators to define the logical behavior if you specify multiple parameters for the condition statement.
Step Iterations Exceeded	Specify the iteration maximum as an integer value.
Join Timeout	You can select this transition type only for steps with AND and COMPLEX joins. Set the join timeout on the Joins page of the step’s Properties view.

- Save the process.

Configuring Custom Transition Line Appearance

Note:

Default transition line appearance settings for all process models are applied by Designer preferences. For more information, see [“Configuring Appearance Preferences” on page 36](#).

You can configure custom transition line appearance for a single transition, or for all the transitions in the process, by configuring the line color, and the label text font, size, and color. You can also reset the connection path to its shortest routing, and whether the line type is straight, curved, or custom.

For more information about line shapes, see [“About Transition Line Shape” on page 124](#).

➤ To customize transition line appearance

1. In Designer, open the process you want to work with.
2. In the process editor, you can modify individual transition lines or all transition lines in the process.

- To customize an individual transition line:

Note:

The following actions for individual transition lines also apply to annotation link lines.

Right-click the transition line and:

- Click **Change Line Shape** to change the transition line characteristics (straight, curved, or custom).
- Click **Reset Connection Path** to redraw the transition line path. This action determines the closest terminal points between the source and target steps and then draws the shortest connection path between them.
- Click **Show Properties**, and on the **Appearance** page in the Properties view, modify the listed in the following table appearance settings as required:

Field	Description
Font Style	Font name, size, variation (bold or italic), and color.
Line Color	Transition line color.
Connection Path	Click Reset to redraw the transition line path or paths. This action determines the closest terminal points between the source and target steps and then draws the shortest connection path between them.

- Straight-line transitions are divided by *inflection points*. You can click and drag a point on the line to move the point. When you drop the inflection point, new points are created on the line segments surrounding it, enabling you to create as many segments as you need. This is useful for routing transition lines around other objects on the process editor's canvas.

- You can reposition the transition label of a transition line. If the line is relocated for any reason, the label returns to its default position on the center of the line.
- To customize all transition lines in the process:

Note:

Any changes applied to all transition lines at the process level will also be applied to annotation link lines. See [“About Notes and Annotations” on page 368](#) for more information.

Right-click the process canvas and:

- Click **Change All Line Shapes** to change the transition line shape (straight, curved, or custom) for all transition lines in the process model.
- Click **Reset All Connection Paths** to redraw all transition line paths in the process model. This action determines the closest terminal points between the source and target steps and then draws the shortest connection path between them.

3. Save the process.

Removing a Transition

> To remove a transition

1. In the process editor, click the transition line you want to remove to select it.
2. Press the **Delete** key on the keyboard, or right-click the selected line and click **Delete**.
3. Save the process.

About Looping

Looping is a common implementation in business process models. The following types of looping are supported:

- **Standard looping.** In this case, looping is applied to a single subprocess step or call activity step. To implement standard looping, you define a loop expression for the step on the **Loop** page in the Properties view for the step. As long as the conditions defined in this loop expression are met, the step will loop back to its input upon completion. For more information, see [“About Standard Looping” on page 129](#).
- **Transition looping.** In BPMN, this is referred to as *sequence flow looping*. In this case, you define a transition from any step’s output to the input of a step earlier in the process flow, or back to the input of the step itself. For more information, see [“About Transition Looping” on page 129](#).

Looping behavior differs somewhat between webMethods subprocess and BPMN subprocesses. For more information, see [“About Subprocess Types” on page 146](#) and [“About Changing webMethods Subprocesses to BPMN Subprocesses” on page 244](#).

About Standard Looping

When a subprocess or call activity is configured for standard looping, Software AG Designer marks the step with a loop marker .

To implement standard looping, you define a loop expression on the **Loop** page in the Properties view for the step. As long as the conditions defined in this loop expression are true, the step will loop back to its input upon completion. The first loop iteration receives the incoming pipeline, and after that, the output of that loop becomes the input to the next loop iteration. For more information, see:

- [“Configuring a Subprocess Standard Loop” on page 246](#)
- [“Configuring a Call Activity Standard Loop” on page 254](#)

For subprocesses and call activities created before Designer version 8.2, the loop expression is tested *after completion of the activity*. With Designer version 8.2 and later, you can configure this behavior to occur either *before the activity* or *after the activity*.

With Designer version 8.2 and later, when you configure standard looping behavior, you can also specify a **Maximum loop count**. This enables you to specify an absolute maximum number of loop iterations, regardless of the evaluation of the loop expression. You can set this number as a static integer value, or as a pipeline field value.

You can monitor loop data stored in the Process Engine. With Designer version 8.2, a loop logging level is available in the process audit logging **Minimum logging level** property (6-Process and all events, activities, and looped activities). In Designer version 9.0, level 6 is eliminated and logging is included in level 5 (5-Process and all events, activities, and looped activities).

After each loop is executed, the value of the `LoopCounter` parameter in the document `pub:prt:ProcessData` is incremented. To create a complex loop expression that makes use of the `LoopCounter` parameter, you can map the parameter into the generated service for the step that is being looped.

When you are looping over a subprocess or call activity step, the `LoopCounter` parameter value is accessed with the input data mapping and output data mapping services within the process package in Integration Server. If these services are not modified, the `LoopCounter` value will remain at zero while the steps inside the subprocess or call activity step execute another iteration.

About Transition Looping

Transition looping is also known as *sequence flow looping* in BPMN. Unlike standard looping, which enables you to loop through a single subprocess or call activity step for a specified number of iterations, a transition loop enables you to loop back to any previous step in the process from any other step in the process. You can, of course, also loop back to the input of the originating step, but for this case, a standard loop is recommended.

For transition looping, the loop conditions are defined in the step's output transition by selecting the transition and then, on the **Condition** page in the Properties view for the transition. For more information, see [“Configuring Transition Behavior” on page 125](#).

You can define a transition loop condition with either of the following transition types:

- Use an **If Condition** transition to implement a transition loop that will continue to loop back to an earlier point in the process as long as the condition expression evaluates to true. In this case, you could create a pipeline field that you increment by 1 each time the loop is taken. You could then define your If Condition to stop looping when the desired number of iterations is reached, at which time a Default transition is taken.
- Use a **Step Iterations Exceeded** transition to implement a transition loop that will continue to loop back to an earlier point in the process until the maximum iteration value is exceeded. In this case, when the step count exceeds the defined value, another transition can be taken. If you attempt to implement this method, be sure to examine your process logic to ensure that the step count will actually increment as required. For example, if the step is within a standard looping subprocess, the step count is reset to 1 for each loop iteration of the subprocess.

Either way, the key point is that this looping mechanism can loop through a flow of multiple steps, depending on where you place the transition destination.

About Process Logic

The Process Engine provides a wide range of logic to support step-to-step processing. This logic offers a flexible framework for dealing with the conditions that can occur during process run time, such as:

- Error conditions
- Timeout conditions
- If and Else condition handling
- Excessive step iterations
- Unsatisfied join conditions

With this logic, you can use the Process Development perspective in Software AG Designer to create process models that enable one or more outputs from a process step, depending on the outcome of the step processing. These step outputs, or *transitions*, enable you to create multiple processing paths within the model.

With BPMN 2.0 functionality, you can implement converging/diverging (joining) patterns by using a *gateway step*. For information about working with BPMN gateway steps, see the *webMethods BPM Process Development Help*. The Process Engine supports the following types of *join steps*:

- OR
- AND
- COMPLEX

Process Expression Evaluation Logic

Binary process expressions are used in transition conditions, join expressions, complex trigger filters (subscription filters), and standard looping conditions. Only literal values are allowed in string comparisons.

If an expression has a field reference and that field does not exist in the pipeline, the expression is evaluated as "false".

For example, the expression `%FieldA% > 2Pipeline` does not contain a field called "FieldA". The expression is evaluated as a false value.

Note:

String comparisons must use literal values. Pattern matching (starts with, ends with, contains, and their negations) is not permitted when comparing variables.

About Joins

In a business process model, you can merge, or *join*, two or more inputs in a single step, and select from available Boolean logic definitions to define the behavior of the join step. In BPMN models, a join is typically accomplished with a gateway step. Software AG Designer supports both step joins and gateway steps. For more information about gateway steps, see [“About Gateways” on page 340](#) and [“About Gateway Types” on page 341](#).

The available join types are:

- **OR.** The step processes the incoming data when any incoming transition is satisfied (that is, provides data). The OR join is available as:
 - A standard OR join, also known as a *synchronized OR*. This is the default behavior for Process Engine version 8.0 and later.
 - An *unsynchronized OR*, provided for compatibility with Process Engine version 7.x.

For more information, see [“About OR Join Behavior \(Synchronized vs. Unsynchronized\)” on page 136](#).

- **AND.** The step processes the incoming data only when all incoming transitions are satisfied (that is, provide data).
- **COMPLEX.** The step processes the incoming data according to the custom join logic you create with a join condition editor.

You can optionally specify a join timeout value. For more information about join timeout behavior, see [“About Process and Step Timeouts” on page 138](#).

About Dead Path Notification and Join Steps

Many business process models contain parallel processing paths, where process data is handled in different paths (also know as tracks) within the process. Eventually, these parallel paths come

together in a gateway or join step. In such cases, one or more of the incoming transitions may not complete because of the failure of an upstream step or some other processing condition.

An incomplete transition of this type is also referred to as a *dead path*, meaning that the required join information will never be provided. As a result, the join step could enter a permanent waiting state, waiting for information that will never arrive.

To avoid this condition, the Process Engine issues an internal *dead path notification* to the downstream join step, behavior introduced in version 8.0. This notification informs the join step that the transition is incomplete and will never be fulfilled, thus preventing the join step from waiting indefinitely.

For more information, see [“How Incomplete Transitions Affect Join Steps and Gateways” on page 132.](#)

How Incomplete Transitions Affect Join Steps and Gateways

A typical business process model merges parallel processing paths with the process into a gateway or join step. On occasion, one or more of the incoming transitions may be unable to complete because of certain processing conditions, such as the failure of a previous step. These incomplete transitions are referred to as *dead paths*. For more information, see [“About Dead Path Notification and Join Steps” on page 131.](#)

When an AND or an OR step has an incomplete transition as an input, the logical behavior of the join is directly affected. For example, an AND join step with two incoming transitions can never be satisfied if one of the two transitions is incomplete.

The Process Engine analyzes each process instance for occurrences of these incomplete transitions, ensuring that all inbound transitions arrive at a gateway or join step, where they are determined to be complete or incomplete. An incomplete transition is considered to be a Boolean false and a complete transition is considered to be a Boolean true for purposes of join evaluation. These result in the following behavior:

The table below lists the inbound transitions and the behaviors.

Inbound Transition A	Inbound Transition B	AND Behavior	OR Behavior
Incomplete	Incomplete	Unsatisfied	Unsatisfied
Incomplete	Complete	Unsatisfied	Satisfied
Complete	Incomplete	Unsatisfied	Satisfied
Complete	Complete	Satisfied	Satisfied

Note:

Unsynchronized OR joins exhibit a unique behavior pattern. For more information, see [“About OR Join Behavior \(Synchronized vs. Unsynchronized\)” on page 136.](#)

If you have process models from an earlier version, be aware of these points:

- In Process Engine version 7.x, incomplete transitions were not evaluated, resulting in the possibility that a join step could wait indefinitely for inputs that would never arrive, effectively stopping the process. As a result, 7.x versions of Process Engine *require a join timeout* for all join steps in order to overcome this issue.
- In Process Engine version 8.0 and later, join timeouts are available *but are not required*. For more information, see [“About Dead Path Notification and Join Steps” on page 131](#).

Note:

The process debugger does not support unsatisfied AND joins.

About Join Failures in Join Steps and Gateway Steps

A join failure occurs in a join step or gateway step when the Process Engine determines that one of the expected inputs to the step is an incomplete transition (or *dead path*) and the join step or gateway cannot be satisfied, as described in [“How Incomplete Transitions Affect Join Steps and Gateways” on page 132](#), above. In this case, you can specify an Unsatisfied Join transition type for the step, providing a processing path to a downstream step that can take additional actions when the a step experiences a join failure.

Note:

When a join step or gateway receives a dead path notification, Process Engine sets the step to satisfied or unsatisfied status, as described in [“How Incomplete Transitions Affect Join Steps and Gateways” on page 132](#). Those steps that are determined to be unsatisfied joins will remain in Waiting status.

Join Timeout when Multiple Timeout Values are Specified

Join timeouts are used to defend against a process that never completes. With a Join timeout, a process that might remain in a "hanging" state is instead executed using an exception path. When the first sequence flow arrives at a Join step, the runtime starts a timer. All other sequence flows to the join have no impact on the timer. The timer is set once and cannot be modified. The documents that arrive after the Join timer is established are queued in the database.

With the timeout in place, the execution continues in one of the following directions:

- The join itself is satisfied and executes the normal pattern.

Once the Join is satisfied, the documents in the queue are moved up to the next position in line for future processing.

- The join timer expires and executes the exception pattern.

The documents that were queued are moved forward in the queue awaiting a new iteration of the Join step. The moving forward of the docs in the queue does not generate a new iteration of the Join, that only occurs when the first sequence flow arrives at the Join step. The timeout is once again set by the value of the first sequence flow that arrives after the timeout.

The Join timeout is set when the very first sequence flow arrives at a new iteration of Join step. It does not use the queued documents to determine the value of the timeout.

Migrating Process Models with Join Steps to Version 9.7 and Later

This section of the documentation applies only to users who are migrating process models with join steps from Process Engine version 8.x through version 9.6 to Process Engine version 9.7 and later.

In version 8.x through version 9.6, join steps featured two optional settings:

- **Suppress join failure**
- **Ignore dead path notification**

In version 9.7, significant advancements were made in Process Engine join handling. Although these optional settings have not been eliminated from the product, the two settings are deprecated and are no longer recommended.

The primary reason for deprecating the use of these options is that they enable a process to be created and implemented in such a way that the behavior of the model in the run time provides misleading or undesired results.

For example, by suppressing a join failure, a process model can actually run to an apparently successful completion, despite the presence of a transition failure within the process. Similarly, ignoring a dead path notification can cause a process instance to appear to be running when it is actually waiting for a transition that will never arrive.

The settings continue to be available on the **Joins** tab in the Properties view of a join step, although they are deprecated and hidden by default under the label **Deprecated properties (Not recommended)**. As the label states, these options are not recommended for use.

For more information, see:

- [“Suppressing a Join Failure \(Deprecated\)” on page 134](#)
- [“Ignoring a Dead Path Notification \(Deprecated\)” on page 135](#)

Suppressing a Join Failure (Deprecated)

Important:

This join option is deprecated and is not recommended for use. In rare instances it might be applicable in certain business cases or for compatibility in models imported from previous versions. For more information, see [“Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134](#).

Steps with OR, AND, or COMPLEX join behavior can be optionally configured to suppress a join failure (join failures are not suppressed by default). When you suppress a join failure, you prevent the step from following a boundary intermediate error event transition (if one exists), thus enabling the process to continue running despite the join failure. In general, this is not good process design practice.

When a join failure is suppressed, the processing path containing the step comes to an end, resulting in an incomplete transition, or dead path, for any downstream steps that expect it to provide an input. Also, if you specify an Unsatisfied Join transition and select the **Suppress Join Failure** check box, the **Suppress Join Failure** option takes precedence, and the Unsatisfied Join transition is not taken.

➤ To suppress a join failure

1. In the process editor, click a step that contains a join to select the step.
2. Click the **Joins** page in the Properties view.
3. Expand the label **Deprecated properties (Not recommended)**.
4. Select the **Suppress Join Failure** check box.
5. Save the process.

Ignoring a Dead Path Notification (Deprecated)

Important:

This join option is deprecated and is not recommended for use. In rare instances it might be applicable in certain business cases or for compatibility in models imported from previous versions. For more information, see [“Migrating Process Models with Join Steps to Version 9.7 and Later”](#) on page 134.

Select this check box to ignore notification that a step providing a join input has failed and that the incoming transition will never be completed (that is, it is a *dead path*). In general, this is not good process design practice.

If you have process models from an earlier version and you regenerate them with Designer version 8.0 and later, you may find that the dead path notification behavior introduced in version 8.0 causes your older process model to behave unexpectedly.

In earlier versions, the join step would continue to wait indefinitely for the failed transition, which required a join timeout value to be set for each join step. For more information, see [“About Dead Path Notification and Join Steps”](#) on page 131, and [“How Incomplete Transitions Affect Join Steps and Gateways”](#) on page 132.

You can configure a join step or gateway in a version 8.0 or later process model to ignore the dead path notification and exhibit the behavior of the earlier versions. In such a case, you must have a join timeout specified for the step. Otherwise, if you apply the **Ignore dead path notification** option without a join timeout, the process will wait indefinitely for an input that will never arrive.

➤ To ignore a dead path notification

1. In the process editor, click a step that contains a join to select the step.

2. Click the **Joins** page in the Properties view.
3. Expand the label **Deprecated properties (Not recommended)**.
4. Select the **Ignore dead path notification** check box.
5. Save the process.

About OR Join Behavior (Synchronized vs. Unsynchronized)

The behavior of OR joins differs between Process Engine version 7.x and Process Engine version 8.0 and later. If you intend to import process models from version 7.x into version 8.0 or later, be aware of the following points:

- In Process Engine version 7.x, the OR join is referred to as an *unsynchronized OR*. In this case, the join is satisfied immediately upon arrival of a valid input. However, if multiple inputs arrive, an output is generated for each of the inputs, resulting in multiple outputs; in other words, the join can be satisfied more than one time.

Note:

The failure of an unsynchronized OR join step for any reason, including an unsatisfied join, always results in a dead path notification. In Process Engine versions 9.6 and earlier, a process-level option was available to enable or suppress dead path notifications for unsynchronized OR join steps. This option is no longer available in versions 9.7 and later. A model imported from earlier versions into version 9.7 and later may behave differently because of this change. In this cases, you must modify your process model to accommodate this change.

- In Process Engine version 8.0 and later, the OR join behaves as a *synchronized OR*. As expected, the OR join is satisfied by one (or more) valid input transitions. However, the synchronized OR waits for *all input transitions to arrive* (valid or dead path) before the join is evaluated. A synchronized OR has the ability to create multiple tracks through a process for every true inbound transition, where this is not possible with an unsynchronized OR join step. A synchronized OR step failure always results in a dead path notification.

When you import a version 7.x process model into a version 8.0 or later environment, any OR joins contained in the model *will continue to function as unsynchronized OR joins*, as described above.

Important:

After you import the version 7.x process model, you are advised to determine whether an unsynchronized or synchronized OR join is the proper construct, depending on your process model.

For these reasons, you can select the join type **Unsynchronized Or** when you specify the join type for a step on the **Joins** page of the Property view.

Note:

Version 7.x process models also handle unsatisfied joins differently than version 8.0 and later models. For more information, see [“How Incomplete Transitions Affect Join Steps and Gateways”](#) on page 132.

About Complex Join Expressions

Many of the available Designer step types support a complex join. The specific behavior of a complex join is defined by a join expression that you create. Evaluation of the join expression at run time determines whether the join is satisfied or unsatisfied.

- If no join expression is defined, process generation will fail with an error stating that the join condition is invalid.
- A warning appears during generation if you do not use all of the available transitions in the condition.
- Designer does not perform any syntax validation. If you enter any unsupported characters in the expression editor, a run-time error will result.

You build a join expression in the expression editor found in the **Join Condition** area of the **Joins** page in the step’s Properties view, as described in [“Defining a Complex Join Expression”](#) on page 138. The join editor enables you to specify the step’s incoming transitions and place them into a logical statement. During run time, the join is satisfied when the conditions in the expression are true.

The join editor has the following features:

- **Expression editing field.** This is where you create and edit the condition expression.

Note:

You cannot type text into the expression editing field. You can only add expression terms using the available controls. This is to ensure that you create a valid expression.

- **Transitions list.** This drop-down list displays all of the step’s incoming transitions. When you select a transition in this list, it is added to the expression as an expression term.
- **Term buttons.** Click these buttons to add a term to the expression. Available terms are **(,)**, **not**, **and**, **or**, and **unsynchronized or**. For more information about unsynchronized or, see [“About OR Join Behavior \(Synchronized vs. Unsynchronized\)”](#) on page 136.
- **Expression editing buttons.** Use these buttons to edit the terms in the expression editing field. Available buttons are Copy, Cut, Paste, Delete, Undo, and Redo. Do *not* use the Paste button to paste content from outside the expression editor field into the expression. Doing so may result in an invalid expression.
- **Delete Expression button.** Click this button to delete all content in the expression editing field.

Example

Consider a step that has three incoming transitions:

- Transition-from-Step-6(StepID:S6)
- Transition-from-Step-5(StepID:S5)
- Transition-from-Step-4(StepID:S4)

You want the join to be considered satisfied if any two of the three incoming transitions are true. The expression would look like this:

```
(Transition-from-S6 and Transition-from-S5) or (Transition-from-S4 and Transition-from-S5)
or (Transition-from-S6 and Transition-from-S4)
```

Defining a Complex Join Expression

If no join expression is defined, process generation will fail with an error stating that the join condition is invalid.

➤ To define a complex join expression

1. In an open process, click a step that has a complex join to select it.
2. Click the **Implementation** page in the Properties view.
3. In the **Join Condition** area, use the join expression editor controls to create the expression you want, as described in [“About Complex Join Expressions” on page 137](#).
4. Save the process.

About Process and Step Timeouts

Software AG Designer enables you to define timeout conditions at the process level and at the step level:

- At the process level, you can configure a timeout that defines the maximum length of time a process can execute. After this time elapses, a designated timeout handler step executes. If you have not created a timeout handler step in the process, the process error handler task is invoked. If no error handler task is present, the process instance fails. This is also known as a *process-wide timeout*. For more information, see [“About Process Timeouts” on page 139](#).
- At the step level, you can configure the following timeouts:
 - A join timeout, for any step with a join.
 - A timer condition, for a boundary timer intermediate event.

Note:

In Process Engine version 8.1 and earlier, a timeout transition could be defined for a process step. With version 8.2 and later, timeout transitions are no longer available and are replaced with the BPMN 2.0 interrupting or non-interrupting boundary intermediate timer events when

an 8.1 or earlier process is imported. For the behavior and availability of boundary timer events, see [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

In general, when you configure a timeout, you can specify:

- A static (fixed) timeout period.
- A timeout based on a process pipeline field.
- A timeout based on a business calendar.

Not all of these options are available in every case. This section describes how the Process Engine handles timeouts.

Defining a Timeout Value

You can define a timeout value for a process, for a join step, or for a boundary timer intermediate event. The value is defined as a *timer condition* for a timer event, and as a *timeout* for a process or join step. For more information, see:

- [“Defining a Process Timeout” on page 139](#)
- [“Defining a Join Timeout” on page 142](#)
- [“Defining a Boundary Timer Event Timer Condition” on page 143](#)

About Process Timeouts

An entire process can have a timeout condition, with an accompanying process timeout handler step. The timeout duration begins with the start of the process instance. You can define the source of the timeout value to be:

- A fixed, absolute timeout value you express.
- A timeout based on a webMethods business calendar.

Typically, you add a process timeout handler step to the process, which is invoked as a result of the timeout. You configure the process timeout handler step to execute whatever behavior you want to apply as a result of the timed-out process.

For more information about designating process timeout values, see [“Defining a Process Timeout” on page 139](#) and [“Process Timer Behavior” on page 141](#).

Note:

You can also set a process timeout dynamically during run time. For more information, see [“Dynamic Timeout Control for Processes” on page 140](#).

Defining a Process Timeout

➤ **To define a process timeout value**

1. In the process editor, click anywhere in the empty process canvas to select the process so you can configure a process timeout.
2. Click the **Timeout** page in the Properties view.
3. In the **Timeout Handler Task** list, select the process step you want to designate as the timeout handler. This step will be executed when the timeout period is exceeded. If you do not define a timeout handler step, the process will attempt to invoke the process error handler step instead. If no error handler step is present, the process fails.
4. In the **Maximum Process Execution Time** area, make the following selections:
 - **Source.** Select one of the following as the source of your timer value:
 - Use a **Static Value** to apply a fixed period of time to the timeout interval. Specify the days, hours, minutes, seconds, and milliseconds of the timeout value.
 - Use a **Business Calendar Value** to select a timer condition based on a business calendar in My webMethods Server. Specify the business calendar you want to work with, and the hours, minutes, and seconds of the timeout value. For more information about configuring a business calendar value, see [“About Business Calendar Timeouts” on page 144](#), [“Business Calendar Prerequisites” on page 144](#), and [“Specifying Business Calendar Timeout Values” on page 145](#).
5. Save the process.

Dynamic Timeout Control for Processes

You can implement dynamic control of process timeouts through a set of WmPRT package services that give you more flexibility during run time to set or cancel a process timer for a specific instance of a process model. These services can be invoked from the process model with a custom flow service, depending on whatever run-time conditions you want to consider.

The public services are listed and described in the following table.

Public Service	Description
pub.prt.timer.process:cancel	WmPRT. This service cancels the process timer for the specified process instance.
pub.prt.timer.process:create	WmPRT. This service creates a process timer for the specified process instance.
pub.prt.timer.process:createWithBusinessCalendar	WmPRT. This service creates a process timer for the specified process instance and the specified business calendar

Public Service	Description
pub.prt.timer.process:createWithDate	WmPRT. This service creates a process timer for the specified process instance and the specified date.
pub.prt.timer.process:get	WmPRT. This service returns the actual date that the timer will expire for the specified process instance.

Process Timer Behavior

When a server starts running the start step of a process instance, the Process Engine for that server creates a timer object in memory and stores the timeout in the Process Engine database. If the Process Engine receives a status control document indicating process completion before the timer expires, the Process Engine cancels the timer.

If the timer expires, the process has timed out. The Process Engine produces a process transition document that identifies the next step to run. The Process Engine publishes the document, and the triggers on all servers retrieve the document and pass it to their own Process Engine.

The following table defines which step will run after the timer expires. The table shows which step takes precedence when more than one type of step is present.

Process has Process-wide Timeout Step	Process has Process-wide Error Step	Result:
✓	✓	The server runs the process timeout handler step.
Not present	✓	The server runs the process error handler step.
Not present	Not present	The process fails.

About Join Timeouts

For most step types, a join timeout can be defined as a static value or dynamic value. A static value is a specific time duration that you define when you design the process model. At run time, the timeout value is applied as defined, and cannot be varied unless you update the process model.

Join timeout on a Intermediate catch message event step, or any join step, starts as soon as either the inbound transition or the document arrives. If the timeout depends on a pipeline variable, you should make sure the process model is designed so that the variable is populated on either of the transitions.

You can also configure a join timeout to use dynamic timeouts that are determined at run time. Dynamic step timeouts apply to AND and COMPLEX join types. You can configure dynamic timeouts in the following ways:

- By specifying a pipeline field value. This enables you to dynamically set the timeout value at run time.
- By specifying a value based on a business calendar. This enables you to count only business days and business hours when calculating the timeout duration. Although the defined timeout period is fixed, the calendar duration of the timeout can vary, depending on presence or absence of working days and non-working days. You can also specify a pipeline field value to use with a business calendar, to provide a true dynamic timeout value.

For more information about setting join timeout values, see [“Defining a Join Timeout” on page 142](#).

Note:

In 7.x versions of Process Engine, join timeouts were required for all join steps. In Process Engine version 8.0 and later, join timeouts are available but are not required. For more information about this change, see [“How Incomplete Transitions Affect Join Steps and Gateways” on page 132](#).

Defining a Join Timeout

A step with an AND or COMPLEX join can be configured with a join timeout condition. For more information about join conditions, see [“About Join Timeouts” on page 141](#) and [“About Joins” on page 131](#).

➤ To define a join step timeout value

1. In the process editor, click a step that contains a join to select the step.
2. Click the **Joins** page in the Properties view.
3. In the **Join Timeout** area, make the following selections:
 - **Source**. Select one of the following as the source of your timer value:
 - Use a **Static Value** to apply a fixed period of time to the timeout interval. Specify the days, hours, minutes, seconds, and milliseconds of the timeout value.
 - Use a **Field Value** to dynamically define the timeout duration by specifying the name of a data field present in the pipeline from an upstream document. Both top-level and nested fields can be specified. The field value is interpreted in milliseconds. For example, if the field value is 60000, the timeout value will be 60000 milliseconds (one minute).
 - Use a **Business Calendar Value** to select a timer condition based on a business calendar in My webMethods Server. Specify the business calendar you want to work with, and the hours, minutes, and seconds of the timeout value. For more information about configuring a business calendar value, see [“About Business Calendar Timeouts” on page 144](#), [“Business Calendar Prerequisites” on page 144](#), and [“Specifying Business Calendar Timeout Values” on page 145](#).
4. Save the process.

About Boundary Timer Event Timer Conditions

You can add a boundary timer intermediate event to all activity types except for user and manual activities. For more information, see [“Adding a Boundary Intermediate Event” on page 289](#). The timer event can be interrupting or non-interrupting, depending on the activity type.

- An interrupting boundary event interrupts the step activity, so when the timer expires, the step stops and the process follows only the transition(s) from the boundary event.
- A non-interrupting boundary event does not interrupt the step activity. When the timer expires, the process follows the transition(s) from the boundary event as well as the transition(s) from the activity as it normally would if there were no timer present.

A boundary timer event can have one or more output transitions. The output transitions from a boundary timer event do not support transition conditions. For information about configuring timer events, see [“About Boundary Intermediate Timer Events” on page 291](#).

When the Process Engine receives the first input for a step with a timer event, it creates a timer object in memory. If the Process Engine receives all step inputs before the timer expires, the Process Engine cancels the timer.

When a server starts running a step's service, the Process Engine creates a timer object in memory. If the server finishes running the service before the timer expires, the Process Engine cancels the timer.

If the timer expires, the step has timed out and transitions to the timeout transition defined for the step. The step produces a process transition document that identifies the next step to run. The Process Engine publishes the document and the triggers for the specific target step, model and model version.

Defining a Boundary Timer Event Timer Condition

For more information about intermediate boundary timer events and setting timeout values, see [“About Boundary Intermediate Timer Events” on page 291](#).

➤ To define a boundary timer event timer condition value

1. In the process editor, click a boundary timer intermediate event to select it.
2. Click the **Timer Condition** page in the Properties view.
3. In the **Timer Condition** area, make the following selections:
 - **Source**. Select one of the following as the source of your timer value:
 - Use a **Static Value** to apply a fixed period of time to the timeout interval. Specify the days, hours, minutes, seconds, and milliseconds of the timeout value.
 - Use a **Field Value** to dynamically define the timeout duration by specifying the name of a data field present in the pipeline from an upstream document. Both top-level and

nested fields can be specified. The field value is interpreted in milliseconds. For example, if the field value is 60000, the timeout value will be 60000 milliseconds (one minute).

- Use a **Business Calendar Value** to select a timer condition based on a business calendar in My webMethods Server. Specify the business calendar you want to work with, and the days, hours, minutes, and seconds of the timeout value. For more information about configuring a business calendar value, see [“About Business Calendar Timeouts” on page 144](#), [“Business Calendar Prerequisites” on page 144](#), and [“Specifying Business Calendar Timeout Values” on page 145](#).

4. Save the process.

About Business Calendar Timeouts

When you configure a process timeout, a join timeout, or a boundary timer event timer condition, you have the choice of basing the timeout value on a webMethods business calendar. Business calendars are created and maintained in My webMethods Server.

Business calendars define standard business days and business hours for your organization, including holidays, weekends, or any other times when your organization is not conducting business. When you specify a business calendar as a timeout value source, only business days and business hours are considered, and non-business days, such as weekends and holidays, and non-business hours are not included in the timeout calculation.

For more information, see:

- [“Business Calendar Prerequisites” on page 144](#).
- [“Specifying Business Calendar Timeout Values” on page 145](#).

For more information about creating and using business calendars, see the PDF publication *webMethods Task Engine User’s Guide*.

Business Calendar Prerequisites

Before you can use the Business Calendar timeout option:

- You must have at least one business calendar defined in My webMethods Server. For more information about creating and using business calendars, see the PDF publication *webMethods Task Engine User’s Guide*.
- You must have an Integration Server configured with the CentralUsers JDBC pool that points to the My webMethods Server instance where the calendar exists. For more information about configuring the CentralUsers JDBC pool, see the PDF publication *webMethods Integration Server Administrator’s Guide*.
- You must have defined an **Integration Server Name** in Designer that points to the configured (and running) Integration Server.
- You must also have Process Engine installed and running.

Specifying Business Calendar Timeout Values

After you specify the business calendar you want to use, you can specify a time value as follows:

- For process timeouts, click the drop-down list in the **Days**, **Hours**, and **Minutes** fields and select a value.
- For join timeouts and timer event timer conditions, click the Browse button next to the field you are working with. This enables you to specify the name of a data field present in the pipeline from an upstream document to dynamically define the value at run time, or to select a fixed value from a list.

The following guidelines apply to the available fields:

- **Days.** Specify the number of days from the starting point of the timeout timer. Only business days as defined by the selected business calendar are counted.
- **Hours.** Specify the number of hours to add or subtract from the **Days** value. A positive value will extend into the next business day, a negative value represents hours before the end of the business day.
- **Mins.** Specify the number of minutes to add or subtract from the **Hours** value.

Note:

Business calendar timing is not precise; the approximate accuracy is about 1 minute. For example, the actual timeout may be triggered up to one minute later than the specified time.

Timeout Recovery Behavior

For process models using non-volatile tracking, timer information is stored in the Process Engine database component. If the host Integration Server terminates unexpectedly or is restarted, the timers can be recovered and restarted.

When the timer is created the expiration timestamp is stored; when the timer is recovered, that same expiration timestamp is used. Note that the Integration Server down time is effectively included in the processing time of the timer; if the timer expires while the Integration Server is offline, when the Integration Server starts, the timer will execute the expiration logic as expected.

Subprocess Concepts

You can create a subprocess in your process model and populate the subprocess with steps and other process components. You can also create a subprocess within a subprocess. Two subprocess types are supported, although one of them is deprecated:

- **BPMN subprocess.** When you add a subprocess to a process model, it is added as a BPMN subprocess. That is, the subprocess implements BPMN 2.0 notation and supports a number of BPMN behaviors that were not available in the webMethods subprocess used in previous releases.

- **webMethods subprocess.** Support for the subprocess behavior used in previous releases is retained, although this subprocess type is deprecated and support will be withdrawn in a future release.

Although webMethods subprocesses will continue to work as expected, there are several changes in subprocess behavior that you must be aware of when you change a webMethods subprocess to a BPMN subprocess.

For more information, see [“Recommendations On Changing the Subprocess Type” on page 244](#) and [“About Changing webMethods Subprocesses to BPMN Subprocesses” on page 244](#). For a summary of the behavior of these two type of subprocess and the differences between them, see [“About Subprocess Types” on page 146](#).

A subprocess exists only within a process model and cannot be run outside the model or referenced from another model. Be sure not to confuse a subprocess with a *callable process* or a *referenced process*, both of which represent a separate process model that exists outside of the current process model.

The primary advantage of a subprocess is that regardless of the number or complexity of the steps contained within it, the subprocess is treated as a single step in the process. You can configure joins and transitions into and out of a subprocess just as you would for a process step, and you can configure a subprocess for standard looping, as described in [“About Standard Looping” on page 129](#).

For more information about subprocesses, see:

- [“About Subprocesses” on page 242](#)
- [“About Subprocess Types” on page 146](#)
- [“Configuring a Subprocess” on page 246](#).
- [“Adding a Subprocess” on page 243](#)
- [“Changing the Subprocess Type” on page 245](#)

About Subprocess Types

In prior releases, subprocesses were built on a unique webMethods container format. Although they supported some BPMN behavior in more recent releases, these containers are referred to as *webMethods subprocesses*. To fully embrace the potential of BPMN 2.0, a BPMN subprocess type is now supported, referred to as a *BPMN subprocess*. When you add a subprocess to a process model, it is added as a BPMN subprocess.

Support for the earlier webMethods subprocess type is retained, although this subprocess type is deprecated and support will be withdrawn in a future release. If you import existing process models from previous releases, the webMethods subprocesses in them will continue to function as they did in previous releases, and the process will build as expected. However, you are encouraged to change these subprocess to BPMN subprocesses at your earliest convenience. For more information, see [“Recommendations On Changing the Subprocess Type” on page 244](#).

The key differences in behavior between the two subprocess types are as follows in the table below:

Behavior	webMethods Subprocess	BPMN Subprocess
Looping and Step Count	In standard looping, each loop iteration increments the step iteration counter. Loop counting is not available.	In standard looping, loop counting is provided. Within each loop iteration, each step in the subprocess begins counting at 1.
Subprocess ID	A subprocess does not have an ID.	Each subprocess has a unique ID.
Subprocess status	A subprocess has no status, and exists only as a collection of steps.	A subprocess always has a status, which is the same as the status of the parent process.
Boundary intermediate event	A boundary intermediate event (timer, message, signal, error) on a subprocess is always non-interrupting.	A boundary error intermediate event on a subprocess is always interrupting. Boundary timer, message, and signal intermediate events can be interrupting or non-interrupting.
End Terminate event behavior	An end terminate in a subprocess always terminates the parent process with a configurable status of Completed, Canceled, or Failed.	End terminate behavior is non-selectable and always ends the subprocess only, with no direct effect on the parent process. For more information, see “About Terminate End Event Behavior in a Subprocess” on page 307.
End behavior	The subprocess ends through the use of a hidden OR join that merges the output of all tracks.	The subprocess ends when there are no active tracks remaining in the subprocess.
Subprocess output	The pipeline output from the subprocess is a merge of all tracks in the subprocess.	The pipeline output from the subprocess is from the last completed track only. To join or merge track outputs, you must do so explicitly in the subprocess.

Call Activity Concepts

When you create a business process, you might have one or more existing process models that you would like to include as part of your new process. If you can invoke these existing processes from within your new process, you can forgo the effort of recreating the existing functionality in your new process. Designer enables you to invoke an existing process model from within another model by implementing a BPMN 2.0 call activity step.

How you invoke the external process depends on the external process type, which can be either a *BPMN callable process* or a *webMethods referenced process*.

- When you create a *BPMN callable process*, you configure it to start with a none start event, and you define an input and output signature for it, known as a *global process specification*. In this case, the call activity step passes the entire pipeline to the none start event in the callable process, and the callable process automatically returns its resulting pipeline to the parent. For more information, see [“About BPMN Callable Processes” on page 148](#).
- A *webMethods referenced process* is any other process model (that is, any process that is not a callable process) that you want to invoke from within your process model. A referenced process model typically starts with a message start event. You must configure start and return documents to facilitate the exchange of pipeline data between the two processes. For more information, see [“About webMethods Referenced Processes” on page 149](#).

Important:

Although still available, the ability to invoke a referenced process from a call activity step is deprecated. You are advised to implement all of your steps that invoke another process with a call activity step that invokes a callable process.

The process containing the call activity step is sometimes referred to as the *parent process*, and the callable (or referenced) process is sometimes referred to as the *child process*. You can configure any call activity step for standard looping, as described in [“About Standard Looping” on page 129](#), regardless of whether it is invoking a callable process or a referenced process.

In addition, you can configure a call activity step to invoke a specific process model statically, or to invoke one or more process models dynamically based on run-time information in the process pipeline. For more information, see [“About Statically-Invoked Processes” on page 151](#) and [“About Dynamically-Invoked Processes” on page 151](#).

For more information about call activity steps and callable processes, see [“About Call Activities” on page 253](#).

About BPMN Callable Processes

In Designer version 8.2 and later, the BPMN 2.0 *call activity* step was introduced to provide the ability to call an external process. A callable process *requires the use of a none start event to start the process*.

Unlike a message start event, the none start event does not take a receive document. A callable process also requires a defined input/output signature, or *global process specification*. When you specify a callable process in the call activity step, the inputs/outputs for the parent call activity step are populated with the input/output signature defined on the **Global Process Specification** page in the Properties view of the callable child process. For more information, see [“Defining a Global Process Specification” on page 97](#).

If a callable process contains a message start event with a subscription filter, that start event and its filter are evaluated *only when the callable process is not behaving as a child process*. When called by a call activity, the required none start event is used. Therefore, a process model can serve as both a callable process and as a referenced process if both a none start event and a message start event are present.

Note:

When a callable process contains both a none start event and a message start event that is configured to use an EDA (deprecated) event type, the message start step and the none start step of the child process must be on different Integration Servers (that is, the Integration Server **Name** property must be different for each of the two steps). Otherwise a validation error occurs during the build and upload procedure.

The callable process does not use the start and return document method of invocation. Instead, the call activity step passes the entire process pipeline to the none start event in the child process. The child process automatically returns its resulting pipeline back to the parent. For more information about call activity steps and callable processes, see [“About Call Activities” on page 253](#).

You can configure a call activity step to use static or dynamic invocation of a callable process:

- **Static invocation.** In this implementation, the call activity step starts an instance of a specific callable process model with a set of process properties that are defined at design time. The same process model is *always* called when the call activity step is executed, and the parent process will always wait for this child process to complete. For more information, see [“About Statically-Invoked Processes” on page 151](#). You can start multiple serial instances of the callable process by configuring the call activity step for looping.
- **Dynamic invocation.** In this implementation, you can start one or more callable processes at run time, and, if desired, multiple instances of each of those processes. Furthermore, you define which callable process models are to be started dynamically at run time, depending on pipeline data. You implement this by creating a document reference list of type `pub.prt:CallActivityModel` called `CallActivityModels` in the pipeline. The data in this document reference list is used to invoke the desired callable process(es). For example, for orders originating in different countries, a document could start process A for one country, and process B for another country. See [“About Dynamically-Invoked Processes” on page 151](#).

About webMethods Referenced Processes

A webMethods referenced process is any process model that can be started with a message start event, typically configured with subscription filters. When a process model contains both a message start event and a none start event and contains a global process specification, it can serve as both a referenced process and a callable process.

Important:

Although still available, the ability to invoke a referenced process from a call activity step is deprecated. You are advised to implement all of your steps that invoke another process with a call activity step that invokes a callable process.

When you invoke a referenced process with a call activity step, you are able to use other BPMN constructs such as boundary timer and error events with your referenced process. When a referenced process is invoked by a call activity, it executes in the same way as a standard webMethods process.

To call a referenced process with a call activity step, you must configure referenced process behavior for the call activity step. In addition:

- The process must contain a message start event.

- You must configure start and return documents. For more information, see [“Specifying Referenced Process Start and Return Documents” on page 96](#).
- The child process must publish all the specified return documents back to the parent. For more information, see [“Specifying Referenced Process Start and Return Documents” on page 96](#).

Upon execution of the call activity step, the parent process will wait for the child process to complete before moving to the next step in the process.

You can configure a webMethods referenced process step to use static or dynamic invocation:

- **Static invocation.** In this implementation, the referenced process step starts an instance of a specific reference process model with a set of process properties that are defined at design time. The same process model is *always* called when the call activity step is invoked, and the parent process will always wait for this child process to complete, regardless of the expectation of return data. See [“About Statically-Invoked Processes” on page 151](#). You can start multiple serial instances of the referenced process by configuring the referenced process step for looping.
- **Dynamic invocation.** In this implementation, you can start one or more referenced processes at run time, and, if desired, multiple instances of each of those processes. Furthermore, you define which process models are to be started dynamically at run time, depending on pipeline data. You implement this by creating a document reference list of type `pub.prt:SubprocessModel`, called `SubprocessModels` in the pipeline. The data in this document reference list is used to invoke the desired reference process(es). For example, for orders originating in different countries, a document could start process A for one country, and process B for another country. See [“About Dynamically-Invoked Processes” on page 151](#).

About the webMethods Referenced Process Step

In Software AG Designer Process Development version 8.1 and earlier, the only means of calling an external process from within a process model was to use a proprietary webMethods referenced process step. This webMethods referenced process step enabled you to invoke one or more referenced processes from within a process model.

The webMethods referenced process step available in version 8.1 and earlier was removed from the palette of steps in version 8.2 and was replaced with the BPMN 2.0 call activity step. For backwards compatibility, a call activity step can be configured to replicate the behavior of the removed webMethods referenced process step. For more information, see [“About webMethods Referenced Processes” on page 149](#).

Important:

Although still available, the ability to invoke a referenced process from a call activity step is deprecated. You are advised to implement all of your steps that invoke another process as BPMN 2.0 call activity steps that invoke a callable process.

Note:

When imported from versions of Designer earlier than 8.2, referenced process steps are converted to call activity steps configured for referenced process behavior.

About Statically-Invoked Processes

When you invoke a callable process or a referenced process statically, you specify the name of a specific process model; this process is *always* the process that is invoked by this step. You can create a static call activity step by dragging an existing process onto the process canvas in Designer, or by creating an empty call activity step on the canvas and configuring it as required. For more information, see [“Configuring Call Activities” on page 254](#).

During run time, the parent process pauses when the callable or referenced process is started, waiting for the child process to complete, just as it would for an internally-executed step, and then resumes after the child process has completed. For referenced processes, you can specify any return documents that the parent process expects to receive back from the referenced process.

A static referenced process call activity step must always wait for the child process to complete before proceeding to the next step in the process. You can execute multiple instances of the child process by configuring the call activity step for standard looping, as described in [“About Standard Looping” on page 129](#).

About Dynamically-Invoked Processes

The dynamically-invoked process feature is extremely flexible. When you invoke a callable process or a referenced process dynamically, you can:

- Invoke one or more process models, without prior design-time knowledge of the callable or referenced process model to use. These processes can be invoked to run simultaneously.
- Invoke multiple instances of a single process model without configuring looping on the call activity step.
- Invoke multiple instances of multiple process models, also without configuring looping.
- Specify whether or not the parent process must wait for the child process to complete before transitioning to the next step.

These dynamically invoked process models are triggered at run time using information in the pipeline, and by applying the standard document handling, step logic, and data mapping features available in the Process Development and Service Development perspectives in Software AG Designer. You make use of a specific document type that describes the information needed to invoke a dynamic referenced process, `pub.prt.CallActivityModel` for callable processes and `pub.prt.SubprocessModel` for referenced processes.

You will need to be familiar with the above features to be able to implement dynamically-invoked processes. For additional information, see:

- [“Mapping Data in Flow Services” in *webMethods Service Development Help*](#).
- [“Configuring a Process” on page 57](#).
- [“pub.prt.CallActivityModel” on page 189](#)
- [“pub.prt.SubprocessModel” on page 198](#).

To obtain this functionality, the recommended approach is to create a call activity step on the process canvas, configure it as required, and then enable it for dynamic execution at run time.

When the Process Engine encounters the dynamic call activity step, it performs a process invocation, but instead of invoking a specific process defined within the step, it invokes the process with values found in the pipeline.

Important: Designer *never* implements Express Pipeline for dynamic call activity steps, regardless of the **Express Pipeline** option setting. For more information, see [“Setting Quality of Service for a Process” on page 111](#).

To implement a dynamic call activity step, you must select the **Allow this step to dynamically invoke one or more processes at run time** check box on the **Implementation** tab of the call activity step's properties in Designer. As with any parallel step executed in a process model, select the **Allow Parallel Execution** check box in the same location. For more information, see [“Parallel Execution \(Step Locking\)” on page 117](#).

You must also:

- Configure any other behavior of the call activity step as required (transitions, looping, KPIs, and so on). For more information, see [“Configuring Call Activities” on page 254](#).
- Define the required invocation data by creating a document reference list of the required document type (`pub.prt.CallActivityModel` for callable processes, `pub.prt.SubprocessModel` for referenced processes). You must create one element of the document reference list for each dynamically invoked process model you want to start. For more information, see [“pub.prt.CallActivityModel” on page 189](#) and [“pub.prt.SubprocessModel” on page 198](#).

Note:

You can also start a referenced process with a publish rather than a direct invocation; however, that approach does not tie the child process to the parent process, and the two processes will not be related when you view them in webMethods Monitor.

Invoking Multiple Instances of a Dynamic Process

The `pub.prt.CallActivityModel` and `pub.prt.SubprocessModel` document type enables you to generate multiple instances of each callable or referenced process you invoke. For example, suppose you are creating a purchase order handling process, and you already have added a call activity step (configured with dynamic process behavior) that invokes an existing process model that calculates the shipping time for an inventory item.

If the purchase order process is handling a purchase order with 10 line items, you can configure the call activity step to invoke the shipping time calculator process 10 times—once for each line item—by implementing standard looping in the call activity step.

By mapping the key data from each line item into a separate document, this document can be passed to the dynamically invoked process for calculation. Each instance of the shipping time calculator process returns an output document to the purchase order process with the results of the calculation for each line item.

These return documents are maintained in the process pipeline as a named output document lists for callable processes, and as a document list for referenced processes. For more information, see [“pub.prt.CallActivityModel” on page 189](#) and [“pub.prt.SubprocessModel” on page 198](#).

Dynamically Invoking Multiple Processes

It is possible to dynamically invoke two or more processes from a single call activity step that is configured for dynamic process behavior. This can be done by mapping pipeline data (for example, purchase order data) into a document reference list that specifies two or more document type instances:

- Use `pub.prt.CallActivityModel` as the document type for callable processes.
- Use `pub.prt.SubprocessModel` as the document type for deprecated referenced processes.

By creating the necessary logic, you can pass pipeline data to each dynamically invoked process as required; furthermore, you can create multiple instances of each process, as described in [“Invoking Multiple Instances of a Dynamic Process” on page 152](#).

Special considerations must be taken to ensure proper document handling. For more information, see [“Handling SubprocessModel Documents with Dynamically Invoked Processes” on page 154](#).

Synchronous or Asynchronous Behavior of Dynamically Invoked Processes

When you dynamically invoke a process from a call activity step, you can configure the call activity step to wait for return information from the invoked process, or to continue to the next step in the process while the dynamically invoked process continues to run in parallel.

This behavior is not available with statically invoked processes. For more information, see:

- [“About the Return Behavior of Child Processes” on page 153](#)
- [“pub.prt.CallActivityModel” on page 189](#)
- [“pub.prt.SubprocessModel” on page 198](#)

About the Return Behavior of Child Processes

Whether a process is started as a referenced process or a callable process, the result is a running instance of a child process. Ideally, the child process completes with no errors or failures. However, errors and cancellations do occur, and there are several factors that affect how the child process interacts with the parent process in these situations.

When a child process fails or is canceled, the call activity step in the parent process considers this to be an exception and performs the usual error handling (if present). This is consistent with the "static" invocation of a callable or referenced process. In other words, the child process must complete normally for the parent step to continue normally.

When a child process has no error handling and an error occurs, the failure is not escalated to the parent process. Instead, the child process status is set to Failed - Not Escalated, allowing other

tracks in the instance to execute until their logical conclusion. When the child process has the Failed status, it can be resubmitted in order to attempt to complete it.

Handling CallActivityModel Documents with Dynamically Invoked Processes

This document type is described in “[pub.prt.CallActivityModel](#)” on page 189. When invoking a process with dynamic process behavior, it is possible to populate the `pub.prt.CallActivityModel` document list to invoke two or more callable processes.

Multiple entries in the `pub.prt.CallActivityModel` list with the same `CallActivityModelId` value are supported. The output document list with the name specified in the **OutputPipelineName** field contains all entries from executed models where the `WaitForCallActivity` parameter is set to `true`. The order of the documents in the output document list is the same as the order in the `pub.prt.CallActivityModel` input list.

Handling SubprocessModel Documents with Dynamically Invoked Processes

This document type is described in “[pub.prt.CallActivityModel](#)” on page 189. When invoking a process with dynamic process behavior, it is possible to populate the `pub.prt.SubprocessModel` document list to invoke two or more callable or referenced processes. For example, when the same dynamically invoked process is called twice, the parent process continues after the first child finishes and does not wait for results from the second child process.

Multiple entries in the `pub.prt.SubprocessModel` list with the same `SubprocessModelId` are supported. The return document list contains all entries from executed models where the `WaitForSubprocess` parameter is set to `true`. The order of the documents in the return document list is the same as the order in the `pub.prt.SubprocessModel` input list. For example:

- ModelA returns Doc1
- ModelB returns Doc1

```
SubprocessModels[0]
  WaitForSubprocess - true
  SubprocessModelId - ModelA
  ReturnDocuments - Doc1
  SubprocessInstances[0] - *
    +Inputs[0]
      #Type - ModelATrigger
      #Document - ModelATriggerDoc1
  SubprocessInstances[1] - *
    +Inputs[0]
      #Type - ModelATrigger
      #Document - ModelATriggerDoc2

SubprocessModels[1]
  WaitForSubprocess - true
  SubprocessModelId - ModelB
  ReturnDocuments - Doc1
  SubprocessInstances[0] - *
```

```
+Inputs[0]
  #Type - ModelBTrigger
  #Document - ModelBTriggerDoc1
```

There are two instances of ModelA and one instance of ModelB invoked by the input structure above. The return document list will contain a list of Doc1 documents in the same order as shown above. Each input instance is marked with a number of asterisks that match the output document:

```
Doc1[0] * Doc1[1] ** Doc1[2] ***
```

Note that multiple entries in the `pub.prt.SubprocessModel` list with the same `SubprocessModelId` value and different `WaitForSubprocess` settings can make it difficult to locate the return documents. `pub.prt.SubprocessModel` entries that have the `WaitForSubprocess` parameter set to `false` will not have return documents in the list. For example:

■ ModelA returns Doc1

```
SubprocessModels[0]
  WaitForSubprocess - false
  SubprocessModelId - ModelA
  ReturnDocuments - Doc 1
  SubprocessInstances[0] - *
  +Inputs[0]
    #Type - ModelATrigger
    #Document - ModelATriggerDoc1

SubprocessModels[1]
  Wait For Subprocess - true
  Subprocess ModelId - ModelA
  ReturnDocuments - Doc1
  SubprocessInstances[0] - *
  +Inputs[0]
    #Type - ModelATrigger
    #Document - ModelATriggerDoc2
  SubprocessInstances[1] - *
  +Inputs[0]
    #Type - ModelATrigger
    #Document - ModelATriggerDoc3
```

There is one instance of ModelA in the first `pub.prt.SubprocessModel` and two instances of ModelA in the second `pub.prt.SubprocessModel` invoked by the input structure above. The return document list will contain a list of Doc1 documents in the same order as shown above. Each input instance is marked with a number of asterisks that match the output document:

```
Doc1[0] ** Doc1[1] ***
```

The `SubprocessInstance` with a single asterisk (*) has the `WaitForSubprocess` parameter set to `false`, so it is not included in the return document list.

About Retries

Software AG business processes provide the capability to automatically retry either a webMethods referenced process or a BPMN callable process associated with a call activity step. To enable this behavior, you must set the **Retry Count** (on the **Implementation** page in the Properties view of

the call activity step) to a value greater than 0 to indicate the number of times the child process is to be retried after an initial failure.

A value of 3, for example would retry the child process 3 times after the initial failure for a total of 4 executions. You may also specify a **Retry Interval** (specified in milliseconds) to indicate how much time should elapse between retry attempts.

When a retry is configured, the parent call activity will retry the child process after the configured retry interval. If the child process fails again, the retry algorithm is invoked, until the retry count is reached.

If the retry count is reached, the error handling configured in the parent call activity is executed. If the child process completes normally during a retry attempt, no further retry attempts are made and the call activity continues as normal.

The retry algorithm is not activated when:

- The child process is canceled. In this case, the normal cancel-handling logic is executed at the parent call activity step.
- The child process ends in failure and the child is configured to “not escalate” the failure to the parent process.

In other words, the retry algorithm is activated only when the child process ends in failure *and* escalates that failure to the parent process.

When a retry attempt is made, a child process instance is created with the same process ID as the original child invocation but with an incremented “instance iteration.” Retry attempts of a child process do not result in separate step iterations of the parent call activity. In other words, a single iteration of a call activity step may be the parent of multiple child processes during the retry algorithm.

Important:

Although it is permissible in the system, it is *not* advised to resubmit a child process that is also configured to retry automatically upon failure. Doing so can result in unexpected conditions.

Handling Process and Step Errors

In Process Engine version 8.1 and earlier, an error transition could be defined for a process step. With version 8.2 and later, error transitions are no longer available and are replaced with the BPMN 2.0 intermediate boundary error events when an 8.1 or earlier process is imported. For more information, see [“About Boundary Intermediate Error Events” on page 291](#).

When you design a process, you can specify a process error handler step to be invoked when an error occurs in the process instance. You define a process wide error task on the **Error** tab in the Properties view of the process in Software AG Designer. The process passes the error pipeline from a failed step to the process error handler step, along with the ExceptionTransitionInfo document. You can then configure the behavior of the process error handler step to act on this information, with the goal of enabling the process to recover from the error and continue processing to a successful completion.

Providing proper error handling for your process model is considered a best practice. Implementing robust error handling within your process model helps ensure stable and predictable operation at run time.

For more information about designating a process error handler task, see [“Configuring a Process” on page 57](#).

About Step Failure Behavior

When a step in a process instance fails, Software AG Designer provides you with the ability to define an error handling step for the process. For more information, see [“Handling Process and Step Errors” on page 156](#).

Errors (sometimes referred to as *exceptions*), such as a step failure, can be processed by an error handling step configured with your own custom logic, which can enable the running process to recover from the error and continue execution to a successful completion.

In a process without any error handling, the failure of a step generally results in the failure of the process. You can troubleshoot a failed process using webMethods Monitor, which also enables you to make changes to the process instance pipeline, and to resubmit the process instance from a failed step or from any other resubmit-enabled step in the process. For more information, see [“Enabling a Step for Resubmission” on page 76](#). For a fuller description of step and process resubmission, see the “Process Monitoring” chapter in the *webMethods Monitor User’s Guide*.

- In Process Engine version 9.7 and later, when a step fails with no error handling in place, the process instance continues to execute until it reaches a logical conclusion, at which point the process instance is given a status of Failed. For example, if a process model has two or more processing tracks, even though a step in one track fails, the remaining tracks continue executing to a logical conclusion. This greatly improves the results when the failed process is resubmitted.
- In versions earlier than 9.7, when a step fails with no error handling, the process instance stops processing immediately, and the process is given a status of Failed. Any parallel tracks are halted after the completion of the currently executing step, and these tracks do not run to a logical conclusion. This makes resubmission more difficult.

If you are migrating process models from earlier versions to version 9.7 or later, this change in unhandled error processing may have an impact on the behavior of your imported models. It is possible to revert to the pre-9.7 behavior on a process-by-process basis by applying the **Enable deprecated error handling behavior** option. However, this option is deprecated and is not recommended due to its negative impact on resubmission. You are advised to modify your process model as required to avoid using it. For more information, see [“Advanced Process Properties” on page 61](#).

About Boundary Event Error Behavior in a Subprocess

Boundary error events can be added to tasks within a subprocess, to the subprocess itself, or to both. In general, when an error occurs, it is caught by the boundary error event placed on the step or subprocess where the error occurred. If no boundary error event is present, the subprocess fails and is waiting to be resubmitted.

The following table describes the behavior of subprocess activity step errors in various situations:

An error occurs in a subprocess activity step	The activity step has a boundary error event	The subprocess has a boundary error event	Process has a process error step	Result:
✓	Yes	No	Not applicable	Error is caught by the activity step boundary error event.
✓	Yes	Yes	Not applicable	Error is caught by the activity step boundary error event.
✓	No	Yes	Not applicable	Error is caught by the subprocess boundary error event.
✓	No	No	Yes	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.
✓	No	No	No	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.

The following table describes the behavior of subprocess errors in various situations:

A subprocess executes an boundary error event	The subprocess has a boundary error event	Process has a process error step	Result:
✓	Yes	Not applicable	Error is caught by the subprocess boundary error event.
✓	No	Yes	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.
✓	No	No	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.

About End Error Event Behavior

You can add an End Error Event directly to a process or inside a subprocess. The following tables describe the behavior of the End Error Event in a subprocess and in a child process called by a call activity step.

The following table lists the end error event behavior in a subprocess:

A subprocess executes an end error event	The subprocess has a boundary error event	Process has a process error step	Result:
✓	Yes	Not applicable	Error is caught by the subprocess boundary error event.
✓	No	No	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.
✓	No	Yes	The status of the subprocess is set to Failed and the subprocess is waiting for resubmission.

The following table lists the end error event behavior in a child process:

A child process executes an end error event	The call activity step has a boundary error event	Process has a process error step	Result:
✓	Yes	Not applicable	Error is caught by the call activity step boundary error event.
✓	No	Yes	The status of the child process is set to Failed and the child process is waiting for resubmission.
✓	No	No	The status of the child process is set to Failed and the child process is waiting for resubmission.

Process Cancellation

At design time, you can designate a step in a process as the cancel handler step (similar to the process error handler step); the Process Engine executes the specified task upon cancellation.

When the user cancels a process instance from webMethods Monitor or with the [pub.prt.admin.changeProcessStatus](#) service:

- If a cancel handler step is specified, the cancel handler step is executed along with any steps downstream from the cancel step. All other process activities are stopped.
- If a cancel handler step is not specified, all process activities are stopped and the process instance is canceled with no further action.

Note:

Any running steps will not be interrupted and will continue running until they attempt the next transition.

For more information about designating a cancel handler step, see [“Configuring a Process” on page 57](#).

Process Suspension

On occasion, you may want to suspend a single process, or suspend all processes that are currently running; the latter capability is useful for enabling periods of maintenance.

The WmPRT Process Engine package installed on webMethods Integration Server contains two built-in public services to support this functionality:

- [“pub.prt.admin:suspendProcesses” on page 186](#)
- [“pub.prt.admin:resumeProcesses” on page 183](#)

You can use the `suspendProcesses` service to suspend a single specified process, or all running processes; the action applies to all versions of the targeted process model(s). Similarly, the `resumeProcesses` service enables you to resume a single specified process, or all suspended processes. This capability can be applied to a single server instance of Process Engine, or to a clustered installation.

For more information about these and other services, see [“Process Engine Services” on page 175](#). This topic is also available in the PDF publication *Administering webMethods Process Engine*.

Process Debugging

From time to time, a process model may fail to execute as expected in the Process Engine run-time environment. Although it is not part of the Process Engine run-time environment, a Process Debug perspective is available in Software AG Designer. By loading a process model into this perspective, you can analyze the process behavior with these capabilities:

- **Trace View** — The Trace view enables you to control your navigation through steps, and to see the progress of those steps, including errors, during a process debugging session.
- **Pipeline Data View** — The Pipeline Data view displays the pipeline data associated with the output of the step selected in the Trace view. You can expand the information displayed to see more details of the data from the pipeline for the selected step, providing an opportunity to troubleshoot the behavior of the overall process at the step level. You can also copy the pipeline of one or more steps to the clipboard, and you can modify certain pipeline data fields to test alternative behaviors.
- **Breakpoints View** — The Breakpoints view displays a list of breakpoints that exist in your workspace. The Breakpoints view has its own tool bar where you can enable and disable selected breakpoints, remove selected or all breakpoints, and skip all breakpoints.

About Document Correlation

Correlation is used to enable external documents to join running process instances. Each step that receives input (a subscription document) can use a correlation. Each of these steps has a **Correlation** page in the Properties view. A message start event, a signal start event, or a receive task designated **Allow this receive task to start new process instance** can start a new process instance.

Catching message intermediate events, Catching signal intermediate events, and receive tasks NOT designated **Allow this receive task to start new process instance** also use correlations.

In a typical run-time environment, instances of many different process models can be running simultaneously. When a published document arrives in the run-time environment, the Process Engine must determine which of the running processes the document is intended for. You must provide a means of correlating a particular document instance with a particular process instance. This can be done in two ways:

- By creating a correlation service the flow service editor in Designer and associating it with a receive step.
- By specifying a field in an incoming document to serve as a correlation field when configuring a receive step.

When you create a receive task in a process model, you can specify that it uses either a correlation field or a correlation service for document correlation, or no correlation. Correlation IDs are stored and tracked by the Process Engine; for more information, see [“Tracking Correlation IDs” on page 168](#).

Correlation services are written in the flow service editor in Designer, and establish or match the correlation ID used by a process instance. For more information about using correlation services, see [“About Correlation Services” on page 162](#).

- For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.
- For specific information about the correlation services, see [“pub.prt.correlate:deleteCorrelation” on page 190](#) and [“pub.prt.correlate:establishCorrelation” on page 191](#).

Tip:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

Note:

For intermediate receive steps, correlation retry behavior is defined in the subscription trigger. See the PDF publication *Publish-Subscribe Developer's Guide*.

About Correlation Services

You can create a correlation service in Software AG Designer as a flow service, using the specification “[pub.prt:CorrelationService](#)” on page 179. The correlation service identifies specifically named IS documents and routes them to a specific running instance of the process model.

All documents bound for the same instance of the process must use the same correlation ID. Similarly, correlation IDs must be unique across all process instances.

In addition, the following related services are available:

- [pub.prt.correlate:deleteCorrelation](#). Deletes all mappings between the specified process instance ID and any correlation IDs or conversation IDs.
- [pub.prt.correlate:establishCorrelation](#). Sets up a correlation between a correlation ID and a process ID or between a conversation ID (for a Trading Networks document) and a process ID.
- [pub.prt.correlate:lookupCorrelation](#). Returns the process instance ID that is associated with the specified correlation ID or conversation ID. If no association exists, creates a new process instance ID and mapping.

To create a correlation service, you should be familiar with flow services, business logic, and the data mapping functionality available in Software AG Designer. Essentially, you create a new, empty flow service and associate it with the specification “[pub.prt:CorrelationService](#)” on page 179.

For additional information, see the *webMethods Service Development Help*.

About Correlation Fields

Process Engine enables you to specify a correlation field to apply correlation on certain receive tasks, signal start events, and message start events, thereby establishing correlation without specifying or creating a correlation service. In this case, the specified field exists in the receive document and serves as the correlation identifier.

When a correlation service is executed, the expected output is a `ProcessCorrelationID` that is used as the correlation identifier. Instead of executing a service, the Process Engine retrieves the specified correlation field from the input document and uses it as the correlation identifier.

Correlation Behavior with Non-Starting Events and Activities

The following components can be configured to receive a document but not start a process or subprocess:

- Catching message intermediate event.
- Catching signal intermediate event.
- Receive task.

In some cases, the document may be delivered to these components when the process or subprocess they belong to is not running. In these cases, the following behavior, as described in the table below, occurs:

For this component The Process Engine	
Top-level Process	Performs a correlation based on a field or service, but because the process instance is not running, the document cannot be correlated. The Process Engine rejects the document and generates an exception that causes the trigger to determine what to do with the document.
BPMN Subprocess	Performs a correlation based on a field or service (assuming the parent process is running). If the BPMN subprocess instance is not running, the Process Engine rejects the document and generates an exception that causes the trigger to determine what to do with the document.
webMethods Subprocess	Performs a correlation based on a field or service (assuming the parent process is running). If the webMethods subprocess instance is not running, the document is correlated and delivered to the subprocess when the subprocess starts running.

The available trigger options are:

- Suspend the trigger after the specified trigger retries are exhausted.
- Discard the document after the specified trigger retries are exhausted.

These are standard trigger options; for more information, see the *Publish-Subscribe Developer's Guide*.

Important:

It is the *trigger properties* that control this document exception behavior and *not* the retry step properties.

Creating Correlation Services

You can create a correlation service with custom logic for a new process. Use the flow service editor in Designer to create the service. For more information about data mapping in Designer, see "Mapping Data in Flow Services" in *webMethods Service Development Help*.

Tip:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

Correlation Service Input and Output Variables

Refer to the following tables for input and output variables you can use when creating correlation services.

When a correlation service receives control, it passes the inputs described in the following table:

Input variable	Description
<i>ProcessModelID</i>	String ID of the process model with which this invocation of the correlation service is involved.
<i>ProcessModelVersion</i>	<p>String Version of the process model with which this invocation of the correlation service is involved.</p> <p>Note: Because a single correlation service can be associated with steps from more than one process model version, you can use the <i>ProcessModelID</i> and <i>ProcessModelVersion</i> to identify the process model version using the correlation service at run time.</p>
<i>LogicalServer</i>	<p>String Name of the logical server that is associated with the step in the process model version with which this invocation of the correlation service is involved. In other words, the name of the logical server on which this correlation service is running.</p> <p>Because a single correlation service can be used with steps that execute on different servers, you can use <i>LogicalServer</i> to identify a specific server at run time.</p>
<i>ProcessStepID</i>	<p>String ID of the step in the process model version with which this invocation of the correlation service is involved (for example, N3).</p> <p>Because a single correlation service can be associated with multiple steps in a process model version, you can use <i>ProcessStepID</i> to identify the specific step at run time.</p>
<i>DocumentName</i>	String Name of this document as used in the process model version (for example, "OrderDocument").
<i>DocumentType</i>	String Name of this document type (for example, "orders.sap:OrderDocument").
<i>Document</i>	Document The document.

Your correlation service should return the output described in the table below:

Output variable	Description
<i>ProcessCorrelationID</i>	<p>String Conditional. An abstract ID that correlates to the actual process instance ID of the running process. For example: "CUSTOMER-0003456977::ORDER-19477593-AR9-1000". All documents bound for the same instance of the process must return the same correlation ID. Similarly, correlation IDs must be unique across all process instances.</p>

CorrelateAsTN

String Conditional. Flag that indicates whether the correlation ID in *ProcessCorrelationID* is a conversation ID.

- A value of true indicates that *ProcessCorrelationID* is a Trading Networks conversation ID.
- A value of false indicates that *ProcessCorrelationID* is not a Trading Networks conversation ID.

For more information, see [“pub.prt:CorrelationService” on page 179](#) and [“pub.prt.correlate:establishCorrelation” on page 191](#). This information is also available in the PDF publication *Administering webMethods Process Engine*.

Specifying Correlations

Receive tasks designated **Allow this receive task to start new process instance**, message start events, and signal start events should initiate a correlation. Receive tasks NOT designated **Allow this receive task to start new process instance**, catching message intermediate events, and catching signal intermediate events should look up a correlation key or service. In all cases, you specify the correlation on the Correlation page in the Properties view of the step.

➤ To specify a correlation

1. In the process editor, select a receive task, message start event, signal start event, catching message intermediate event, or catching signal intermediate event.
2. On the Correlation page in the Properties view, do one of the following:
 - Select **Not Used** to specify no correlation service.
 - Select **Field** to specify a field from the subscription document as the correlation key. Expand the list to see available fields from which you can select.
 - Select **Browse** to open the Choose Service window and locate a service on an Integration Server or in metadata.
 - Select **New** to open the New Flow Service window and create a new service on a configured Integration Server.
 - Select **View** to open the selected service in the flow service editor.

Note:

The **View** button is available only when a service is specified in the **Service** field.

Tip:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

About Document Merging

Although the Process Engine allows you to create a process model that will merge documents in the pipeline, this configuration can yield unpredictable behavior when you attempt to do this with different instances of an IS document with the same name. For example:

- You pass different instances of an IS document with the same name to two branches within a process and later merge them with an AND join.
- You implement transition loop in the process, where the process flow loops back to a join step where two or more documents of the same name are merged.

Important:

The Process Engine does not support parallel data management and you are strongly advised to avoid allowing the same document in a process to be accessed concurrently on different branches within the process.

Tip:

Best practices recommend that you manage the pipeline by implementing services within a process loop to ensure that old document references are removed from the pipeline when a looping condition occurs, as well as when they are no longer needed.

About Process Tracking

This section explains how a Process Engine tracks the execution of a process. The Process Engine uses the same tables for process tracking that are used for process execution (see [“Process Execution Table” on page 110](#)).

Tracking Process Start

When a Process Engine receives an external document from the subscription trigger, the Process Engine reads the Process Audit Log database component to find out if the process model version is enabled. If it is, the Process Engine starts the process. The sections below describe how Process Engines track process start based on your configuration and tuning settings.

Using Volatile Tracking

The Process Engine creates a process instance ID. It stores the process instance ID, sets the process status to **Started**, and the iteration count 1 in server memory. The Process Engine then publishes the new process status as described in [“Tracking Process Status” on page 167](#), below.

Using the Process Engine Database Component

The Process Engine creates a process instance ID. It stores the process instance ID and sets the process status to **Started** in the PRTPROCESS table in the Process Engine database component, and increments the instance iteration count in the PRTINSTANCEITER table by 1. The Process Engine then publishes the new process status as described in [“Tracking Process Status” on page 167](#), below.

Tracking Process Status

The status of a process can be changed programmatically (for example, when a step fails) or manually (for example, when a webMethods Monitor user suspends a process). Before running a step, each Process Engine checks the process status to determine if its server should run the step.

When a webMethods Monitor user has suspended the process, the Process Engine will not run subsequent steps until the process has been resumed. When a step fails, the Process Engine will not run subsequent steps until the process has been recovered or resubmitted.

Notification of the change in process status is carried out as follows:

- When a step changes the process status, the Process Engine for the server that ran the step publishes a status control document.
- When a webMethods Monitor user changes the process status, webMethods Monitor publishes a status control document.

The method used by the Process Engine to track process status depends on your configuration and tuning settings.

- When volatile tracking is enabled (the document is stored in memory), the control triggers on all servers that run process steps retrieve the document and pass it to their Process Engines. Each Process Engine updates the process status in its server memory.
- When volatile tracking is disabled (the document is stored in the Process Engine Database Component), the control triggers on all servers that run process steps retrieve the document and pass it to their Process Engines. Each Process Engine updates the process status in the PRTPROCESS table in the Process Engine database component.

In both of the above cases, the Process Engines also publish a document to the Integration Server audit subsystems indicating the new process status. The audit subsystems update the PRA_PROCESS table in the Process Audit Log database component with the new status.

Tracking Process Completion

The Process Engine uses the *process thread count* to track process completion. Each change to the process thread count indicates whether a step produced output as one of the following: a pipeline, a process transition document, or a referenced process document.

As each step executes the Process Engine keeps track of the number of outstanding process threads that are currently executing. Each branch of a process model is a thread of execution. When all threads have completed and the total thread count is 0 the process is complete.

The method used by the Process Engine to track process completion depends on your configuration and tuning settings.

- When volatile tracking is enabled, each server that runs process steps maintains the process thread count in memory. After a server runs a step, the Process Engine for that server publishes a tracking document. Each Process Engine retrieves the document and updates its server's memory.

If the count shows that a step was the last step, the Process Engine for the server that ran the last step publishes a status control document indicating process completion. The triggers on all servers retrieve the document and pass it to their Process Engines. The Process Engines store the process status in server memory.

- When volatile tracking is disabled, each server that runs process steps maintains the process thread count in the WMPRTSTORE table in the Process Engine's database component. After a server runs a step, the Process Engine for that server updates the WMPRTSTORE table.

If the count shows that a step was the last step, the Process Engine publishes a status control document indicating process completion. The control triggers on all servers retrieve the document and pass it to their respective Process Engines.

In both of the above cases, the Process Engines also publish a document to the Integration Server audit subsystems indicating the new process status. The audit subsystems update the PRA_PROCESS table in the Process Audit Log database component with the new status.

Tracking Correlation IDs

The following sections describe how the Process Engine tracks correlation IDs based on your configuration and tuning settings.

- [“Tracking Correlation IDs with Volatile Tracking” on page 168](#)
- [“Tracking Correlation IDs with the Process Engine Database Component \(Volatile Tracking Disabled\)” on page 168](#)

Tracking Correlation IDs with Volatile Tracking

The Process Engine stores correlation IDs in server memory. Each correlation ID is associated with the process instance ID for a running process.

When the Process Engine receives an external document for a process that uses a correlation ID, the Process Engine runs the correlation service for the appropriate step and compares the correlation ID to the correlation IDs in server memory.

- If the ID matches an ID for a running process, the Process Engine loads the external document content into server memory and the server executes the step as part of the running process.
- If the ID does not match an ID for a running process, the Process Engine creates a process instance ID, stores the process instance ID and the correlation ID in server memory, and starts a new process.

Tracking Correlation IDs with the Process Engine Database Component (Volatile Tracking Disabled)

The Process Engine stores correlation IDs in the WMPRTXREF table in the Process Engine database component. Each correlation ID is associated with the process instance ID for a running process.

When a Process Engine receives an external document for a process that uses a correlation ID, the Process Engine runs the correlation service for the appropriate step and compares the correlation ID to the correlation IDs in the WMPRTXREF table.

- If the ID matches an ID for a running process, the Process Engine loads the external document content into the appropriate tables in the Process Engine database component and the server runs the step as part of the running process.
- If the ID does not match an ID for a running process, the Process Engine creates a process instance ID, stores the process instance ID and the correlation ID in the WMPRTXREF table, and starts a new process.

Process Logging Behavior

The Process Engine provides the following types of logging:

- Process Engine logging. For more information, see [“About Process Engine Logging” on page 169](#).
- Process instance logging. When implemented, this is a subset of Process Engine logging. For more information, see [“About Process Instance Diagnostic Logging” on page 170](#).
- Process audit logging. For more information, see [“About Process Audit Logging” on page 171](#).
- Step input and output field logging. For more information, see [“Log Inputs and Outputs” on page 93](#).

Note:

You can also collect a package of comprehensive process information for troubleshooting purposes. For more information, see the chapter [“Collecting Process Troubleshooting Information”](#) in the PDF publication *Administering webMethods Process Engine*.

About Process Engine Logging

During run time, the Process Engine transmits log messages to the Integration Server `server.log` file (sometimes referred to as the *journal log*), located by default in the folder `Software AG_directory/IntegrationServer/serverName/instances/instance_name/logs`. A new file is created for each date of operation, with the date appended to the file name, for example, `server.log.20120815`.

You may find the contents of the `server.log` file useful for troubleshooting or debugging purposes. The log levels listed in the following table are available:

Level	Highest message type written to <code>server.log</code> , plus all lower messages
Fatal	Failure that likely affects other operations or products.
Error	Failure that does not likely affect other operations or products.
Warn	Problems that do not end operations, or unexpected or unusual conditions.
Info	Success of an event (this is the default setting).

Level	Highest message type written to server.log, plus all lower messages
Debug	Code-level statements recording unusual conditions or decisions.
Trace	Code-level statements recording program flow and state.
Off	No information is written to the server log.

- To change the Process Engine server log level, see [“Changing the Process Engine Logging Levels”](#) on page 170.
- For detailed information about Integration Server logging, see Chapter 8, *Setting Up the Server Log*, in the PDF publication *webMethods Integration Server Administrator’s Guide*.

In addition to Process Engine-related messages, the server.log file also contains messages generated by individual process instances. However, you can enable process instance logging for each process model so that messages from instances of that model are sent to a separate log file. For more information, see [“About Process Instance Diagnostic Logging”](#) on page 170.

Changing the Process Engine Logging Levels

➤ To change the Process Engine logging levels

1. In Integration Server Administrator: **Settings > Logging**.
2. In the Logger List, click **Server Logger**.
3. Click **Edit Server Logger**.

Note:

For a list of the available logging levels, see [“About Process Engine Logging”](#) on page 74.

4. Expand the **WmPRT Package** entry and do one or both of the following:
 - Change the Process Engine server log level by selecting a new log level for the **0101 General** entry.
 - Change the process instance log level by selecting a new log level for the **0102 Process Execution** entry.
5. Click **Save Changes**.

About Process Instance Diagnostic Logging

Log messages from individual process instances are always sent to the server.log file, as described in [“About Process Engine Logging”](#) on page 169. However, these messages are mixed in with Process Engine messages as well as messages from other process instances, and it can be hard to find the specific messages you are looking for.

The Process Engine also supports process instance diagnostic logging for individual process models. When you enable process instance diagnostic logging for a process model, the process instance log messages are sent to both the `server.log` file and to a separate process instance log file. You can then access the process instance diagnostic log file to see the messages from an individual process instance.

For more information, see these topics:

- “Enabling and Disabling Process Instance Diagnostic Logging” in Chapter 5 of the PDF publication, *webMethods Monitor User’s Guide*.
- [“Viewing a Process Instance Diagnostic Log File” on page 171.](#)

Viewing a Process Instance Diagnostic Log File

When a process model is enabled for process instance diagnostic logging, a log file is created in the directory *Software AG_directory/IntegrationServer/serverName/instances/instance_name/packages/WmPRT/log*, with a file name of *processInstanceID.log*. If the process is running on multiple servers (for example, in a distributed environment), a diagnostic log file is created *on each server that runs the process*. In this case, you must view all of the instance logs to get a complete picture of the instance activity. The log file name is the same on each server.

You can view the file in any text editor, or you can dynamically monitor the file in a command session using the `tail` command. The `tail` command is available on all Linux and UNIX systems, and on some Windows systems. If your Windows system does not offer the `tail` command, you can download it from the following locations:

- As part of the [Windows Server 2003 Resource Kit Tools](#) available from Microsoft.
- As an executable from [Sourceforge](#).

Message entries are formatted as follows:

```
[timestamp messageID threadID] processInstanceID:iteration stepID message
```

All messages that are not pertinent to the process instance (for example, correlation of an incoming document) are sent to the `server.log` file.

About Process Audit Logging

The Process Engine sends audit logging data to the Process Audit Log database component.

You determine the amount of data captured by audit logging by specifying the quality of service setting **Minimum Logging Level** for each process model, as described in [“Setting Quality of Service for a Process” on page 111.](#)

webMethods Monitor uses the information captured by audit logging:

- For more information, see [“Process Audit Log Database Component Tables” on page 172.](#)

- For complete information about webMethods Monitor, see the PDF publication *webMethods Monitor User's Guide*.

A Process Engine does not write process logging data for the wrapper services that actually call the services for process steps.

Process Audit Log Database Component Tables

The Process Engine writes process logging data to the Process Audit Log database component tables shown below. webMethods Monitor uses the information in these tables. For complete information about webMethods Monitor, see the PDF publication *webMethods Monitor User's Guide*.

You specify the minimum level of logged data with the quality of service setting **Minimum Logging Level** for each process model, as described in [“Setting Quality of Service for a Process” on page 111](#).

The following table describes the Process Audit Log database component tables.

Table	Description
WMPROCESS DEFINITION	When a webMethods Monitor user changes a process model version's status, the Process Engine writes a row containing a flag that indicates whether the model version is enabled or disabled.
PRA_PROCESS_ CUSTOM	Each time a process starts running, the Process Engine logs a row containing the process instance ID. If you specified a custom ID for a process using the service “pub.prt.log:logCustomID” on page 197 , the Process Engine logs a row containing the custom ID.
PRA_PROCESS	If you are logging process statuses, the Process Engine logs a row for each process status.
PRA_PROCESS_ STEP	If you are logging step statuses, the Process Engine logs a row for each step status. Note: When host or port is changed, the SERVERID column in the PRA_PROCESS_STEP table must be updated manually.
PRA_PROCESS_ STEP_LOOP	If you are logging step statuses, the Process Engine logs a row for each step status, containing the process instance iteration, step iteration, and the loop iteration.
PRA_STEP_ TRANSITION	If you are logging transitions, the Process Engine logs a row for each transition between steps; that is, it logs the step ID for the beginning and ending step of each transition.
PRA_STEP_LOOP_ LOGGED_FIELD	If you are logging step statuses, the Process Engine logs a row for each looping step with its loop iteration (in addition to process instance iteration and step iteration). Logging of custom data and custom looped data are enabled by the values defined on the Logged Fields page in the

Table	Description
	Properties view in Designer and are not related to the logging level set for the process.
PRA_STEP_LOGGED_FIELD	If you are logging run-time values for specified input or output document fields for process steps, the Process Engine logs a row for each value.
PRA_ERROR	If a step fails, the Process Engine logs a row for the error.
WMSERVICE	If a step calls a service and the service is set for logging, the Integration Server logs the service data.
PRA_STEP_MESSAGE	If a step calls a service and you are logging user-defined messages for that service using the service “pub.prt.log:logActivityMessages” on page 196 , the Process Engine logs a row for each message.
PRA_STEP_LOGGED_FIELDS	If you have marked specific step inputs and outputs for logging, this information is stored in this table. The information logged here is not subject to the Minimum Logging Level setting of the process model, but is set on the process model’s Logged Fields page in the Properties view at design-time.

About Expression Operators

You can define the following conditional behavior for a step:

- Standard looping.
- An If Condition transition.
- A document subscription filter.

These conditions are specified with a conditional expression you create on the appropriate page of the Properties view.

The following operators are available when configuring these conditional expressions. Not all operators are available for all fields. For example, if a numerical field is specified in the **Field Name** selection, only numerical operators are available.

The table below lists and describes the operators.

Operator	Description	Operator	Description
=	Equals the specified comparison value	starts with	Starts with the specified comparison value
!=	Does not equal the specified comparison value	does not start with	Does not start with the specified comparison value

Operator	Description	Operator	Description
>	Is greater than the specified comparison value	ends with	Ends with the specified comparison value
>=	Is greater than or equals the specified comparison value	does not end with	Does not end with the specified comparison value
<	Is less than the specified comparison value	exists	The specified comparison value is present
<=	Is less than or equals the specified comparison value	does not exist	The specified comparison value is not present
contains	Contains the specified comparison value		
does not contain	Does not contain the specified comparison value		

11 Process Engine Services

■ Process Engine Built-In Services Location	176
■ Summary of Elements in the WmPRT\pub Folder	176

Process Engine Built-In Services Location

The built-in services in this chapter are installed on the Integration Server as part of the WmPRT package. These Java services can be found in the indicated folder location in the Package Navigator view of Designer, or in the Package Management link in the Integration Server Administrator.

This chapter describes the services and supporting elements found in the \pub folder. You can use these services as templates to create services in Designer that perform a wide variety of actions on the services running in the Process Engine on the connected Integration Server.

For additional information about working with services in Designer, see *webMethods Service Development Help*.

Summary of Elements in the WmPRT\pub Folder

The available elements in this folder are listed in the following table:

Element	Package and Description
pub.prt:CorrelationService	WmPRT. Specification that describes the inputs and outputs required for a correlation service.
pub.prt.admin:changeProcessStatus	WmPRT. Changes the process status by broadcasting a request to all servers participating in the process.
pub.prt.admin:deleteProcess	WmPRT. Deletes information for a specific process instance from the Process Engine database.
pub.prt.admin:resumeProcesses	WmPRT. Service that allows users to resume processing for one or all previously suspended process models.
pub.prt.admin:scanPackage	WmPRT. Tells the Process Engine to scan a specified package for new or updated process model version fragments and use these fragments to update its internal model index.
pub.prt.admin:suspendProcesses	WmPRT. Service that allows users to suspend processing for any or all available process models.
pub.prt.audit.truncateProcessAtRest	WmPRT. Truncates the Process Audit table WMPROCESSATREST. You can schedule this service to run on an interval of your choosing using the Integration Server Scheduler.

Element	Package and Description
pub.prt.correlate:deleteCorrelation	WmPRT. Deletes all mappings between the specified process instance ID and any correlation IDs or conversation IDs.
pub.prt.correlate:establishCorrelation	WmPRT. Sets up a correlation between a correlation ID and a process ID or between a conversation ID (for a Trading Networks document) and a process ID.
pub.prt.correlate:lookupCorrelation	WmPRT. Returns the process instance ID that is associated with the specified correlation ID or conversation ID. If no association exists, creates a new process instance ID and mapping.
pub.prt.debugger:cleanupDebuggerTables	WmPRT. Cleans up process debugger database tables by deleting records before a given timestamp.
pub.prt:ErrorService	WmPRT. This specification has been deprecated and should no longer be used.
pub.prt.ExceptionTransitionInfo	WmPRT. This document type describes the information passed in the pipeline from a step when that step has taken one of the various Error Transitions. This document type is provided as a convenience to the Designer user to map any or all of the fields described in this document type.
pub.prt.jms:send	WmPRT. Sends a JMS message. This service encodes an IS Document into a JMS message and sends it to the specified destination using the specified options. The main difference between this service and the <code>pub.jms:send</code> service in WmPublic is that this service allows the user to easily specify the type of the document, which is required by the Process Engine to kick off a process instance, as well as make it convenient to format the JMS message appropriately for use with the Process Engine.
pub.prt.log:logActivityMessages	WmPRT. Logs process activity messages to the IS Core Audit Log database.
pub.prt.log:logCustomID	WmPRT. This service associates a "friendly name" (the customID) with a Process Instance identifier. This friendly name can be used to search for the process instance in monitor.

Element	Package and Description
pub.prt:ProcessData	WmPRT. Document type that describes the structure of the <i>ProcessData</i> section of the pipeline for a process.
pub.prt.SubprocessModel	WmPRT. Document type that describes the information needed to invoke a dynamic referenced process.
pub.prt.timer.process:cancel	WmPRT. This service cancels the process timer for the specified process instance.
pub.prt.timer.process:create	WmPRT. This service creates a process timer for the specified process instance.
pub.prt.timer.process:createWithBusinessCalendar	WmPRT. This service creates a process timer for the specified process instance and the specified business calendar.
pub.prt.timer.process:createWithDate	WmPRT. This service creates a process timer for the specified process instance and the specified date.
pub.prt.timer.process:get	WmPRT. This service returns the actual date that the timer will expire for the specified process instance.
pub.prt.tn:deleteByCID	WmPRT. Deletes a process instance associated with a given conversation ID.
pub.prt.tn:getPIDforCID	WmPRT. Returns the process instance ID for a given conversation ID.
pub.prt.tn:getRoleInfo	WmPRT. Fetches role information for a specified role in process.
pub.prt.tn:handleBizDoc	WmPRT. Sends a Trading Networks BizDocEnvelope (Trading Networks document) to the Process Engine, to allow the document to be processed as part of a business process.
pub.prt.tn:mapCIDtoPID	WmPRT. Sets up a mapping between the specified conversation ID and process instance ID.
pub.prt.tn:MatchBizDoc	WmPRT. Matches the supplied business document ID to a valid process model. This service provides support for webMethods Rosetta Net in Process Engine.

Element	Package and Description
pub.prt.tn:RoleInfo	WmPRT. Document type that describes information maintained for roles in a process.

pub.prt:CorrelationService

WmPRT. Specification that describes the inputs and outputs required for a correlation service.

A correlation service is associated with one or more receive steps in a process model version. The Process Engine uses the correlation service associated with a step to route IS documents published as inputs into that step to a running instance of the model, where appropriate. For more about correlation services, see "Correlation Services" in the *webMethods BPM Process Development Help*.

Input Parameters

The following table lists the input parameters.

<i>ProcessModelID</i>	String ID of the process model with which this invocation of the correlation service is involved.
<i>ProcessModelVersion</i>	String Version of the process model with which this invocation of the correlation service is involved.
	<p>Note: Because a single correlation service can be associated with steps from more than one process model version, you can use the <i>ProcessModelID</i> and <i>ProcessModelVersion</i> to identify the process model version using the correlation service at run time.</p>
<i>LogicalServer</i>	<p>String Name of the logical server that is associated with the step in the process model version with which this invocation of the correlation service is involved. In other words, the name of the logical server on which this correlation service is running.</p> <p>Because a single correlation service can be used with steps that execute on different servers, you can use <i>LogicalServer</i> to identify a specific server at run time.</p>
<i>ProcessStepID</i>	<p>String ID of the step in the process model version with which this invocation of the correlation service is involved (for example, N3).</p> <p>Because a single correlation service can be associated with multiple steps in a process model version, you can use <i>ProcessStepID</i> to identify the specific step at run time.</p>
<i>DocumentName</i>	String Name of this document as used in the process model version (for example, "OrderDocument").

<i>DocumentType</i>	String Name of this document type (for example, "orders.sap:OrderDocument").
<i>Document</i>	Document The document.

Output Parameters

The following table lists the output parameters.

<i>ProcessCorrelationID</i>	String Conditional. An abstract ID that correlates to the actual process instance ID of the running process. For example: "CUSTOMER-0003456977::ORDER-19477593-AR9-1000". All documents bound for the same instance of the process must return the same correlation ID. Similarly, correlation IDs must be unique across all process instances.
<i>CorrelateAsTN</i>	String Conditional. Flag that indicates whether the correlation ID in <i>ProcessCorrelationID</i> is a conversation ID. The following values apply: <ul style="list-style-type: none">■ <code>true</code> — Indicates that <i>ProcessCorrelationID</i> is a Trading Networks conversation ID.■ <code>false</code> — Default. Indicates that <i>ProcessCorrelationID</i> is not a Trading Networks conversation ID.

pub.prt.admin:changeProcessStatus

WmPRT. Changes the process status by broadcasting a request to all servers participating in the process.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String ID of the process instance that you want to change the status of.
<i>ProcessIteration</i>	String The iteration of the process instance that you want to change the status of.
<i>ProcessModelID</i>	String Optional. ID of the process model (ModelID) associated with the process you want to change the status of.
<i>ProcessModelVersion</i>	String Optional. Version of the process model to change the status of.
<i>Action</i>	String Process action that is to be applied to the specified process instance. The following values apply: <ul style="list-style-type: none">■ <code>SUSPEND</code>. Causes a process instance to temporarily stop executing.

- RESUME. Causes a suspended process to continue executing.
- CANCEL. Causes a process instance to terminate without an error condition.
- FAIL. Causes a process instance to terminate with an error condition.

EscalateFailure

String Conditional. Flag that indicates whether the parent process takes control of a failure in a referenced process. The following values apply:

- `true` — Default. The parent process receives notification of the failure from a referenced process and continues executing. The failed referenced process cannot be resubmitted as the parent process is no longer waiting for a response.
- `false` — The parent process does not receive notification of the failure from a referenced process and continues to wait for a response; this enables the referenced process to be resubmitted for another attempt at normal process completion. See the Usage Notes for more information.

Output Parameters

None.

Note:

If this service runs to completion, it means that the request for a status change has been made. The servers involved handle these requests asynchronously. You can track the status of a process using webMethods Monitor.

Usage Notes

The following table shows all valid status change combinations. All combinations that are not listed in the table are invalid.

Status	Action	New Status
Started	SUSPEND	Suspended
Started	FAIL	Failed
Started	CANCEL	Canceled
Suspended	FAIL	Failed
Suspended	CANCEL	Canceled
Suspended	RESUME	Resumed/Started
Completed	FAIL	Failed

Status	Action	New Status
Failed	CANCEL	Canceled

The *EscalateFailure* parameter enables the developer to determine if a referenced process is recoverable by resubmittal after an error occurs. Prior to version 8.0 SP1, the failure of a referenced process was always escalated to the parent process. In this case, the parent process would continue execution, and the failed referenced process cannot be successfully resubmitted because the parent process is no longer waiting for a response.

For backward compatibility, the *EscalateFailure* parameter is set to true by default, replicating this behavior. However, when [pub.prt.admin:changeProcessStatus](#) is invoked and the *EscalateFailure* parameter is set to false, the failure of the referenced process is not escalated to the parent process, and the parent process continues to wait for a response. This allows the failed referenced process to be resubmitted, and upon successful completion, the referenced process will rejoin the parent process.

To implement this scenario, you must design the referenced process so that it will call [pub.prt.admin:changeProcessStatus](#) (on a terminate step, for example) with the *EscalateFailure* parameter set to false. Then, if the referenced process fails, the parent process will remain as "started", and you can then resubmit the referenced process. If the referenced process fails again, the same behavior occurs. If the referenced process completes, it will rejoin the parent instance.

A referenced process failure that occurs by a cause other than the `changeProcessStatus` service (for example, an error occurs and there is no error handler step) is not affected by this parameter and will continue to escalate the failure to the parent instance.

Note:

To be able to resubmit the failed referenced process, you must enable the failed step for resubmission in webMethods Monitor. For more information, see the *webMethods Monitor User's Guide*.

pub.prt.admin:deleteProcess

WmPRT. Deletes information for a specific process instance from the Process Engine database.

Use this service to delete the information for a specific process instance from the Process Engine database. For example, if that process that has become unresponsive or has entered an indeterminate state. The information is removed from the Process Engine database only. The Process Audit database remains unaffected, and the process instance will continue to appear in webMethods Monitor. If you want to retain the process instance information but remove mappings to correlation IDs, use the [pub.prt.correlate:deleteCorrelation](#) service.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String ID of the process instance for which you want to delete information.
--------------------------	--

ProcessIteration **String** The iteration of the process instance that you want to delete from the Process Engine database.

Output Parameters

The following table lists the output parameters.

success **String** Flag indicating whether the process instance information was deleted. The following values apply:

- `true` — The process instance information was deleted.
- `false` — The process instance information was not deleted.

Usage Notes

In normal operation, the Process Engine Storage Cleaner utility removes older process instance information from the Process Engine database at regularly scheduled intervals. You can set the Storage Cleaner intervals on the Settings page of the Process Engine home page in the Integration Server Administrator. For more information, see the topic “Configuring Process Engine Settings” in Chapter 4, “Configuring and Monitoring the Process Engine” . You can use the `pub.prt.admin:deleteProcess` service to remove this information outside of the regularly scheduled Storage Cleaner operation. Note that the Storage Cleaner removes process instance data for completed and failed instances that are eligible for deletion, and it also removes correlation data. For more information, see “About the Process Engine Storage Cleaner” in Chapter 1, “Concepts” .

See Also

“`pub.prt.tn:deleteByCID`” on page 203

`pub.prt.admin:resumeProcesses`

WmPRT. Service that allows users to resume processing for one or all previously suspended process models.

The service can be invoked either on a per-model basis or for all available models on a given Process Engine server. When this service is executed, the subscription triggers for the qualifying models are enabled, facilitating the creation of new instances. In addition, any previously suspended instances are set to status = RESUMED. In the case of a cluster or a distributed Process Engine set up, the node that the service is invoked on is considered primary and it broadcasts the resume action across all participating nodes that it is aware of.

Input Parameters

The following table lists the input parameters.

<i>ProcessModelID</i>	String Optional. This is the process key for the model that the user wants to resume. All versions of a given model are affected.
<i>resumeAll</i>	String Optional. Indicates whether or not the resume action will affect all models or only the model that is specified as the value for the parameter <i>ProcessModelID</i> . The following values apply: <ul style="list-style-type: none">■ <code>true</code> — The <i>ProcessModelID</i> value, if provided, is ignored and the resume action is applied to all available models, their corresponding process instances, and their subscription triggers.■ <code>false</code> — Default. The resume action is applied only to the process model defined in the parameter <i>ProcessModelID</i> (all process instances and subscription triggers).

Output Parameters

The following table lists the output parameters.

<i>message</i>	String Optional. Displays any exceptions encountered during the service call.
<i>success</i>	String Optional. Returns one of the following: <ul style="list-style-type: none">■ <code>error</code> — Indicates that one or more exceptions occurred during the service call.■ <code>true</code> — Indicates that the service call executed without any exceptions. It does <i>not</i> indicate that all internal tasks initiated by the service call have run to completion. To verify that the resume action is complete, refer to the Usage Notes section.

Usage Notes

This service is intended for use during scheduled maintenance periods that require processing to be paused for a period of time and then subsequently resumed. The service may be used to resume processing by either resuming a single process model or all available models on the system that were previously suspended.

The service affects all versions of targeted process models, and in the case of a cluster or distributed set up, the resume action is relayed to all participating nodes. To resume only a specific model, users can invoke this service and provide the appropriate value for the *ProcessModelID* input parameter. To resume all models on a given system, invoke the service by setting the value of the *resumeAll* parameter to `true`.

The service resumes subscription triggers for all targeted services. Any qualifying instances that were previously set to suspended by other means (for example, manually with webMethods Monitor) will be resumed, as the invocation of this service affects all qualifying instances without exception.

Note:

Be aware that the complementary `suspendProcesses` service suspends subscription triggers for all targeted process models; during the suspension period, any incoming documents directed to those models will be cached until such time as the model is resumed. Applying the `resumeProcesses` service will enable the subscription triggers for all targeted models, and all cached documents will be processed. This could cause a temporarily heavy load on the system.

This service may not work as expected for environments or process models that use volatile tracking. In this case, the resume feature relies solely on the Process Engine cache in RAM to determine the instances that will be affected. However, due to its dynamic nature, the Process Engine cache may be momentarily out-of-sync, especially following a server restart or a package re-load. In such cases, resuming at the instance-level using `webMethods Monitor` may be the only alternative.

The resume action is a series of independent tasks across all participating nodes. To verify that the resume action is indeed complete, the users are advised to use the following actions:

- Check the Integration Server error log to make sure there were no exceptions.
- Use `webMethods Monitor` to verify that the status of all the qualifying instances are set to `RESUMED`.
- Use the `webMethods Broker` (deprecated) trigger management page (**Settings > Messaging > Broker/Local Trigger Management**) in the IS Administrator interface to verify that all the qualifying subscription triggers are enabled.

See Also

“`pub.prt.admin:suspendProcesses`” on page 186

pub.prt.admin:scanPackage

`WmPRT`. Tells the Process Engine to scan a specified package for new or updated process model version fragments and use these fragments to update its internal model index.

This service updates the Process Engine index as follows:

- If a fragment file is new, `scanPackage` adds information from this file to the Process Engine index.
- If a fragment file is modified, `scanPackage` replaces existing index information to reflect the modifications.
- If a fragment file no longer exists, `scanPackage` deletes the corresponding index information.

Input Parameters

The following table lists the input parameters.

Package **String** Name of the package that you want to scan. If the named package does not exist on the Integration Server, information about any fragments previously loaded from that package will be deleted from the Process Engine model index.

Output Parameters

The following table lists the output parameters.

<i>ExistingFragments</i>	String List Conditional. Number of fragments contained in the package that were already known to the Process Engine.
<i>ModifiedFragments</i>	String List Conditional. Number of fragments contained in the package that had been modified since the Process Engine last read them.
<i>MissingFragments</i>	String List Conditional. Number of fragments contained in the package that have been deleted since the Process Engine last read them.
<i>NewFragments</i>	String List Conditional. Number of fragments contained in the package that are new since the Process Engine last scanned the named package.

Usage Notes

Use this service before or after replicating a package that contains Process Engine process model version fragments to accomplish the following:

- After unzipping a package onto a new server but before enabling it, invoke this service to force the Process Engine to pick up the new fragments.
- After disabling all related process model versions, zipping a package, and deleting the package from an old server, invoke this service to force the Process Engine to discard information about the old model version.

pub.prt.admin:suspendProcesses

WmPRT. Service that enables users to suspend processing for any or all available process models.

The service can be invoked either on a per-model basis or for all available models on a given Process Engine server. When this service is executed, the subscription triggers for the qualifying models are suspended and no new instances are created. In addition, any currently running instances are set to status = SUSPENDED. In the case of a cluster or a distributed Process Engine set up, the node that the service is invoked on is considered primary and it broadcasts the suspend action across all participating nodes that it is aware of.

Input Parameters

The following table lists the input parameters.

<i>ProcessModelID</i>	String Optional. This is the process key for the model that the user wants to suspend. All versions of a given model are affected.
<i>suspendAll</i>	<p>String Optional. Indicates whether or not the suspend action will affect all models or only the model that is specified as the value for the parameter <i>ProcessModelID</i>. The following values apply:</p> <ul style="list-style-type: none"> ■ <code>true</code> — The <i>ProcessModelID</i> value, if provided, is ignored and the suspend action is applied to all available models, their corresponding process instances, and their subscription triggers. ■ <code>false</code> — Default. The suspend action is applied only to the process model defined in the parameter <i>ProcessModelID</i> (all process instances and subscription triggers).

Output Parameters

The following table lists the output parameters.

<i>message</i>	String Optional. Displays any exceptions encountered during the service call.
<i>success</i>	<p>String Optional. Returns one of the following:</p> <ul style="list-style-type: none"> ■ <code>error</code> — Indicates that one or more exceptions occurred during the service call. ■ <code>true</code> — Indicates that the service call executed without any exceptions. It does <i>not</i> indicate that all internal tasks initiated by the service call have run to completion. To verify that the suspend action is complete, refer to the Usage Notes section.

Usage Notes

This service is intended for use during scheduled maintenance periods that require processing to be paused for some duration of time. The service may be used to suspend processing in bulk, impacting either a single process model or all available models on the system.

The service affects all versions of targeted process models, and in the case of a cluster or distributed set up, the suspend action is relayed to all participating nodes. To suspend only a specific model, users can invoke this service and provide the appropriate value for the *ProcessModelID* input parameter. To suspend all models on a given system, invoke the service by setting the value of the *suspendAll* parameter to `true`.

The service suspends subscription triggers for all targeted services. Note that when the triggers are suspended, the change to the trigger state is persisted through subsequent server restarts. Transition triggers are not suspended, and some preliminary transition trigger processing may occur as the result of incoming documents. Incoming documents directed to suspended models will be cached until such time as the model is resumed.

This service may not work as expected for environments or process models that use volatile tracking. In this case, the suspend feature relies solely on the Process Engine cache in RAM to determine the instances that will be affected. However, due to its dynamic nature, the Process Engine cache may be momentarily out-of-sync, especially following a server restart or a package re-load. In such cases, suspending at the instance-level using webMethods Monitor may be the only alternative.

The suspend action is a series of independent tasks across all participating nodes. To verify that the suspend action is indeed complete, the users are advised to use the following actions:

- Check the Integration Server error log to make sure there were no exceptions.
- Use webMethods Monitor to verify that the status of all the qualifying instances are set to SUSPENDED.
- Use the webMethods Broker (deprecated) trigger management page (**Settings > Messaging > Broker/Local Trigger Management**) in the IS Administrator interface to verify that all the qualifying subscription triggers are enabled.
- Use the IS Administrator Service Usage page (**Server > Service Usage**) to ensure that there are no currently running threads. This may happen if there are any long-running services that are waiting to finish.

See Also

“pub.prt.admin.resumeProcesses” on page 183

pub.prt.audit.truncateProcessAtRest

WmPRT. Truncates the Process Audit table WMPROCESSATREST. You can schedule this service to run on an interval of your choosing using the Integration Server Scheduler. For more information, see the topic “Scheduling Services” in the PDF publication *webMethods Integration Server Administrator’s Guide*.

This service is relevant for users who are implementing the optional database partitioning scripts for Process Audit data archiving. In that case, the ProcessAtRest table contains an entry for every completed process instance which, at some point, must be cleaned up. Use this service for that purpose only.

If you are not using database partitioning for Process Audit Archive, this service is not relevant as the table will contain 0 entries by definition.

Input Parameters

None.

Output Parameters

The following table lists the output parameters.

<i>success</i>	<p>String Flag indicating whether the table was truncated. The following values apply:</p> <ul style="list-style-type: none"> ■ <code>true</code> — The table was truncated. ■ <code>false</code> — No truncation occurred.
----------------	--

pub.prt.CallActivityModel

WmPRT. Document type that describes the information needed to dynamically invoke one or more callable processes.

Important:

This document type is used with callable processes only. If you are working with the deprecated ability to dynamically invoke a referenced processes from a call activity step, see [“pub.prt.SubprocessModel” on page 198](#) for more information.

Input Parameters

The following table lists the input parameters.

<i>WaitForCallActivity</i>	<p>String Flag indicating whether to wait for the child process (as done with statically invoked process), or to launch it asynchronously and not expect any return documents. Applies to all process instances started from the document type.</p> <ul style="list-style-type: none"> ■ <code>true</code> — Default. Wait for the child process. ■ <code>false</code> — Start the child process asynchronously and do not expect any return documents.
<i>CallActivityModelID</i>	<p>String The identifier of the callable process model in the format: Project/Process.</p>
<i>OutputPipelineName</i>	<p>String The name of the output pipeline to be returned back from each child process. No value is needed here if a value of false is used for <i>WaitForCallActivity</i>.</p>
<i>InputPipelines</i>	<p>Document list A list of instances that are to be started for the specified callable process model. For example, if you have 10 line items to process, there will be 10 documents in this list.</p>

Output Parameters

None.

Usage Notes

To enable successful execution of a dynamically invoked callable process, you must ensure that these values are in the process pipeline prior to any activity (for example, input data mapping) that applies to the call activity step.

Suppose you have specified *OutputPipelineName* and a *WaitForCallActivity* value of `true`. In this case, the Process Engine waits for all instances of the child process to complete and populates the pipeline with data from each child process instance, obtained from the returned document(s). This data is now available for use.

In another example, suppose that three instances of the child process *LineItem* are started, and each of these instances is expected to return an output pipeline of the name *LineItemPrice*. In this case, the *LineItemPrice* data is added to the pipeline and the data in it is available for use. The order of the documents in the list is the same as the order of invocation. This enables easy location of the data in the pipeline by using the *OutputPipelineName* value.

pub.prt.correlate:deleteCorrelation

WmPRT. Deletes all mappings between the specified process instance ID and any correlation IDs or conversation IDs.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID for the mapping(s) you want to delete.
--------------------------	--

Output Parameters

The following table lists the output parameters.

<i>success</i>	String Flag indicating whether any mappings were deleted. The following values apply: <ul style="list-style-type: none">■ <code>true</code> — One or more mappings were deleted.■ <code>false</code> — No mappings were deleted.
----------------	--

Usage Notes

Use this service with care. Deleting correlation mappings for running process instances could have unpredictable results.

pub.prt.correlate:establishCorrelation

WmPRT. Sets up a correlation between a correlation ID and a process ID or between a conversation ID (for a Trading Networks document) and a process ID.

Input Parameters

The following table lists the input parameters.

<i>ProcessCorrelationID</i>	String The correlation ID or conversation ID that you want to map to the specified process instance ID.
<i>ProcessInstanceID</i>	String Process instance ID that you want to map to the specified correlation ID or conversation ID.
<i>MappingType</i>	String Optional. Flag indicating the type of ID that you supplied. <ul style="list-style-type: none"> ■ IS — Default. This is a correlation ID. ■ TN — This is a Trading Networks conversation ID.

Output Parameters

The following table lists the output parameters.

<i>success</i>	String Flag indicating whether the mapping took place. The following values apply: <ul style="list-style-type: none"> ■ true — The mapping was successfully established. ■ false — The mapping could not be established.
----------------	---

Usage Notes

The Process Engine automatically establishes these mappings when Trading Networks or IS documents are used to start processes, or when a Trading Networks document is output from a running process. Use this service when there is no way for the Process Engine to determine the mapping itself (for example, when a process starts with an IS document and then waits for a Trading Networks document).

Use this service with care. Be sure to create correct mappings; an invalid mapping could prevent other processes from completing successfully.

pub.prt.correlate:lookupCorrelation

WmPRT. Returns the process instance ID that is associated with the specified correlation ID or conversation ID. If no association exists, creates a new process instance ID and mapping.

Input Parameters

The following table lists the input parameters.

<i>ProcessCorrelationID</i>	String Correlation ID or conversation ID for which you want to return the process instance ID.
<i>MappingType</i>	String Optional. Flag indicating whether <i>ProcessCorrelationID</i> specifies a correlation ID or a conversation ID. The following values apply: <ul style="list-style-type: none">■ IS — Default. <i>ProcessCorrelationID</i> is a correlation ID for an IS document.■ TN — <i>ProcessCorrelationID</i> is a Trading Networks conversation ID for a Trading Networks document.

Output Parameters

The following table lists the output parameters.

<i>ProcessInstanceID</i>	String Conditional. The process instance ID mapped to the specified correlation ID or conversation ID (if any).
<i>success</i>	String Flag indicating whether a process instance ID was returned. The following values apply: <ul style="list-style-type: none">■ true — A process instance ID was found or created and returned.■ false — A process instance ID was not returned because the correlation ID or conversation ID was not established.

Usage Notes

Use this service to check on mappings that were established with previous calls to [pub.prt.correlate:establishCorrelation](#) or, under certain circumstances, to check on the existence of a process with a particular correlation ID or conversation ID.

See Also

“[pub.prt.correlate:establishCorrelation](#)” on page 191

pub.prt.debugger:cleanupDebuggerTables

WmPRT. This service cleans up process debugger database tables by deleting records before a given timestamp.

Process Debug database records are deleted when the session ends or a new session is started. However, sometimes in the case of failures, such as the package being reloaded during debugging,

data can be left in the database tables. This service can be used to delete all old records based on a timestamp provided by the user.

Input Parameters

The following table lists the input parameters.

<i>beforeDate</i>	String Optional. Date prior to which to delete Process Debug records. For example, 05/10/12. The <i>beforeDate</i> parameter requires the use of the <i>pattern</i> parameter. The current timestamp is used to populate this field if no value is specified. This results in the deletion of all Process Debug records created prior to running this service.
<i>pattern</i>	String Optional. Format of <i>beforeDate</i> value. For example, dd/MM/yy. The <i>pattern</i> parameter requires the use of the <i>beforeDate</i> parameter. This service uses the Java SimpleDateFormat class to parse dates.
<i>daysBefore</i>	String Optional. Number of days prior to today. For example, 4.

Output Parameters

None.

Usage Notes

This service can be scheduled using a wrapper service or run manually. Do not run this service while debug sessions are active. Allow any active debug sessions to complete before running this service.

If the service cannot parse the date, it does not delete any records, and displays the following message: Parameters supplied could not be formed into a date for deletion.

The service displays a message upon successful deletion of records: Database tables were deleted before *date* . The date is localized and uses the following format: MM dd yyyy HH:mm:ss a. For example: Aug 12, 2012 12:00:00 AM.

pub.prt:ErrorService

WmPRT. This specification has been deprecated and should no longer be used.

pub.prt.ExceptionTransitionInfo

WmPRT. This document type describes the information passed in the pipeline from a step when that step has taken one of the various Error Transitions. This document type is provided as a convenience to the Designer user to map any or all of the fields described in this document type.

Input Parameters

The following table lists the input parameters.

<i>ErrorMessage</i>	String The error message describing the reason for the error transition.
<i>SourceStepID</i>	String The Step ID of the step that encountered the error
<i>SourceStepIteration</i>	String The Step iteration count of the step that encountered the error
<i>ExceptionType</i>	String One of the following: Cancel, UnsatisfiedJoin, RetriesExceeded, ProcessTimeout, JoinTimeout, StepTimeout, or StepError

Output Parameters

None.

pub.prt.jms:send

WmPRT. Sends a JMS message. This service encodes an IS Document into a JMS message and sends it to the specified destination using the specified options. The main difference between this service and the `pub.jms:send` service in WmPublic is that this service allows the user to easily specify the type of the document, which is required by the Process Engine to kick off a process instance, as well as make it convenient to format the JMS message appropriately for use with the Process Engine.

Input Parameters

The following table lists the input parameters.

<i>connectionAliasName</i>	String Set this value to define the connection alias you want to use. The default <code>PE_NONTRANSACTIONAL_ALIAS</code> is used if no value is specified.
<i>destinationName</i>	String Set this parameter to the value of the "Destination Name" in the Subscription trigger generated by Designer for the process model that this JMS message should start. It will be similar to "YourProjectName_YourProcessName_SUBQUEUE".
<i>destinationType</i>	String Set this parameter to the value of the "Destination Type" in the Subscription trigger generated by Designer for the process model that this JMS message should start. It will usually be "QUEUE".
<i>deliveryMode</i>	String Set this to <code>PERSISTENT</code> or <code>NON_PERSISTENT</code> . This field is mapped directly to the <code>JMSMessage.header.deliveryMode</code> field in the <code>pub.jms:send</code> service in WmPublic. For more information about the <code>pub.jms:send</code> service see the <i>webMethods Integration Server Built-In Services Reference</i> .

<i>priority</i>	String This field is mapped directly to the <code>JMSMessage.header.priority</code> field in the <code>pub.jms:send</code> service in <code>WmPublic</code> . For more information about the <code>pub.jms:send</code> service see the <i>webMethods Integration Server Built-In Services Reference</i> .
<i>timeToLive</i>	String This field is mapped directly to the <code>JMSMessage.header.timeToLive</code> field in the <code>pub.jms:send</code> service in <code>WmPublic</code> . Refer to the <i>webMethods Integration Server Built-In Services Reference</i> for more information about that field.
<i>data</i>	Document Set this parameter to the IS Document that should be send in the body of the JMS message. This will be the document that actually kicks off the process instance.
<i>documentType</i>	String Set this parameter to the fully qualified name of the document type that was used for the "data" parameter. This will be the same document type that was specified in Designer as the Receive Document type.
<i>useCSQ</i>	String Set this field to <code>false</code> if guaranteed transitions are required. Otherwise set it to "true" to utilize client-side queuing.

Output Parameters

The following table lists the output parameters.

<i>JMSTimestamp</i>	String Indicates when the JMS provider sent the message
<i>JMSMessageID</i>	String Handle to lock object and not drop from pipeline.

Usage Notes:

This service is used to initiate a process instance. If you do not specify a connection alias for the `connectionAliasName` parameter, the JMS connection alias `PE_NONTRANSACTIONAL_ALIAS` is used by default to connect to a JMS provider and send the JMS message. You can specify a different connection alias with the `connectionAliasName` parameter. The connection alias must exist and be properly configured by an IS administrator.

This service is a thin wrapper on top of the `pub.jms:send` service in `WmPublic`, but that service may also be used to initiate a process instance. Most of the parameters specified above map directly to similarly named parameters in the `WmPublic` service, with the following exception:

- The `pub.jms:send` service has no dedicated input parameter with which to set the `documentType` value that is required by the Process Engine to correctly map a JMS message to the correct process model. Instead, callers of `pub.jms:send` must instantiate a `documentType` field in the `properties` document, and set the value of that field to the fully qualified name of the IS document expected by the desired process model.

For more information about the `pub.jms:send` service see the *webMethods Integration Server Built-In Services Reference*.

pub.prt.log:logActivityMessages

WmPRT. Logs process activity messages to the IS Core Audit Log database.

Note:

This service does not reference the logging level set by the user in webMethods Monitor, so all pertinent information is logged regardless of the logging level setting.

Input Parameters

The following table lists the input parameters.

<i>FullMessage</i>	String Optional. Complete message to record in the IS Core Audit Log database. The message can be up to 1024 bytes.
<i>BriefMessage</i>	String Optional. Shortened version of the full message. The message can be up to 240 bytes.
<i>EntryType</i>	String Flag indicating the type of message. The following values apply: <ul style="list-style-type: none">■ Message — Indicates that the message is informational and no action is needed.■ Warning — Indicates that the message is a warning message. The process can complete successfully even if the circumstance causing the warning is not addressed.■ Error — Default. Indicates that the message is an error message. The process cannot complete successfully until the circumstance causing the error is resolved.

Output Parameters

None.

Usage Notes

This service can be added either to the flow service generated for a process step or to services called within that step. The service logs the input parameters to the `PRA_STEP_MESSAGE` log file in the IS Core Audit Log database.

Logged activity messages can be viewed in webMethods Monitor on the **Process Instance Status** and **Service Details** pages. For more information about viewing activity messages in webMethods Monitor, see the webMethods Monitor documentation.

pub.prt.log:logCustomID

WmPRT. This service associates a "friendly name" (the customID) with a Process Instance identifier. This friendly name can be used to search for the process instance in webMethods Monitor.

Note:

This service does not reference the logging level set by the user in webMethods Monitor, so all pertinent information is logged regardless of the logging level setting.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process Instance ID to be associated with the customID.
<i>customID</i>	String The "friendly name" of the Process Instance you wish to associate with the ProcessInstanceID.

Note:

The use of the characters "&" and "=" are restricted in this parameter. For example, if you create a custom ID with a format of `<fieldname1>=<valuenam1>` or `<fieldname1>=<valuenam1>&<fieldname2>=<valuenam2>`, then Monitor will create a column for each field name and display the value of the value name in that column.

Output Parameters

None.

pub.prt:ProcessData

WmPRT. Document type that describes the structure of the *ProcessData* section of the pipeline for a process.

This pipeline data is automatically filled in by the Process Engine for every step of a process. The service for that step is then executed with this data.

Parameters

The following table lists the parameters.

<i>ProcessInstanceID</i>	String Process instance ID of the running process.
<i>ProcessIteration</i>	String Number of times the process has been restarted (that is, iteration count).

<i>ProcessModelID</i>	String ID of the process model used by the running process.
<i>ProcessModelVersion</i>	String Version of the process model used by the running process.
<i>ProcessStepID</i>	String ID of the running step in the process.
<i>LogicalServer</i>	String Name of the logical server on which the running step was assigned and is executing.
<i>TryCount</i>	String The current iteration of the step.
<i>LoopCounter</i>	String The number of completed loops. After each loop is executed, the value of this field is incremented.
<i>AuditContext</i>	String
<i>Roles</i>	Document Conditional. If this process involves Trading Networks, this will contain information about the roles in the process. The key will be the role name, and the value will be an instance of the pub.prt.tn:RoleInfo IS document type.

See Also

“[pub.prt.tn:RoleInfo](#)” on page 208

pub.prt.SubprocessModel

WmPRT. Document type that describes the information needed to dynamically invoke a referenced process.

Important:

This document type is used with the deprecated ability to dynamically invoke a referenced processes from a call activity step. If you are working with dynamically invoked callable processes, see “[pub.prt.SubprocessModel](#)” on page 198 for more information.

Input Parameters

The following table lists the input parameters.

<i>WaitForSubprocess</i>	String Flag indicating whether to wait for the child process (as done with statically invoked referenced processes) or to launch it asynchronously and not expect any return documents. Applies to all process instances started from the document type. <ul style="list-style-type: none">■ <code>true</code> — Default. Wait for the child process.■ <code>false</code> — Start the child process asynchronously and do not expect any return documents.
--------------------------	--

<i>SubprocessModelID</i>	String The identifier of the referenced process model in the format: Project/Process.
<i>ReturnDocuments</i>	String List A list of document types that the parent process expects back from each child process. No value is needed here if a value of false is used for <i>WaitForSubprocess</i> .
<i>SubprocessInstances</i>	Document list A list of instances that are to be started for the specified referenced process model. For example, if you have 10 line items to process, there will be 10 documents in this list.

The following table describes the **Document list**.

Key	Description
<i>Inputs</i>	Document list A list of document types that are needed to invoke the specified referenced process model. For example, if your model requires a LineItem document and a Customer document, you will need two entries in this list for that instance.

The following table describes the **String** and the **Document**.

Key	Description
<i>Type</i>	String The fully qualified type of the document (for example, "MyPackage.docs:MyDocumentType").
<i>Document</i>	Document An input document for the given subprocess instance, as defined by the <i>Type</i> key.

Output Parameters

None.

Usage Notes

To enable successful execution of the dynamically invoked referenced process, you must ensure that these values are in the process pipeline prior to any activity (for example, input data mapping) that applies to the referenced step.

Suppose you have specified *ReturnDocuments* and a *WaitForSubprocess* value of `true`. In this case, the Process Engine waits for all instances of the child process to complete and populates the pipeline with data from each child process instance, obtained from the returned document(s). This data is now available for use.

In another example, suppose that three instances of a `LineItem` child process are started, and each of these instances is expected to return a `LineItemPrice` document. In this case, a document list named `LineItemPrice` is added to the pipeline and the data in it is available for use; the order of the documents in the list is the same as the order of invocation. This enables easy location of the data in the pipeline by using the return document name.

pub.prt.timer.process:cancel

WmPRT. This service cancels the process timer for the specified process instance.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID of the process timer you want to cancel.
--------------------------	--

Output Parameters

None.

pub.prt.timer.process:create

WmPRT. This service creates a process timer for the specified process instance.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID of the process for which you are creating the timer.
--------------------------	--

<i>BaseDate</i>	String The basis for the process timer. The following values apply: <ul style="list-style-type: none">■ <code>InstanceStart</code> — Sets the timer relative to the start time of the process instance.■ <code>currentTime</code> — Sets the timer relative to the current time.■ <code>ExistingTimeout</code> — Sets the timer relative to the expiration time of the current timer that exists for the process instance.
-----------------	---

<i>Days</i>	String The number of days to be applied to the timer expressed as a whole number.
-------------	--

<i>Hours</i>	String The number of hours to be applied to the timer expressed as a whole number.
--------------	---

<i>Minutes</i>	String The number of minutes to be applied to the timer expressed as a whole number.
<i>Seconds</i>	String The number of seconds to be applied to the timer expressed as a whole number.
<i>Milliseconds</i>	String The number of milliseconds to be applied to the timer expressed as a whole number.

Output Parameters

None.

Usage Notes

The timer is expressed in terms of days, hours, minutes, seconds, and milliseconds. A blank value is assumed to be 0 (zero). If a *BaseDate* of `ExistingTimeout` is specified and there is no existing timer for the process, the current time will be used as the basis for the timer (same as *BaseDate* of `CurrentTime`).

pub.prt.timer.process:createWithBusinessCalendar

WmPRT. This service creates a process timer for the specified process instance and the specified business calendar.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID of the process for which you are creating the timer.
<i>BaseDate</i>	<p>String The timer start date, defined by entering one of the following string values:</p> <ul style="list-style-type: none"> ■ <code>InstanceStart</code> — Sets the timer relative to the start time of the process instance. ■ <code>CurrentTime</code> — Sets the timer relative to the current time. ■ <code>ExistingTimeout</code> — Sets the timer relative to the expiration time of the current timer that exists for the Process Instance. If <code>ExistingTimer</code> is specified and there is no existing timer for the process, the current time will be used as the basis for the timer (same behavior as specifying <code>CurrentTime</code>).

In all cases, the actual timer expiration is created based on the specified business calendar.

<i>BusinessCalendar</i>	String The alias name of the business calendar in My webMethods Server that you want to reference.
<i>Days</i>	String The number of days to be applied to the timer expressed as a whole number. If you do not specify days, or if you specify an invalid value, the value is assumed to be 0 (zero).
<i>Hours</i>	String The number of hours to be applied to the timer expressed as a whole number. If you do not specify hours, or if you specify an invalid value, the value is assumed to be 0 (zero).
<i>Minutes</i>	String The number of minutes to be applied to the timer expressed as a whole number. If you do not specify minutes, or if you specify an invalid value, the value is assumed to be 0 (zero).

Output Parameters

None.

Usage Notes

The timer is expressed in terms of days, hours, minutes, seconds, and milliseconds. A blank value is assumed to be 0 (zero). If a *BaseDate* of `ExistingTimer` is specified and there is no existing timer for the process, the current time will be used as the basis for the timer (same as *BaseDate* of `CurrentTime`).

pub.prt.timer.process:createWithDate

WmPRT. This service creates a process timer for the specified process instance and the specified date.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID of the process for which you are creating the timer.
<i>Date</i>	Object The actual date/time the timer will expire.

Output Parameters

None.

pub.prt.timer.process:get

WmPRT. This service returns the actual date that the timer will expire for the specified process instance.

Input Parameters

The following table lists the input parameters.

<i>ProcessInstanceID</i>	String Process instance ID for the process timer you are retrieving.
--------------------------	---

Output Parameters

The following table lists the output parameters.

<i>ProcessTimeout</i>	Object The actual date the process timer is set to expire.
-----------------------	---

pub.prt.tn:deleteByCID

WmPRT. Deletes a process instance associated with a given conversation ID.

Use this service to delete process state for processes that involve webMethods Trading Networks and for which you have a conversation ID rather than a process ID.

Input Parameters

The following table lists the input parameters.

<i>ConversationID</i>	String Conversation ID for the process instance for which you want to delete process state information.
-----------------------	--

Output Parameters

The following table lists the output parameters.

<i>success</i>	String Flag indicating whether the process state was deleted. The following values apply: <ul style="list-style-type: none">■ <code>true</code> — The process state was deleted.■ <code>false</code> — The process state was not deleted.
----------------	---

Usage Notes

This service invokes [pub.prt.correlate:lookupCorrelation](#) to look up the mapped *ProcessInstanceID* and invokes [pub.prt.admin:deleteProcess](#) to delete the process state information.

Using this service to delete state of a running process will produce unpredictable results.

See Also

“[pub.prt.admin:deleteProcess](#)” on page 182

pub.prt.tn:getPIDforCID

WmPRT. Returns the process instance ID for a given conversation ID.

Use this service within a process that involves webMethods Trading Networks when you have a conversation ID but need the corresponding process instance ID. This service is a wrapper around [pub.prt.correlate:lookupCorrelation](#).

Input Parameters

The following table lists the input parameters.

<i>ConversationID</i>	String Conversation ID for which you want the associated process instance ID.
-----------------------	--

Output Parameters

The following table lists the output parameters.

<i>ProcessInstanceID</i>	String Conditional. The process instance ID related to the specified conversation ID (if there is one).
<i>success</i>	String Flag indicating whether the process ID was retrieved. The following values apply: <ul style="list-style-type: none">■ <code>true</code> — The process ID was retrieved.■ <code>false</code> — The process ID was not retrieved.

See Also

“[pub.prt.tn:mapCIDtoPID](#)” on page 206

pub.prt.tn:getRoleInfo

WmPRT. Fetches role information for a specified role in process.

Use this service within processes that involve webMethods Trading Networks.

The returned information includes the internal ID of the partner within the Trading Networks system, which you can use to retrieve the Trading Networks profile information.

Input Parameters

The following table lists the input parameters.

<i>ProcessData</i>	Document The <i>ProcessData</i> portion of the pipeline, which is standard information available for all processes. The structure of this document (IData object) is defined by pub.prt.admin:changeProcessStatus .
<i>roleName</i>	String Name of the role for which you want to retrieve information.

Output Parameters

The following table lists the output parameters.

<i>roleInfo</i>	Document Conditional. Role information that is currently available for the specified role. The structure of this document (IData object) is defined by pub.prt.tn:RoleInfo . This parameter is not present if no documents have been sent to or received for this role.
-----------------	--

See Also

“[pub.prt.tn:RoleInfo](#)” on page 208

“[pub.prt.admin:changeProcessStatus](#)” on page 180

pub.prt.tn:handleBizDoc

WmPRT. Sends a Trading Networks BizDocEnvelope (Trading Networks document) to the Process Engine, to allow the document to be processed as part of a business process.

Input Parameters

The following table lists the input parameters.

<i>bizdoc</i>	Object <code>com.wm.app.tn.doc.BizDocEnvelope</code> — Trading Networks BizDocEnvelope document that you want to send to the Process Engine.
<i>ConversationID</i>	String Optional. Conversation ID for the document. Specify <i>ConversationID</i> if the document has no conversation ID or if you want to use an alternate conversation ID.

ProcessModelID **String** Optional. ID of the process model that you want the Process Engine to use to process the document.

ProcessModelVersion **String** Optional. Version of the process model that you want the Process Engine to use to process the document.

Note:

Specifying *ProcessModelID* and *ProcessModelVersion* overrides the normal process of matching a document to a process model version.

prtIgnoreDocument **String** Optional. A flag indicating whether this document is to be ignored or processed by the Process Engine. The following values apply:

- `true` — Ignore the document and *do not* send it to the Process Engine for processing. Setting the flag to `true` causes this service to do nothing.
- `false` — Send the document to the Process Engine for processing.

Output Parameters

None.

Usage Notes

Trading Networks automatically sends documents to the Process Engine if it extracts a conversation ID from the document. If you did not have Trading Networks extract a conversation ID, you can use this service to supply a conversation ID and send the document to the Process Engine to be processed as part of a business process.

pub.prt.tn:mapCIDtoPID

WmPRT. Sets up a mapping between the specified conversation ID and process instance ID.

This service is a wrapper around [pub.prt.correlate:establishCorrelation](#).

Input Parameters

The following table lists the input parameters.

ConversationID **String** Conversation ID that you want to map to the specified process instance ID.

ProcessInstanceID **String** Process instance ID that you want to map to the specified conversation ID.

Output Parameters

The following table lists the output parameters.

<i>success</i>	<p>String Flag indicating whether the mapping was established. The following values apply:</p> <ul style="list-style-type: none"> ■ <code>true</code> — The mapping was established. ■ <code>false</code> — The mapping was not established.
----------------	---

Note:

If this service runs to completion, the mapping has been established.

Usage Notes

The Process Engine automatically establishes this mapping when a Trading Networks document (bizdoc) is used to start a process or is modeled as an output from a process step. Use this service when there is no way for the Process Engine to determine the mapping itself (for example, when a process is started with a non-Trading Networks document and later waits for a Trading Networks document).

Use this service with care. Be sure to create correct conversation ID to process instance ID mappings; an invalid mapping could prevent other processes from completing successfully.

See Also

“pub.prt.tn:getPIDforCID” on page 204

pub.prt.tn:MatchBizDoc

WmPRT. Matches the supplied business document ID to a valid process model. This service provides support for webMethods Rosetta Net in Process Engine.

Input Parameters

The following table lists the input parameters.

<i>bizdoc</i>	<p>Object <code>com.wm.app.tn.doc.BizDocEnvelope</code> — Trading Networks BizDocEnvelope document that you want to send to the Process Engine.</p>
<i>useThisMid</i>	<p>String Contains the model identifier to match to the <i>bizdoc</i>.</p>
<i>requireStart</i>	<p>String A flag that indicates whether the associated step starts the business process. The following values apply:</p> <ul style="list-style-type: none"> ■ <code>true</code> — Specify that the step associated with the specified <i>bizdoc</i> is a step that starts the business process.

- `false` — Specify that the step associated with the *bizdoc* does not have to start the business process.

Output Parameters

The following table lists the output parameters.

<i>modelId</i>	String The process model identifier that matches the specified business document.
<i>trackCount</i>	String The internal process track count for the matching process instance.
<i>stepID</i>	String The step identifier.
<i>errorMessage</i>	String Text of any generated error message (present only if a Service Exception is raised).

Usage Notes

The service invokes the original `TNDispatcher.matchBizDoc()` that returns the `MatchResult` object. This object is parsed for the output parameters used by the Rosetta Net implementation.

pub.prt.tn:RoleInfo

WmPRT. Document type that describes information maintained for roles in a process.

Parameters

The following table lists the parameters.

<i>ProfileID</i>	String Trading Networks internal ID of this trading partner.
<i>CorporationName</i>	String Corporation name that is specified in the Trading Networks profile for this trading partner.
<i>OrgUnitName</i>	String Organizational unit name that is specified in the Trading Networks profile for this trading partner.
<i>Type</i>	String The type of software the partner uses to connect to the trading network. The following values apply: <ul style="list-style-type: none">■ <code>TNServer</code> — The partner is using webMethods Trading Networks.■ <code>TNPartner</code> — The partner is using webMethods for Partners.■ <code>Browser</code> — The partner is using a Web browser.■ <code>Other</code> — The partner is using some other method.

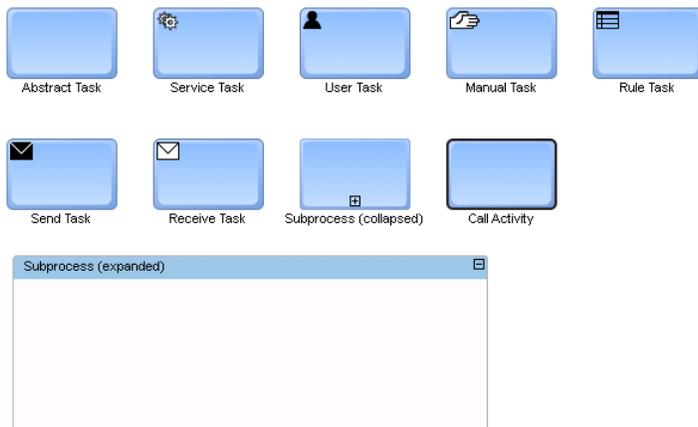
<i>Status</i>	String Status (active or inactive) of the Trading Networks profile for this trading partner.
<i>PreferredProtocol</i>	String The delivery protocol that the partner prefers you to use when sending documents to it. The following values apply: <ul style="list-style-type: none">■ ftp1 - The partner prefers documents sent using the primary FTP protocol.■ ftp2 - The partner prefers documents sent using the secondary FTP protocol.■ http1 - The partner prefers documents sent using the primary HTTP protocol.■ http2 - The partner prefers documents sent using the secondary HTTP protocol.■ https1 - The partner prefers documents sent using the primary HTTPS protocol.■ https2 - The partner prefers documents sent using the secondary HTTPS protocol.■ smtp1 - The partner prefers documents sent using the primary e-mail protocol.■ smtp2 - The partner prefers documents sent using the secondary e-mail protocol.■ null - The partner prefers documents sent using the polling protocol.
<i>LastSendingLocale</i>	String Locale associated with the last transmission from this trading partner. Whenever a document is received from this trading partner, this field is updated with the locale information specified in the transmission. For example, if the trading partner uses HTTP to post an XML document to a Trading Networks server and specifies the "ja_JA" locale in the HTTP transmission, this field will contain ja_JA.

12 Activity Steps

■ About Activities	212
■ About Task Types	214
■ About Abstract Tasks	221
■ About Service Tasks	222
■ About User Tasks	225
■ About Manual Tasks	228
■ About Rule Tasks	230
■ About Send Tasks	233
■ About Receive Tasks	237
■ About Subprocesses	242
■ About Call Activities	253

About Activities

Activities are actions that represent work done as part of a business process. Activities are categorized by *activity types*. Except for the abstract task and the call activity, each activity type displays an ornament image that represents the specific behavior of that activity type. The following activity types are available as displayed on the design canvas:



You can define your own custom ornament images for activity steps, with the exception of the subprocess step. For more information, see [“Assigning Custom Step Ornaments”](#) on page 77.

About Activity Types

Activities are divided into three activity types: task activities, subprocess activities, and call activities. Task activities, or simply tasks, are categorized by task types. Subprocess activities are subprocesses. Call activities are always referred to as call activities.

About Activity Behavior

- [“About Subprocess Behavior”](#) on page 212
- [“About the Compensating Check Box”](#) on page 213
- [“Applying a Compensating Marker to an Activity Step”](#) on page 213
- [“Configuring a Subprocess Standard Loop”](#) on page 246
- [“Configuring a Call Activity Standard Loop”](#) on page 254

About Subprocess Behavior

Subprocess and call activity steps can have subprocess markers . The difference is that subprocesses always have subprocess markers, whereas call activities only have subprocess markers if they contain a reference to another (callable) process and are thus expandable. There is no configuration for a subprocess marker.

BPMN processes can be designated as *callable*. A BPMN *callable process* has a defined input and output signature, known as a *global process specification*, and can be invoked by a call activity. See [“Call Activity Concepts” on page 147](#) and [“Defining a Global Process Specification” on page 97](#) for more information.

About the Compensating Check Box

All activities provide a **Compensating** check box on the **Implementation** page of the activity’s Properties view. You must be in Process Developer mode to view the **Implementation** page. To learn more, see [“About Process Development Modes” on page 52](#).

When you select the **Compensating** check box, a compensation marker ◀ is displayed on the activity step. This is a notational option only, and it has no effect at all on the behavior of the activity step. If you want the step to provide compensating behavior, you must also configure the step to execute the compensating actions you want to apply.

Applying a Compensating Marker to an Activity Step

You can apply a compensating marker to an activity step. This is a notational option only, and it has no effect at all on the behavior of the activity step.

➤ To apply a compensation marker

1. Open the process you want to work with in the process editor.
2. Click the activity you want to work with to select it.
3. In the Properties view, click the **Implementation** page.
4. Select the **Compensating** check box. A compensating marker appears on the activity step.
5. Save the process.

About Looping Behavior

You can implement looping behavior for subprocess steps and call activities steps. This behavior is represented by a  loop marker placed on the step. For more information about step looping, see [“About Standard Looping” on page 129](#).

Subprocess looping and call activity looping are configured differently. For configuration procedures, see [“Configuring a Subprocess Standard Loop” on page 246](#) and [“Configuring a Call Activity Standard Loop” on page 254](#).

Note:

You can also implement transition looping for any step. For more information, see [“About Transition Looping” on page 129](#).

About Task Types

Tasks are one type of activity. Activities represent work that is done within a process, and task activities are specialties of those actions. Designer supports the listed in the following table task types:

Palette Icon	Task Type	Description
	Abstract Task	“About Abstract Tasks” on page 221
	Service Task	“About Service Tasks” on page 222
	User Task	“About User Tasks” on page 225
	Manual Task	“About Manual Tasks” on page 228
	Rule Task	“About Rule Tasks” on page 230
	Send Task	“About Send Tasks” on page 233
	Receive Task	“About Receive Tasks” on page 237

Adding a Task Activity

➤ To add a task activity to a process

1. Open the process model you want to work with in the process editor and do one of the following:
 - Drag the task activity you want to add from the palette to the design canvas, or, in the process editor palette, click the task activity you want to add, then click the design canvas where you want to place it. You must use one of these methods the first time you add a task activity to an empty process editor.
 - Hover the cursor over an existing step in the process editor to display the speed buttons. Click the speed button for the activity type you want to add (last selected from palette, subprocess, or call activity), and then click the canvas at the location where you want to add the task activity. A transition line is created automatically from the source step to the task activity.

Tip:

If the task activity is not the one you want, right-click the task activity and click **Change Task Type** to specify a different task activity type.

Tip:

You can set the label of the task activity to be empty, or to have the same name as another step in the process. You can also change the placement of the label. For more information, see [“About Step Labels” on page 72](#).

2. Add transition lines if needed and configure the task activity as necessary in the Properties view.
3. Save the process.

Configuring Task Types

Each task type is configured on the pages of the Properties view of the task. Each task type contains properties and settings specific to its type. For more information about configuring task activities, see the following topics:

- [“Configuring Abstract Tasks” on page 221](#)
- [“Configuring Service Tasks” on page 222](#)
- [“Configuring User Tasks” on page 225](#)
- [“Configuring Manual Tasks” on page 228](#)
- [“Configuring Rule Tasks” on page 230](#)
- [“Configuring Send Tasks” on page 234](#)
- [“Configuring Receive Tasks” on page 238](#)

Removing Task Types

➤ To remove a task type from a process

1. Click the task you want to remove to select it.
2. Press the Delete key, or right-click the task and click **Delete**.
3. Save the process.

Changing Task Types

While developing a process in Designer, you can change the current task type step to different task type step.

Existing process documentation is unaffected when you change one task type into another. Designer retains the following properties from the previous task type step (when present):

- Flow service
- Web service
- Rule service

- webMethods business task ID
- Referenced process ID
- Task name
- Referenced process name
- Retry count
- Retry interval
- Integration Server ID
- Allow parallel execution value

These values are preserved so that you can easily switch back to the previous step type.

➤ To change a task to a different type

You can change a task into any other task type in several ways:

- Use the procedures described in [“Changing an Activity, Event, or Gateway Type” on page 83](#).
- Drag an Integration Server (IS) service, a Web service, or an Integration Server document onto an existing task activity step to change it into the corresponding step type.
- To change any task activity to a user task activity, right-click the task activity and click **Create Implementation > New Task**.
- To change any task activity to a service task activity that uses an Integration Server (IS) service, right-click the task activity and click **Create Implementation > New Flow Service**.
- To change any task activity to a service task activity that uses a Web service, right-click the task activity and click **Create Implementation > New Web Service Connector**.

Changing Task Images

You can change the default image that Designer displays inside a task step. You cannot change the task step icon. When you change the task type icon, you can choose from a default set of system images within Designer, or you can use a custom image.

Note:

You cannot change the internal images for call activities, subprocesses, gateways, or events.

➤ To change a task image

1. In the process editor, right-click the task.
2. Click **Choose Image** and do one of the following:

- To replace the default image with a system image, click the image you want on the **Shared Images** tab.
 - To remove a system image and return to the default image, click **Restore Defaults**.
 - To apply a custom image, click **Browse** to locate and select an image from your system. The default image size is 21 x 21 pixels. You can use images in SVG, JPG, JPEG, and PNG formats. Designer resizes images to 21 x 21 pixels if they are not already this size.
3. Click **OK**.
 4. Save the process.

Changing Task Label Placement

By default, Designer places a label for tasks, call activity, and collapsed subprocess steps beneath the step (this is the existing webMethods style). Designer also supports the BPMN style of label placement, with the label on the task, call activity, and collapsed subprocess step. This behavior does not apply to event, gateway, and subprocess steps.

➤ To change the task, call activity, and collapsed subprocess label placement

1. You can set the default behavior for label placement with the **Default step label location** preference by specifying either of the following:
 - **On step**. Click this setting to display labels on top of task and call activity steps.
 - **Below step**. Click this setting to display labels below task and call activity steps.

For more information, see [“Configuring Appearance Preferences” on page 36](#).
2. To change the label placement for tasks, call activities, and collapsed subprocess in a process open in the process editor, click the Step Label Position button  on the tool bar.
3. Save the process.

Basic Task Properties

Note:

Manual tasks do not have **Inputs / Outputs** or **KPIs** pages in the Properties view, as a manual task does not represent data moving through the process.

The following table describes the basic task properties.

Properties Page	Property	Description
General	Type	The task activity type. You can use this field to change the task type to another task type.
	Label	Step name. Default is Task1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Allow this receive task to start new process instance	Available for webMethods receive tasks only. Select the check box to allow a receive task to start a new process instance.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name , Font Size , Bold , Italic , and Font Color . Default settings are Tahoma 8 point black , with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned Documentation Field .
KPIs	KPIs for <Step Name>	Step-level KPI (Key Performance Indicator) definitions. For more information about the fields on this page and how to work with them, see “About KPIs” on page 372 .
	KPI Properties	Information used to describe a KPI: Name , Description , Unit of Measure , Associated Field , Aggregation Type , and Dimensions . See “About Step-Level KPIs” on page 375 .
	Name	Name of the KPI. Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names on the target server. However, if you duplicate a KPI

Properties Page	Property	Description
		name in a process, generation of the process produces a warning.
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.
	Associated Field	Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used.
		<p>Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.</p>
	Aggregation Type	Aggregation type can be set to SUM , AVERAGE , or LAST VALUE
	Dimensions	Use this area to add, delete, and manage data dimensions for a selected KPI. You can associate a field from the step output with one or more dimensions.
		<p>Note: Dimension names must be unique on a given Optimize server. Designer cannot, at design time, detect all the dimension names on the target server. However, if you duplicate a dimension label in a process, generation of the process produces a warning.</p> <p>For more information, see “About KPIs” on page 372.</p>
Inputs / Outputs	Inputs	<p>Data flowing into the step. You can add and remove inputs manually, as well as add or update from the service signature.</p> <p>Available for call activities (including webMethods referenced process behavior),</p>

Properties Page	Property	Description
	Outputs	<p>abstract tasks, service tasks, user tasks, rule tasks, and send tasks.</p> <p>For more information, see “Create Inputs and Outputs” on page 91 and “Remove Inputs and Outputs” on page 92.</p> <p>Data flowing out of the step. You can add and remove outputs manually, as well as add or update from the service signature.</p> <p>Available for call activities (including webMethods referenced process behavior), abstract tasks, service tasks, user tasks, rule tasks, and receive tasks.</p>
Transitions	Transition Type	<p>Defines the transition behavior. Options are If Condition, Default, Join Timeout, Iterations Exceeded, and Unsatisfied Join.</p> <p>For more information about these transition types and their configuration fields, see:</p> <ul style="list-style-type: none"> ■ “Configuring Transition Behavior” on page 125. ■ “About Transition Type Behavior” on page 118. ■ “About Transition Types” on page 118. ■ “How Incomplete Transitions Affect Join Steps and Gateways” on page 132.
Participants	Participants	<p>Available for user and manual tasks only. Add or remove the task participants. You can define these as roles or individuals.</p>

Advanced Task Properties

Advanced task properties differ for task types, based on their functions in the process.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

For details, see the following topics:

- [“Advanced Abstract Task Properties” on page 221](#)

- [“Advanced Service Task Properties” on page 222](#)
- [“Advanced User Task Properties” on page 226](#)
- [“Advanced Manual Task Properties” on page 228](#)
- [“Advanced Rule Task Properties” on page 230](#)
- [“Advanced Send Task Properties” on page 234](#)
- [“Advanced Receive Task Properties” on page 238](#)

About Abstract Tasks

The Empty step in legacy Designer migrates to an Abstract task. Abstract tasks have no run-time implementation. Because Abstract tasks have no run-time implementation, then any Flow Elements added in the Flow view will not be executed at run time. Those Flow Elements are treated as part of the step and until the step is implemented, no part of the step (including the Flow Elements) will be executed.

Configuring Abstract Tasks

Abstract tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities” on page 212](#).

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

➤ To configure an abstract task

1. In an open process, click the abstract task you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described in [“Basic Task Properties” on page 217](#).
 - Configure the advanced task properties described in [“Advanced Abstract Task Properties” on page 221](#).
3. Save the process.

Advanced Abstract Task Properties

The following table describes the advanced abstract task properties.

Properties Page	Property	Description
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.

About Service Tasks

Service tasks call services, either Integration Server services or Web services (rule services are associated with the rules task activity). Service tasks are configured on the pages of the Properties view.

An Integration Server Service step or Web Service step imported from older versions of Designer becomes a service task activity when you open the process in Designer 8.2 or later.

Configuring Service Tasks

Service tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities” on page 212](#).

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

> To configure a service task

1. In an open process, click the service task you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described in [“Basic Task Properties” on page 217](#).
 - Configure the advanced task properties described in [“Advanced Service Task Properties” on page 222](#).
3. Save the process.

Advanced Service Task Properties

The following table describes the advanced service task properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Service	Service type: IS Service (default) or Web Service . <ul style="list-style-type: none"> ■ Click Browse to select an existing service from the connected Integration Server. ■ Click New to open the Create a New Flow Service wizard. ■ Click View to open the service associated with the step for viewing.
	Generated Service Name	Name of the generated flow service used at run time.
	Retry Count	Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries. <div data-bbox="894 982 1458 1220" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an <code>EXIT</code> with a <code>FAILURE</code> error, causes the step to fail.</p> </div>
	Retry Interval	Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.

Properties Page	Property	Description
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p>

Properties Page	Property	Description
		This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
	Ignore dead path notification	This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.

About User Tasks

User tasks are tasks completed by a human within the scope of the process, and call a webMethods task to provide the interface. They are configured on the pages of the Properties view. Previously existing webMethods tasks (not the same as BPMN task activities) become new user tasks when you open the process in Designer 8.2 or later.

For more information about working with tasks, see *webMethods BPM Task Development Help*.

Configuring User Tasks

User tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities”](#) on page 212.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes”](#) on page 52.

> To configure a user task

1. In an open process, click the user task you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described in [“Basic Task Properties”](#) on page 217.
 - Configure the advanced task properties described in [“Advanced User Task Properties”](#) on page 226.
3. Save the process.

Advanced User Task Properties

The following table describes the advanced user task properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Task Name	Name of the task. Click the ... button to select from the tasks available in the workspace.
	Task Type ID	ID assigned to the task type.
	Task Control Set	If a task has available control sets, select the control set you want to use from the drop-down list.
	Generated Service Name	Name of the generated flow service used at run time.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR , AND , COMPLEX , or Unsynchronized OR . See “About Joins” on page 131 .
	Join Timeout	The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be: Note: The Joins page is displayed only when a step has more than one incoming transition.

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Manual Tasks

Manual tasks are not like other tasks in that there is nothing in a manual task to process. The manual task icon is a hand, indicating that a manual task is something you do with your hands. Any activity that does not involve back-end processing is a good candidate for a manual task. Since there is no data to process, a manual task can be considered as a placeholder for a manual activity.

Note:

Manual tasks are unrelated to manual decisions, which are used with rule tasks.

Configuring Manual Tasks

Manual tasks are configured on the pages of the Properties view. Manual tasks do not have inputs or outputs, or KPIs in the Properties view, as they do not represent data moving through the process. For more about activities, see [“About Activities” on page 212](#).

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

› To configure a manual task

1. In an open process, click the manual task you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described [“Basic Task Properties” on page 217](#).
 - Configure the advanced task properties described in [“Advanced Manual Task Properties” on page 228](#).
3. Save the process.

Advanced Manual Task Properties

The following table describes the advanced manual task properties.

Properties Page	Property	Description
Implementation	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR , AND , COMPLEX , or Unsynchronized OR .

Properties Page	Property	Description
		See “About Joins” on page 131 .
	Join Timeout	<p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> <p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

Properties Page	Property	Description
		<p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Rule Tasks

With version 8.2 and later, webMethods Business Rules are available for rule-based processing. When you create a new rule task, you use webMethods Business Rules.

Configuring Rule Tasks

Rule tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities”](#) on page 212.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes”](#) on page 52.

If you use an event rule in a rule task, there must be a JMS trigger on the corresponding Integration Server. For more information about working with event rules, see *webMethods BPM Rules Development Help*.

> To configure a rule task

1. In an open process, click the rule task you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described in [“Basic Task Properties”](#) on page 217.
 - Configure the advanced task properties described in [“Advanced Rule Task Properties”](#) on page 230.
3. Save the process.

Advanced Rule Task Properties

The following table describes the advanced rule task properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Rule Type	Designer supports both webMethods Business Rules.
	Rule Information	For webMethods Business Rules:
	Available fields depend on the Rule Type selection.	Select either Rule Set or Decision Entity . If you select Rule Set , specify the Rule Set Name ; if you select Decision Entity , specify the Decision Entity Name . Click the Browse button to open the webMethods Business Rules window and select a rule set or a decision table, depending on your earlier selection.
		<p>Note: An event rule is a decision entity. It specifies the results triggered by an event that occurs during rule execution. If you use an event rule in a rule task, there must be a JMS trigger on the corresponding Integration Server. For more information about working with event rules, see <i>webMethods BPM Rules Development Help</i>.</p> <p>The Rule Project Name is automatically populated based on the Rule Set or Decision Entity.</p>
	Generated Service Name	Name of the generated flow service used at run time.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.

Properties Page	Property	Description
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p>

Properties Page	Property	Description
		<p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Send Tasks

The send task is designed to work together with a receive task. A send task delivers a message to a targeted recipient (the receive task) using one of the following protocols:

- Subscription (for publishable documents)
- JMS (for JMS -triggered processes) (default)
- Simple service (for synchronous receive/send)

Send task steps behave as follows:

- In a top level process: the send task will publish the associated document.
- In a child process called using the dynamic referenced process step: The send task will publish the document, and if it is listed as one of the return documents, the send task will send the document to the parent process instance directly.
- In a child process called using a call activity step: the send task will publish the document. The send task will publish the document, and the pipeline will pass from the child to the parent when the child process completes, which could be several steps after the send task step.

Note:

When creating a callable process that is to be called by a call activity step, always implement a send task in the callable process to return to the parent process. If you use either a message end event or message intermediate event, those steps will only publish a document and will not return it to the parent.

When a send task step is used in a process invoked as a referenced subprocess:

- If an IS document is configured both as a **Send Document** and a **Return Document** for the parent process, the document is not published to the messaging provider.
- If an IS document is configured as a **Send Document**, but not as a **Return Document** for the parent process, the document is published to the messaging provider.

Publish steps in models created with versions of Designer earlier than 8.2 are converted to send tasks upon import. These send tasks are automatically configured to use the publish document from the original publish step. Reply steps that use the simple service protocol in earlier Designer models are also converted to send tasks.

Configuring Send Tasks

Send tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities” on page 212](#).

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

> To configure a send task

1. In an open process, click to select the send task you want to configure.
2. In the Properties view, do one or both of the following:
 - Configure the basic task properties described in [“Basic Task Properties” on page 217](#).
 - Configure the advanced task properties described in [“Advanced Send Task Properties” on page 234](#).
3. Save the process.

Advanced Send Task Properties

The following table describes the advanced send task properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Send Protocol	Type of send protocol to use. Select Subscription (For Publishable Documents) , JMS (for JMS Triggered Processes) (default), or Simple Service (For Synchronous Receive/Send) from the list.
	Protocol Properties	Available for JMS and Simple Service protocols only. <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS or click the browse button to select another

Properties Page	Property	Description
		<p>alias (an Integration Server connection is required).For more information, see “Configuring a JMS-Triggered Process” on page 450.</p> <ul style="list-style-type: none"> ■ Destination Name. Required. <i>You must specify a destination name.</i> Click the browse button (an Integration Server connection is required).
	Send Document	<p>Integration Server document to send.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field.
	Send synchronous reply to	Name of the Receive Task to which the Send Task is responding. The Receive Task must be configured to use the Simple Service protocol before you can select it.
	Generated Service Name	Name of the generated flow service used at run time.
	Retry Count	<p>Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an <code>EXIT</code> with a <code>FAILURE</code> error, causes the step to fail.</p> </div>
	Retry Interval	Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time

Properties Page	Property	Description
		exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>

Properties Page	Property	Description
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Receive Tasks

Pre-BPMN Designer receive steps that use **Simple Service (For Synchronous Receive/Send)** messaging protocol steps migrate to receive tasks.

As noted in [“About Send Tasks” on page 233](#), send and receive tasks replace the request and reply model in legacy Designer processes.

The receive task works with a partner activity, the send task. A send task delivers a message to a targeted receive task using the **Simple Service** protocol. This is the same protocol used in legacy Designer request and reply steps.

Receive tasks support the **Subscription (For Publishable Documents)** (default), **JMS (for JMS Triggered Processes)**, and **Simple Service (For Synchronous Receive/Send)** messaging protocols.

Legacy Designer receive steps that cannot start a process instance and *do not* have incoming sequence flows migrate to receive tasks.

Legacy Designer receive steps that cannot start a process instance and *do* have incoming sequence flows migrate to message intermediate events.

Configuring Receive Tasks

Receive tasks are configured on the pages of the Properties view. For more about activities, see [“About Activities” on page 212](#).

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

➤ To configure a receive task

1. In an open process, click the receive task you want to configure to select it.
2. In the Properties view, do one of the following:
 - Configure the basic task properties described in [“Basic Task Properties” on page 217](#).
 - Configure the advanced task properties described in [“Advanced Receive Task Properties” on page 238](#).
3. Save the process.

Advanced Receive Task Properties

The following table describes the advanced receive task properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Receive Protocol	Type of receive protocol to use. Select Subscription (For Publishable Documents) (default), JMS (for JMS Triggered Processes) , or Simple Service (For Synchronous Receive/Send) from the list.
	Receive Document	The document type to receive.

Note:

The **Receive Protocol** is not editable in Adapter Notification steps.

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ Click Browse to open the Choose Document dialog box and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type dialog box and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <p>Note: The Receive Document is not editable in Adapter Notification steps.</p> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p>
	Protocol Properties	<p>Available for JMS protocol only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRASACTIONAL_ALIAS or click the browse button to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. ■ Destination Name. Required. <i>You must specify a destination name.</i> Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click the browse button (an Integration Server connection is required).
	Subscription Filter	<p>Available for the Subscription (For Publishable Documents) and JMS (For JMS Triggered Processes) protocols only. The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444.</p>

Properties Page	Property	Description
	Generated Service Name	Name of the generated flow service used at run time.
	Retry Count	<p>Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries.</p> <p>Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an EXIT with a FAILURE error, causes the step to fail.</p>
	Retry Interval	Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
	E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>Receive tasks that start processes and those that do not start processes can use e-forms.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p> <p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle</p>

Properties Page	Property	Description
		forms). Otherwise, they will not trigger the receive task.
	Content Repository	Select Browse to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	Selecting an e-form as the step's Receive Document automatically populates this field. For more information about using e-forms, see “Using E-forms in a Process” on page 69 and .
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR , AND , COMPLEX , or Unsynchronized OR . See “About Joins” on page 131 .
	Join Timeout	<p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> <p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value.

Properties Page	Property	Description
	Join Condition	<p>For more information, see “About Join Timeouts” on page 141.</p> <p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>
Correlation	Correlation	<p>Select Not Used, Field, or Service. For more information, see “Specifying Correlations” on page 165.</p>

About Subprocesses

In BPMN 2.0 terminology, a subprocess is referred to as *subprocess activity*, but in general, this documentation uses the term *subprocess* to refer to a subprocess activity. For more information about activities, see [“About Activities”](#) on page 212. Although earlier webMethods processes are supported, they are deprecated and you are advised to use BPMN subprocesses when you create a new subprocess, and to convert any imported webMethods subprocesses to BPMN subprocesses. For more information, see [“Subprocess Concepts”](#) on page 145 and [“About Subprocess Types”](#) on page 146.

A subprocess is not a self-contained process, and can exist only inside a process model. A subprocess contains a group of steps, and the subprocess is treated as a step in the parent process. You can also define KPIs for a subprocess. In addition, a subprocess can contain one or more subprocesses. For more information, see [“About Subprocess Nesting” on page 243](#).

A subprocess can be resized, collapsed, and expanded in the process editor. You cannot, however, collapse a subprocess that crosses swimlanes.

About Subprocess Nesting

A subprocess can contain one or more subprocesses, nested to as many levels as you require. However, some limitations apply when nesting subprocesses. These limitations arise from support in Designer for two subprocess types, webMethods subprocesses and BPMN subprocesses. For more information, see [“About Subprocess Types” on page 146](#).

- When you add a subprocess to an existing subprocess, the child subprocess type is set to the same subprocess type as the parent subprocess. For example, if you add a subprocess to a BPMN subprocess, the subprocess type of the newly added child process will be automatically set to a BPMN subprocess.
- You cannot change the process type for child subprocesses. The **Type** controls on the **Implementation** page of the properties view are disabled for each child subprocess.
- If you change the subprocess type for the parent subprocess, the subprocess type change is also applied to all child subprocesses.
- If you move a child subprocess out of its parent subprocess and onto the canvas as a top-level subprocess, the ability to change the subprocess type is restored.

For more information about changing subprocess types, see [“Recommendations On Changing the Subprocess Type” on page 244](#).

Adding a Subprocess

➤ To add a subprocess to a process

1. Click the button on the palette that represents a subprocess.
2. Click the process editor where you want to place the new subprocess.

Tip:

The palette supports dragging as well as clicking operations.

After you add a subprocess, you can add and configure the task activity types you need, add error handling, and otherwise implement the subprocess to your requirements.

Recommendations On Changing the Subprocess Type

Two types of subprocess are supported, as described in [“Subprocess Concepts” on page 145](#) and [“About Subprocess Types” on page 146](#). The subprocess type is shown on the **Implementation** page of the subprocess Properties view.

You can change a webMethods subprocess to a BPMN subprocess, and you can change a BPMN subprocess to a webMethods subprocess, as described in [“Changing the Subprocess Type” on page 245](#).

Note:

You cannot change the subprocess type for subprocesses that are nested within another subprocess. For more information, see [“About Subprocess Nesting” on page 243](#).

When changing the subprocess type, the following recommendations apply:

- **webMethods subprocess to BPMN subprocess (recommended).** When you import a process model created with an 8.2.2 or earlier version of Designer Process Development, all subprocesses are imported as webMethods subprocesses, without changes. These subprocesses will build and execute as expected in the later version. However, you are strongly advised to change them to BPMN subprocesses at your earliest convenience, as support for the webMethods subprocess type will be removed in a future release. For more information, see [“About Changing webMethods Subprocesses to BPMN Subprocesses” on page 244](#).
- **BPMN subprocess to webMethods subprocess (not recommended).** Although the interface enables you to make this change, it is not generally recommended. If you do so, you must remove all BPMN subprocess behavior that is not supported in a webMethods subprocess. The non-compatible features are listed in the Problems view when you change the subprocess type. Refer to [“About Subprocess Types” on page 146](#) to determine the differences in the two subprocess types, and how your BPMN subprocess will be affected if you make this change. Support for the webMethods subprocess type will be removed in a future release.

About Changing webMethods Subprocesses to BPMN Subprocesses

Process models created with webMethods Process Development version 8.2.2 and earlier use webMethods subprocesses. With later versions, webMethods subprocesses are deprecated, and the use of the BPMN subprocess type is recommended.

There are two methods available for you to change your webMethods Subprocesses (Deprecated) to BPMN Subprocesses. You can:

- Create a new BPMN subprocess that duplicates the behavior of the old webMethods subprocess. You can then delete the webMethods subprocess.
- Change the webMethods Subprocess (Deprecated) type to BPMN Subprocess as described in [“Changing the Subprocess Type” on page 245](#).

Before you do either of these, you must examine the logical behavior of the webMethods subprocess being changed to determine if additional modifications are needed in the following areas:

- **Boundary intermediate error events.** In a webMethods Subprocess (Deprecated), these are non-interrupting only. They are changed to interrupting when you change to a BPMN Subprocess.
- **The subprocess output.** In a BPMN Subprocess, the output will no longer be a merge of all subprocess tracks. The pipeline output from a BPMN subprocess is from the last completed track only. To merge track outputs, you must now do so explicitly in the subprocess.
- **Subprocess completion.** A webMethods Subprocess (Deprecated) ends when all tracks in the subprocess complete normally. A BPMN Subprocess ends when there are no active tracks remaining in the subprocess.
- **Terminate behavior.** When a webMethods Subprocess (Deprecated) contains a terminate end event, it terminates the parent process with a configurable status of Completed, Canceled, or Failed. A BPMN subprocess with a terminate end event always ends the subprocess only, with no direct effect on the parent process.
- **Loop counting.** In a webMethods Subprocess (Deprecated) with standard looping, each loop iteration increments the step iteration counter of steps in the subprocess. For example, a step iteration count would be 1 for the first loop iteration, 2 for the second iteration, and so on.

BPMN processes now provide loop counting as well as step counting, and the two are independent. Within each loop iteration, each step in the subprocess begins counting at 1.

This is significant if you depend on step counts in your webMethods Subprocess (Deprecated) to trigger any process logic. It is likely that the step count will not be reached and your logic will not be triggered.

- **Transition looping.** When a webMethods Subprocess (Deprecated) uses a Step Iterations Exceeded transition type, it is possible that with the new independent loop and step counting mechanism in BPMN subprocesses (described above), this transition looping may not work as expected after conversion to a BPMN subprocess. You are advised to use the **Maximum Loop Count** property of the subprocess to implement this behavior.

Changing the Subprocess Type

Before you change a subprocess type, be sure you understand the results of doing so. For more information, see [“Recommendations On Changing the Subprocess Type” on page 244](#) and [“About Changing webMethods Subprocesses to BPMN Subprocesses” on page 244](#).

» To change the subprocess type

1. Open the process you want to work with in the process editor.
2. In the process editor, click the subprocess you want to change.
3. On the **Implementation** tab in the Properties view, do one of the following:
 - Click **webMethods Subprocess (Deprecated)** to change the subprocess type for a BPMN subprocess.

- Click **BPMN Subprocess** to change the subprocess type for a webMethods subprocess.

Note:

You will likely need to modify the subprocess after the change to obtain the expected results. For more information, see [“Recommendations On Changing the Subprocess Type” on page 244](#) and [“About Changing webMethods Subprocesses to BPMN Subprocesses” on page 244](#).

4. Save the process.

Configuring a Subprocess

A subprocess is configured on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

> To configure a subprocess

1. In an open process, click the subprocess you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic subprocess properties described in [“Basic Subprocess Properties” on page 248](#).
 - Configure the advanced subprocess properties described in [“Advanced Subprocess Properties” on page 250](#).

To implement a standard loop, see [“Configuring a Subprocess Standard Loop” on page 246](#).

3. Save the process.

Configuring a Subprocess Standard Loop

When you loop a subprocess, you specify conditions that must be met for it to execute a loop by defining a loop expression. In other words, the subprocess looping will continue as long as the loop expression is true.

> To configure a standard loop for a subprocess

1. In the process editor, open the process you want to work with and click a subprocess to select it.
2. On the **Loop** page in the Properties view of the subprocess, set the following:

Property	Description
Loop Expression	<p>Click  Add to add information that defines the loop expression. You can add multiple condition lines by specifying the AND/OR operator and clicking Add again.</p> <ul style="list-style-type: none"> ■ Field Name. Click this table cell and select an available document field from the dropdown list. The loop expression is based on the value of this field. ■ Operator. Click this table cell and select an operator to use to compare the Field Name value and the Comparison Value/Field value. See “About Expression Operators” on page 173 for more information. ■ Comparison Value/Field. Click this table cell and select an available document from the list and select a comparison field in that document, or type a value that you want to compare with the value selected for Field Name. ■ AND/OR. Use the AND and OR operators to define the logical behavior if you specify multiple parameters for the loop expression.
Test Loop Expression	<p>Select one of the following options to define if the loop expression is evaluated before the step activity begins, or after the step activity completes:</p> <p>before activity or after activity</p>
Maximum Loop Count	<p>Specify the maximum number of loop iterations:</p> <ul style="list-style-type: none"> ■ Click value and type a static integer value. ■ Click field and select an available pipeline field. <p>The specified number will override the loop expression.</p>

Removing a Subprocess

You can remove a subprocess from a process model in the process editor.

> To remove a subprocess from a process

1. In Designer, open the process model you want to work with.
2. In the process editor, right-click the subprocess you want to remove.
3. Click **Delete**.
4. Save the process.

Setting the Subprocess Color

You can specify the background color displayed within a subprocess as follows:

- Set the default color for subprocesses in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**.
- Set a specific background color for an individual subprocess as described below.

➤ To set a background color for an individual subprocess

1. Open the process you want to work with in the process editor.
2. Right-click the subprocess and click **Choose Color**.
3. In the Color dialog box, click a color or define a custom color and then click **OK**.
4. Save the process.

Basic Subprocess Properties

The following table describes the basic subprocess properties.

Properties Page	Property	Description
General	Label	Step name. Default is Subprocess1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name , Font Size , Bold , Italic , and Font Color . Default settings are Tahoma 8 point black, with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned Documentation Field .

Properties Page	Property	Description
KPIs	KPIs for <Step Name>	Step-level KPI (Key Performance Indicator) definitions. For more information about the fields on this page and how to work with them, see “About KPIs” on page 372 .
	KPI Properties	Information used to describe a KPI: Name, Description, Unit of Measure, Associated Field, Aggregation Type, and Dimensions . See “About Step-Level KPIs” on page 375 .
	Name	Name of the KPI. Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names on the target server. However, if you duplicate a KPI name in a process, generation of the process produces a warning.
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.
	Associated Field	Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used. Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.
	Aggregation Type	Aggregation type can be set to SUM, AVERAGE, or LAST VALUE
	Dimensions	Use this area to add, delete, and manage data dimensions for a selected KPI. You can associate a field from the step output with one or more dimensions. Note:

Properties Page	Property	Description
		<p>Dimension names must be unique on a given Optimize server. Designer cannot, at design time, detect all the dimension names on the target server. However, if you duplicate a dimension label in a process, generation of the process produces a warning.</p> <p>For more information, see “About KPIs” on page 372.</p>
Transitions	Transition Type	<p>Defines the transition behavior. Options are If Condition, Default, Join Timeout, Iterations Exceeded, and Unsatisfied Join.</p> <p>For more information about these transition types and their configuration fields, see:</p> <ul style="list-style-type: none"> ■ “Configuring Transition Behavior” on page 125. ■ “About Transition Type Behavior” on page 118. ■ “About Transition Types” on page 118. ■ “How Incomplete Transitions Affect Join Steps and Gateways” on page 132.

Advanced Subprocess Properties

The following table describes the advanced subprocess properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Type: webMethods Subprocess (Deprecated)	Click this option to specify that the subprocess is a webMethods subprocess type. This is the default setting when a subprocess is imported from an earlier version. This subprocess type is deprecated. For more information, see “Recommendations On Changing the Subprocess Type” on page 244 .
	Type: BPMN Subprocess	Click this option to specify that the subprocess is a BPMN subprocess type. This is the default

Properties Page	Property	Description
		setting when a subprocess is added to the process. For more information, see “Recommendations On Changing the Subprocess Type” on page 244.
	Retry Count	Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries. Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an <code>EXIT</code> with a <code>FAILURE</code> error, causes the step to fail.
	Retry Interval	Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR , AND , COMPLEX , or Unsynchronized OR . See “About Joins” on page 131. Note: The Joins page is displayed only when a step has more than one incoming transition.
	Join Timeout	The timeout duration value, after which this join fails. This option is available for <code>AND</code> and <code>COMPLEX</code> joins only. The source of this value can be: <ul style="list-style-type: none"> ■ A static value that you define.

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>
Loop	Loop Expression	<p>Define the loop expression using the table and provided operators.</p> <p>For more information about configuring standard looping, see “Configuring a Subprocess Standard Loop” on page 246.</p>

Properties Page	Property	Description
	Test Loop Expression	Select one of the following options to define if the loop expression is evaluated before the step activity begins, or after the step activity completes: before activity or after activity
	Maximum Loop Count	Set the maximum number of loops using a static value or a field.

About Call Activities

A call activity can be used to invoke either a *callable process* or a *referenced process*. In addition, these processes can be called statically or dynamically. For more information, see [“Call Activity Concepts” on page 147](#).

Important:

Although still available, the ability to invoke a referenced process from a call activity step is deprecated. You are advised to implement all of your steps that invoke another process with a call activity step that invokes a callable process.

The pipeline returned to the call activity step is the pipeline captured when the child process ends. Ensure that you join all paths of execution accordingly to construct the appropriate pipeline.

Note:

When you are setting Quality of Service options for a process, be advised that Designer *never* implements the Express Pipeline setting for a call activity step, regardless of the **Express Pipeline** option setting. For more information, see [“Setting Quality of Service for a Process” on page 111](#).

A call activity displays a subprocess marker when it is configured to call an existing callable process or referenced process. An unconfigured call activity step does not show a subprocess marker. You can configure a call activity for standard looping.

Adding Call Activities

» To add a call activity to a process

1. Click the button on the palette that represents a call activity.
2. Click the process editor where you want to place the new call activity.

Tip:

The palette supports dragging as well as clicking operations.

Configuring Call Activities

Call activities are configured on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

> To configure a call activity

1. In an open process, click the call activity you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic call activity properties described in [“Basic Call Activity Properties” on page 255](#).
 - Configure the advanced call activity properties described in [“Advanced Call Activity Properties” on page 258](#).

To implement a standard loop, see [“Configuring a Call Activity Standard Loop” on page 254](#).

3. Save the process.

Configuring a Call Activity Standard Loop

When you loop a call activity, you specify conditions that must be met for it to execute a loop by defining a loop expression. In other words, the call activity looping will continue as long as the loop expression is true.

> To configure a standard loop for a call activity

1. In the process editor, open the process you want to work with and click a call activity to select it.
2. On the **Loop** page in the Properties view of the subprocess, set the following properties, as described in the table below:

Property	Description
Loop Expression	Click  Add to add information that defines the loop expression. You can add multiple condition lines by specifying the AND/OR operator and clicking Add again.

Property	Description
	<ul style="list-style-type: none"> ■ Field Name. Click this table cell and select an available document field from the list. The loop expression is based on the value of this field. ■ Operator. Click this table cell and select an operator to use to compare the Field Name value and the Comparison Value/Field value. See “About Expression Operators” on page 173 for more information. ■ Comparison Value/Field. Click this table cell and select an available document from the list and select a comparison field in that document, or type a value that you want to compare with the value selected for Field Name. ■ AND/OR. Use the AND and OR operators to define the logical behavior if you specify multiple parameters for the loop expression.
Test Loop Expression	<p>Select one of the following options to define if the loop expression is evaluated before the step activity begins, or after the step activity completes:</p> <p>before activity or after activity</p>
Maximum Loop Count	<p>Specify the maximum number of loop iterations:</p> <ul style="list-style-type: none"> ■ Click value and type a static integer value. ■ Click field and select an available pipeline field. <p>The specified number will override the loop expression.</p>

Removing Call Activities

You can remove a call activity from the an open process in the editor.

➤ To remove an call activity from a process

1. Open a process in the process editor and click the call activity you want to remove to select it.
2. Press the **Delete** key.
3. Save the process.

Basic Call Activity Properties

The following table describes the basic call activity properties.

Properties Page	Property	Description
General	Label	Step name. Default is Call Activity1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name, Font Size, Bold, Italic, and Font Color . Default settings are Tahoma 8 point black , with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned Documentation Field .
KPIs	KPIs for <Step Name>	Step-level KPI (Key Performance Indicator) definitions. For more information about the fields on this page and how to work with them, see “About KPIs” on page 372 .
	KPI Properties	Information used to describe a KPI: Name, Description, Unit of Measure, Associated Field, Aggregation Type, and Dimensions . See “About Step-Level KPIs” on page 375 .
	Name	Name of the KPI Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names on the target server. However, if you duplicate a KPI name in a process, generation of the process produces a warning.
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.

Properties Page	Property	Description
	Associated Field	<p>Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used.</p> <p>Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.</p>
	Aggregation Type	<p>Aggregation type can be set to SUM, AVERAGE, or LAST VALUE</p>
	Dimensions	<p>Use this area to add, delete, and manage data dimensions for a selected KPI. You can associate a field from the step output with one or more dimensions.</p> <p>Note: Dimension names must be unique on a given Optimize server. Designer cannot, at design time, detect all the dimension names on the target server. However, if you duplicate a dimension label in a process, generation of the process produces a warning.</p> <p>For more information, see “About KPIs” on page 372.</p>
Inputs / Outputs	Inputs	<p>Data flowing into the step. You can add and remove inputs manually, as well as add or update from the service signature. For more information, see “Log Inputs and Outputs” on page 93.</p> <p>This works for call activities (including webMethods referenced process behavior), abstract tasks, service tasks, user tasks, rule tasks, and send tasks.</p>
	Outputs	<p>Data flowing out of the step. You can add and remove outputs manually, as well as add or update from the service signature.</p>

Properties Page	Property	Description
		This works for call activities (including webMethods referenced process behavior), abstract tasks, service tasks, user tasks, rule tasks, and receive tasks.
Transitions	Transition Type	<p>Defines the transition behavior. Options are If Condition, Default, Join Timeout, Iterations Exceeded, and Unsatisfied Join.</p> <p>For more information about these transition types and their configuration fields, see:</p> <ul style="list-style-type: none"> ■ “Configuring Transition Behavior” on page 125. ■ “About Transition Type Behavior” on page 118. ■ “About Transition Types” on page 118. ■ “How Incomplete Transitions Affect Join Steps and Gateways” on page 132.

Advanced Call Activity Properties

The following table describes the advanced call activity properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name (logical server).
	Type	<p>The type of process to be invoked. Choose BPMN Callable Process (default) or webMethods Referenced Process (Deprecated).</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The ability to invoke a webMethods referenced process from a call activity step is deprecated. You are advised to implement all call activity step invocations with a callable process.</p> </div> <p>BPMN Callable Process</p> <p>Click the Browse button in the Process Name/Process ID area to select a statically-invoked BPMN callable process. Designer populates the Process Name and</p>

Properties Page	Property	Description
		Process ID fields in the call activity. For more information, see “Call Activity Concepts” on page 147 .
	webMethods Referenced Process (Deprecated)	Click the Browse button in the Process Name/Process ID area to select a statically-invoked webMethods referenced process. Designer populates the Process Name and Process ID fields. For more information, see “Call Activity Concepts” on page 147 .
	Allow this step to dynamically invoke one or more processes at run time	<p>Select the check box to invoke one or more processes dynamically.</p> <ul style="list-style-type: none"> ■ If the Type selection is BPMN Callable Process, only callable processes can be invoked. ■ If the Type selection is webMethods Referenced Process (Deprecated), only webMethods referenced processes can be invoked. <p>For more information about dynamically invoking process in general, see “About Dynamically-Invoked Processes” on page 151.</p>
	Generated Service Name	Name of the generated flow service used at run time.
	Retry Count	<p>Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries. This field is not displayed when configuring a dynamic referenced process.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an <code>EXIT</code> with a <code>FAILURE</code> error, causes the step to fail.</p> </div>
	Retry Interval	Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or

Properties Page	Property	Description
		60 seconds. This field is not displayed when configuring a dynamic referenced process.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	Compensating	Places a BPMN compensation marker on the step, indicating that it is used for compensation. This is notational only and has no effect on the behavior of the activity.
Logged Fields	Inputs	Input fields to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93 .
	Outputs	Output fields to log. Outputs are logged after the step executes.
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>

Properties Page	Property	Description
	Join Condition	<p>Use the join condition editor to define a condition for the gateway join. This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>
Start / Return Documents	Available Input Documents	From the Available Input Documents list, select an input document. Use the Add >> and Remove << buttons to move it to the Document Inputs list.
	Document Inputs	This becomes the Start Document for the referenced process.
	Available Output Documents	From the Available Output Documents list, select an output document. Use the Add >> and Remove << buttons to move it to the Document Outputs list.
	Document Outputs	This becomes the Return Document for the referenced process.
Loop	Loop Expression	Define the loop expression using the table and provided operators.

Properties Page	Property	Description
		For more information, see “Configuring a Call Activity Standard Loop” on page 254.
	Test Loop Expression	Select one of the following options to define if the loop expression is evaluated before the step activity begins, or after the step activity completes: before activity or after activity
	Maximum Loop Count	Set the maximum number of loop iterations using a static value or a field.

13 BPMN Event Steps

■ About BPMN Events	264
■ Adding an Event to a Process	265
■ Removing an Event from a Process	267
■ About Start Events	267
■ About Intermediate Events	275
■ About End Events	298
■ Basic Event Properties	310

About BPMN Events

A BPMN event represents something that happens during the execution of a process. An event affects the flow of the process and is usually triggered by some action in the process. In most cases, execution of the event results in some kind of action.

Before you work with BPMN events in Software AG Designer, you should be familiar with the BPMN 2.0 specification in general. For more information, see the BPMN specification at <http://www.omg.org/spec/BPMN/2.0/>.

In BPMN, the following primary event types are available:

- Start events
- Intermediate events
- End events

Each of these primary events can be expressed as various secondary event types. For example, a none start event, or a terminate end event. Software AG Designer does not support all of the event types that are defined in the BPMN 2.0 specification. The event types listed in the following table are supported.

Table Legend

Column

1	Event type
2	Standard
3	Subprocess Interrupting
4	Subprocess Non-interrupting
5	Catching
6	Boundary interrupting
7	Boundary Non-interrupting
8	Throwing
9	Standard

Entry

X Not defined in the BPMN specification.

NS Not supported by Software AG Designer.

Event types that are entirely unsupported are not shown.

Large icons indicate event step types. Small icons indicate event boundary types.

The following table lists the event type icons.

1	Start			Intermediate			End	
	2	3	4	5	6	7	8	9
None		X	X	X	X	X		
Message		NS	NS					
Timer	NS	NS	NS	NS			X	X
Error	X	NS	X	X		X	X	
Signal		NS	NS					
Terminate	X	X	X	X	X	X	X	

Software AG Designer icon symbology conforms to [BPMN 2.0 usage](#) in form and line. However, the colors applied to some events may not be used by other modeling tools.

In general, the behavior of BPMN events in Software AG Designer conforms to the BPMN 2.0 specification.

For specific information about the implementation of these events in Software AG Designer, see the following topics:

- Start events: [“About Start Events” on page 267.](#)
- Intermediate events: [“About Intermediate Events” on page 275.](#)
- End events: [“About End Events” on page 298.](#)

About Throwing and Catching Events

Some of the BPMN intermediate events supported by Software AG Designer can be configured as a catching event or a throwing event, as shown in [“About BPMN Events” on page 264.](#)

You can configure catching and throwing behavior on the **General** page in the Properties view. For more information, see [“Configuring an Intermediate Event” on page 275.](#) If an event cannot be configured for catching or throwing, these properties are not available.

Adding an Event to a Process

Note:

If you want to add a boundary event to an existing activity or subprocess event, see [“Adding a Boundary Intermediate Event” on page 289.](#)

> To add an event to a process

1. Open the process model you want to work with in the process editor and do one of the following:
 - Drag the event you want to add from the palette to the design canvas, or, in the process editor palette, click the event type you want to add, then click in the design canvas where you want to place the new event. You must use one of these methods the first time you add an event to an empty canvas.
 - Drag an EDA (deprecated) event from the Registry Explorer, Package Navigator, or Search view onto the process editor canvas to create a Message Start Event.
 - Use the palette view to add a BPM event step. You can also drag and drop an EDA (deprecated) event onto an existing None Start, Message Start, Message Intermediate, or Message End Event. The target event step is automatically associated with the document dropped onto it.
 - Hover the cursor over an existing event on the canvas to display the speed buttons. Click the speed button for the event type you want to add, and then click the canvas at the location where you want to add the event. A transition line is created automatically from the source event to the target event.

Note:

Start events cannot be created from the speed buttons. The purpose of the speed buttons is to be able to add a process object such as an activity, event, or gateway, with an automatic transition from the source object. Because start events have no incoming transitions, they are not available as a speed button.

Tip:

If the event is not the one you want, right-click the event and click **Change Event Type** to specify a different event type.

Tip:

You can set the label of the event to be empty, or to have the same name as another step in the process. You can also change the placement of the label. For more information, see [“About Step Labels” on page 72](#).

2. Add transition lines if needed and configure the event as necessary in the Properties view.
3. Save the process.

Note:

You can change the event type after you add it. For more information, see [“Changing an Activity, Event, or Gateway Type” on page 83](#).

Removing an Event from a Process

➤ To remove an event from a process

1. Open the process model you want to work with in the process editor.
2. On the design canvas, click the event you want to remove to select it.
3. Press the Delete key.
4. Save the process.

About Start Events

Software AG Designer supports the following start event types:

- None start events: [“About None Start Events” on page 268.](#)
- Message start events: [“About Message Start Events” on page 269.](#)
- Signal start events: [“About Signal Start Events” on page 272.](#)

Configuring a Start Event

Start events are configured on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52.](#)

➤ To configure a BPMN start event

1. In an open process, click the event you want to configure to select it.
2. In the Properties view, do one of the following:
 - Configure the basic event properties described in [“Basic Event Properties” on page 310.](#)
 - Configure the advanced start event properties described in one of the following topics:
 - [“Advanced None Start Event Properties” on page 268.](#)
 - [“Advanced Message Start Event Properties” on page 269.](#)
 - [“Advanced Signal Start Event Properties” on page 272.](#)
3. Save the process.

About None Start Events

A none start event is not capable of starting a process instance. A none start event represents the beginning of the flow in a BPMN *callable process*, which is called by a BPMN call activity.

If you implement a call activity, you *must* use a none start event in the targeted BPMN callable process. A none start event is not associated with a receive document. As other none events (start and end), it has no generated wrapper service.

Tip:

You can convert a None Start Event to a Message Start Event by dragging a document from the Registry Explorer, Package Navigator, or Search view, and dropping it onto a None Start Event. The Message Start Event is automatically associated with the document dragged to the step.

Advanced None Start Event Properties

You must be working in Process Developer mode to configure the properties described in the following table.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name . Note: When a callable process contains both a none start event and a message start event that is configured to use an EDA (deprecated) event type, the message start step and the none start step of the child process must be on different Integration Servers (that is, the Integration Server Name property must be different for each of the two steps). Otherwise a validation error occurs during the build and upload procedure.
	Receive Protocol	Type of receive protocol to use. Select Subscription (For Publishable Documents) , JMS (for JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) from the list.
	Protocol Properties	Available for JMS and EDA (deprecated) protocols only. ■ Connection Alias . For the JMS protocol, accept the default PE_NONTRANSACTIONAL_ALIAS or click Browse to select another alias (an

Properties Page	Property	Description
		Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. For the EDA (deprecated) protocol, accept the default EventBus alias or click Browse to select another alias.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.

About Message Start Events

A Message Start Event starts a process by receiving a message that conforms to the document type associated with the Message Start Event.

Tip:

You can convert a None Start Event to a Message Start Event by dragging a document from the Registry Explorer, Package Navigator, or Search view, and dropping it onto a None Start Event. The Message Start Event is automatically associated with the document dragged to the step.

If you drag a Message Start Event onto an Abstract, Service, User, or Rule task activity, or onto a Call Activity, it is added to the activity step as a non-interrupting Boundary Message Intermediate Event.

Software AG Designer extends BPMN functionality to support the use of EDA (deprecated) event types and the EDA (deprecated) protocol in BPMN Message Events.

Advanced Message Start Event Properties

You must be working in Process Developer mode to configure the properties described in the following table.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name . Note: When a callable process contains both a none start event and a message start event that is configured to use an EDA (deprecated) event type, the message start step and the none start step of the child process must be on different Integration Servers (that is, the Integration Server Name property must be different for each

Properties Page	Property	Description
	Receive Protocol	<p>of the two steps). Otherwise a validation error occurs during the build and upload procedure.</p> <p>Type of receive protocol to use. Select Subscription (For Publishable Documents) or JMS (for JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) from the list.</p>
	Receive Document	<p>Note: Dragging and dropping an event type automatically configures the Receive Protocol to use EDA (For EDA Event Triggered Processes).</p> <p>The document type to receive.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <p>Note: If you created the step by dropping an IS document or event type onto the canvas, or if you drop an IS document or event type onto an existing message start step, this field is automatically populated with the IS document name.</p> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p>

Properties Page	Property	Description
	Receive EDA Event	For the EDA (deprecated) protocol only. The Integration Server document type associated with the EDA (deprecated) event type is displayed when using EDA (For EDA Event Triggered Processes) as the Receive Protocol . This field is not editable.
	Protocol Properties	<p>Available for JMS and EDA (deprecated) protocols only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRASACTIONAL_ALIAS for JMS or EventBus for EDA, or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450 and “Configuring an EDA-Triggered Process” on page 452. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Subscription Filter	Available for the Subscription and JMS protocols only. The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444 .
	Generate Service	<p>Selected by default to generate a wrapper service. If cleared, no wrapper service is generated. This improves Process Engine performance.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: If a wrapper service already exists, selecting this option <i>deletes the existing service</i>.</p> </div>
	Generated Service Name	Name of the generated flow service used at run time
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	E-form	Selecting an e-form as the step's Receive Document automatically selects the Enable

Properties Page	Property	Description
		<p>content repository support check box and enables E-form properties.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task.</p> </div>
	Content Repository	Select Browse to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	<p>Selecting an e-form as the step's Receive Document automatically populates this field.</p> <p>For more information about using e-forms, see “Using E-forms in a Process” on page 69.</p>
Logged Fields	Outputs	Output fields to log. Outputs are logged after the step executes. See “Log Inputs and Outputs” on page 93.
Correlation	Correlation	Select Not Used , Field , or Service . For more information, see “Specifying Correlations” on page 165.

About Signal Start Events

A signal start event starts a process by receiving a message.

If you drag a signal start event onto an abstract, service, user, or rule task activity, or onto a call activity, it is added to the activity as a non-interrupting signal boundary intermediate event.

Advanced Signal Start Event Properties

You must be working in Process Developer mode to configure the properties described in the following table.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Receive Protocol	Type of receive protocol to use. Select Subscription (For Publishable Documents) or JMS (for JMS Triggered Processes) (default) from the list.
	Receive Document	<p>The document type to receive.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> </div>
	Protocol Properties	<p>Available for the JMS protocol only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Subscription Filter	The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444.

Properties Page	Property	Description
	Generate Service	<p>Selected by default to generate a wrapper service. If cleared, no wrapper service is generated. This improves Process Engine performance.</p> <p>Important: If a wrapper service already exists, selecting this option <i>deletes the existing service</i>.</p>
	Generated Service Name	Name of the generated flow service used at run time
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p> <p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task.</p>
	Content Repository	Select Browse to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	<p>Selecting an e-form as the step's Receive Document automatically populates this field.</p> <p>For more information about using e-forms, see “Using E-forms in a Process” on page 69 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
Logged Fields	Outputs	Output fields to log. Outputs are logged after the step executes. See “Log Inputs and Outputs” on page 93 .

Properties Page	Property	Description
Correlation	Correlation	Select Not Used , Field , or Service . For more information, see “Specifying Correlations” on page 165 .

About Intermediate Events

Intermediate events do not start or end activities; they execute during the flow of a process. They can exist as top-level entities in a process, or they can exist on the boundaries of activities as boundary intermediate events.

Software AG Designer supports the use of the EDA (deprecated) protocol with BPMN Message Events as a BPMN extension.

You can configure the following behavior for intermediate events:

- Intermediate events: throwing or catching behavior for message intermediate events and signal intermediate events.
- Boundary intermediate events: interrupting or non-interrupting behavior for boundary message events, boundary signal events, and boundary timer events.

Configuring an Intermediate Event

Intermediate events are configured on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

Note:

To configure boundary intermediate events, see [“Configuring a Boundary Intermediate Event” on page 290](#).

➤ To configure a BPMN intermediate event

1. In an open process, click the event you want to configure to select it.
2. In the Properties view, do one of the following:
 - Configure the basic event properties described in [“Basic Event Properties” on page 310](#).
 - Configure the advanced intermediate event properties described in one of the following topics:
 - [“Advanced None Intermediate Event Properties” on page 276](#).
 - [“Advanced Message Intermediate Event Properties” on page 278](#).

- [“Advanced Signal Intermediate Event Properties” on page 283.](#)

3. Save the process.

About None Intermediate Events

In conformance with the BPMN 2.0 specification, a none intermediate event is a throwing event by default and cannot be configured as a catching event. A none intermediate event is typically used to show that some kind of event is thrown in the process, but is bypassed in the run time, similar to an abstract task.

A throwing none intermediate event does not actually throw anything. It has no associated flow service and no Process Engine implementation. As other none events (start and end), it has no generated wrapper service.

Advanced None Intermediate Event Properties

You must be working in Process Developer mode to configure the properties described in the table below.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name
	Receive Protocol	Type of receive protocol to use. Select Subscription (For Publishable Documents) , JMS (for JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) from the list.
	Protocol Properties	Available for JMS and EDA (deprecated) protocols only. <ul style="list-style-type: none"> ■ Connection Alias. For the JMS protocol, accept the default PE_NONTRANSACTIONAL_ALIAS or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. For the EDA (deprecated) protocol, accept the default EventBus alias or click Browse to select another alias.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming

Properties Page	Property	Description
		<p>transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>See “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

Properties Page	Property	Description
		<p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Message Intermediate Events

A message intermediate event throws (sends) or catches (receives) a message. It cannot start a process instance.

A catching message intermediate event results in the generation of a trigger condition that is always active. In other words, a published message that matches the condition of the catching message intermediate event is always received and held until such time as the sequence flow(s) are received.

In BPMN terms, this is described as the step or subprocess being *activated*. In strict BPMN terms, a catching message intermediate event should not fire until the inbound sequence flow has reached that step. It can be argued that the current implementation does not adhere strictly to BPMN 2.0 because the message is received and held before the sequence flow reaches the catching message intermediate event.

However, strict adherence to BPMN 2.0 in this case could result in unwanted results, such as race conditions and lost messages. This interpretation of BPMN 2.0 avoids these potential problems.

Software AG Designer extends BPMN functionality to support the use of EDA (deprecated) event types and the EDA (deprecated) protocol in BPMN Message Events.

Note:

When creating a callable process that is to be called by a call activity step, always implement a send task in the callable process to return to the parent process. Message intermediate events and message end events will only publish a document and will not return it to the parent.

Advanced Message Intermediate Event Properties

You must be working in Process Developer mode to configure the properties described in the following table.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Receive Protocol or	For a catching event, identifies the type of receive protocol to use.

Properties Page	Property	Description
	Send Protocol	<p>For a throwing event, identifies the type of send protocol to use.</p> <p>Select Subscription (For Publishable Documents), JMS (for JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) from the list.</p> <p>Note: Dragging and dropping an EDA Event Type automatically configures the Receive Protocol to use EDA (For EDA Event Triggered Processes)</p>
	Receive Document or Send Document	<p>For a catching event, identifies the document type to receive.</p> <p>For a throwing event, identifies the document type to send.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <p>Note: If you drop an IS document or event type onto an existing intermediate message step, this field is automatically populated with the IS document name.</p> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p>
	Receive EDA Event	For the EDA (deprecated) protocol only.

Properties Page	Property	Description
	or Send EDA Event	<p>For a catching event, identifies the document type to receive.</p> <p>For a throwing event, identifies the document type to send.</p> <p>The Integration Server document type associated with the EDA (deprecated) event type is displayed when using EDA (For EDA Event Triggered Processes) as the Receive Protocol or the Send Protocol. This field is not editable.</p>
	Protocol Properties	<p>Available for JMS and EDA (deprecated) protocols only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS for JMS or EventBus for EDA, or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450 and “Configuring an EDA-Triggered Process” on page 452. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Subscription Filter	<p>For catching events only, not available for EDA (deprecated) protocol: The instances of a subscription document or event that can trigger the process. See “Working with Subscription Filters” on page 444.</p>
	Generated Service Name	<p>For catching only: Name of the generated flow service used at run time</p>
	Retry Count	<p>For catching only: Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The retry mechanism is invoked only when a step service generates an</p> </div>

Properties Page	Property	Description
		ISRuntimeException error. Any other exception, such as an EXIT with a FAILURE error, causes the step to fail.
	Retry Interval	For catching only: Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
		<p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive step.</p>
	Content Repository	Select Browse to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	<p>Selecting an e-form as the step's Receive Document automatically populates this field.</p> <p>For more information about using e-forms, see “Using E-forms in a Process” on page 69 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
Logged Fields	Outputs	For catching only: Output fields to log. Outputs are logged after the step executes. For more information, see “Log Inputs and Outputs” on page 93

Properties Page	Property	Description
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>For more information, see “About Joins” on page 131.</p> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>See “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p>

Properties Page	Property	Description
		This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
		Ignore dead path notification
		This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
Correlation	Correlation	For catching only: select Not Used , Field , or Service . For more information, see “Specifying Correlations” on page 165.

About Signal Intermediate Events

A signal intermediate event receives a signal. It cannot start a process instance. A signal intermediate event can be configured as catching or throwing.

A catching signal intermediate event results in the generation of a trigger condition that is always active. In other words, a published signal that matches the condition of the catching signal intermediate event is always received and held until such time as the sequence flow(s) are received.

In BPMN terms, this is described as the step or subprocess being activated. In strict BPMN terms, an catching signal intermediate event should not fire until the inbound sequence flow has reached that step. It can be argued that the current implementation does not adhere strictly to BPMN 2.0 because the signal is received and held before the sequence flow reaches the catching signal intermediate event.

However, strict adherence to BPMN 2.0 in this case could result in unwanted results, such as race conditions and lost signals. This interpretation of BPMN 2.0 avoids these potential problems.

Advanced Signal Intermediate Event Properties

You must be working in Process Developer mode to configure the properties described in the following table.

Properties Page	Property	Description
General	Throw Event	Select this option to implement throwing (sending) behavior.
	Catch Event	Select this option to indicate catching (receiving) behavior. Selected by default.
Implementation	Integration Server Name	Assigned Integration Server Name .

Properties Page	Property	Description
	Receive Protocol or Send Protocol	<p>For a catching event, identifies the type of receive protocol to use.</p> <p>For a throwing event, identifies the type of send protocol to use.</p> <p>Select Subscription (For Publishable Documents), or JMS (for JMS Triggered Processes) (default) from the list.</p> <p>Note: Dragging and dropping an Event Type automatically configures the Receive Protocol to use EDA (For EDA Event Triggered Processes).</p>
	Receive Document or Send Document	<p>For a catching event, identifies the document type to receive.</p> <p>For a throwing event, identifies the document type to send.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p>
	Protocol Properties	<p>Available for the JMS protocol only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS or click Browse to select another alias (an Integration Server connection is required).

Properties Page	Property	Description
		<p>For more information, see “Configuring a JMS-Triggered Process” on page 450.</p> <ul style="list-style-type: none"> ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Subscription Filter	For a catching event only. The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444.
	Generated Service Name	Name of the generated flow service used at run time.
	Retry Count	For catching only: Number of times the Process Engine retries the service in the case of an Integration Server run-time exception. The default value is 0, or no retries.
		<p>Note: The retry mechanism is invoked only when a step service generates an <code>ISRuntimeException</code> error. Any other exception, such as an EXIT with a FAILURE error, causes the step to fail.</p>
	Retry Interval	For catching only: Number of milliseconds (ms) the Process Engine waits between retry attempts in the case of an Integration Server run-time exception. The default value is 60000 ms, or 60 seconds.
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
	E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>

Properties Page	Property	Description
		<p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task.</p>
	Content Repository	Select Browse to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	<p>Selecting an e-form as the step's Receive Document automatically populates this field.</p> <p>For more information about using e-forms, see “Using E-forms in a Process” on page 69 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
Logged Fields	Outputs	For catching only: Output fields to log. Outputs are logged after the step executes.
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p>
	Join Timeout	<p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> <p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and

Properties Page	Property	Description
		minutes, or specify a pipeline field to set the day, hours, or minutes value. See “About Join Timeouts” on page 141 .
	Join Condition	Use the join condition editor to define a condition for the join. This option is available only for a complex join. When the terms of this condition are met (that is, the condition is true), the join is considered satisfied. For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138 .
	Deprecated properties (not recommended)	These deprecated properties are not available for unsynchronized OR joins. Suppress Join Failure This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134 . Ignore dead path notification This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134 .
Correlation	Correlation	For catching only: select Not Used , Field , or Service . For more information, see “Specifying Correlations” on page 165 .

About Boundary Intermediate Events

Designer supports a number of boundary intermediate events. Not all boundary intermediate events are configurable as both interrupting or non-interrupting in all cases. For more information about supported event types and their behavior, see [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

Note:

The palette does not contain an entry for all possible boundary events. For more information, see [“Adding a Boundary Intermediate Event” on page 289](#).

Important:

You cannot configure interrupting boundary event behavior for call activities, rule tasks, or user tasks. No boundary intermediate events are available for manual task activities.

About Interrupting Behavior for Boundary Intermediate Events

Software AG Designer follows the BPMN specification for interrupting behavior, although not all event types or interrupting behaviors are supported. The following table describes the available interrupting behavior, indicated with an **X** entry. No entry means the event type is not available.

The table below lists those behaviors.

Boundary Intermediate Event Type	Activity Type										
	Abstract	Service	Receive	Send	User	Rule	Manual	webM Sub.	BPMN Sub.	Call Act.	webM Ref. Proc
Timer Interrupting	X*	X*	X*	X*					X*		
Timer Non-Interrupting	X	X	X	X		X*		X*		X*	X*
Error Interrupting	X*	X*	X*	X*	X*	X*			X*	X*	X*
Error Non-Interrupting								X*			
Message Interrupting	X*	X*									
Message Non-Interrupting	X	X			X*	X*				X*	X*
Signal Interrupting	X*	X*									
Signal Non-Interrupting	X*	X			X*	X*		X*		X*	X*

* Default.

About Throwing and Catching Boundary Intermediate Events

For a boundary event to catch an externally triggered event from a different track or process, the associated activity must be active. For example, if a catching boundary message intermediate event is added to a call activity, the call activity must be active before the boundary event can catch a message.

This is also true when debugging. Process Debug runs only one track at a time, meaning only one track is active at any given moment. Therefore, an activity in one track will never be active when another track is being debugged.

This situation can also conceivably occur in the run time if two or more processes are interacting, and one of the processes is not running.

Adding a Boundary Intermediate Event

➤ To add a boundary intermediate event to an activity

1. Open the process model you want to work with in the process editor and do one of the following:
 - Right-click the activity you want to work with, click **Add Boundary Event**, and click the desired boundary event in the resulting menu. Only those boundary events that apply to the selected activity are available. For example, a boundary timer intermediate event is not available for a user task.
 - Drag a boundary event from the palette to an activity. If the boundary event is not supported by the activity, you will not be allowed to drop the event. Not all boundary events are available from the palette. See the note below.
 - In the process editor palette, click a boundary event type, then click the activity in the process editor where you want to place the new boundary event.

Note:

The palette does not contain an entry for all possible boundary events. If you drag any of the following events onto an abstract, service, user, or rule task activity, or onto a call activity, it is added to the activity as noted in the following table.

If you drag this to the activity	This event type is added to the activity
Message start event	Non-interrupting boundary message intermediate event.
Signal start event	Non-interrupting boundary signal intermediate event.
Message end event	Non-interrupting boundary message intermediate event.
Signal end event	Non-interrupting boundary signal intermediate event.
Error end event	Interrupting boundary error intermediate event.

2. Add any transition lines and configure the boundary event as necessary in the Properties view.
3. Save the process.

Note:

You can change the boundary event type after you add it. For more information, see [“Changing an Activity, Event, or Gateway Type” on page 83](#).

Removing a Boundary Intermediate Event

➤ To remove a boundary intermediate event from a step

1. Open the process model you want to work with in the process editor.
2. Click the boundary intermediate event you want to remove to select it.
3. Press the Delete key.
4. Save the process.

Configuring a Boundary Intermediate Event

You configure a boundary intermediate event on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52](#).

Note:

To configure intermediate events, see [“Configuring an Intermediate Event” on page 275](#).

➤ To configure a BPMN boundary intermediate event

1. In an open process, click the boundary intermediate event you want to configure to select it.
2. In the Properties view, do one of the following:
 - Configure the basic event properties described in [“Basic Boundary Intermediate Event Properties” on page 296](#).
 - Configure the advanced none start event properties described in:
 - [“Advanced Boundary Message Intermediate Event Properties” on page 292](#).
 - [“Advanced Boundary Signal Intermediate Event Properties” on page 294](#).

Note:

There are no advanced properties for:

- A boundary timer intermediate event.
- A boundary error intermediate event.

3. Save the process.

About Boundary Intermediate Timer Events

Boundary intermediate timer events can be *interrupting* or *non-interrupting* depending on the activity type. The supported types are described in [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

You can configure the interrupting behavior with the **Allow Boundary Timer Event to interrupt activity** check box on the **General** page in the Properties view, except as noted in the supported event types table referenced above.

Note:

There are no advanced properties for a boundary timer intermediate event.

Non-configurable interrupting behavior provides consistency with the timeout transition behavior of a non-BPMN referenced process step. For example, it is possible for a call activity to follow its timeout transition and to also follow the default transition, as the boundary intermediate timer event does not interrupt.

About Boundary Intermediate Error Events

Boundary intermediate error events always interrupt, except for webMethods subprocesses (deprecated), where they are always non-interrupting. There is no option to configure the interrupting behavior of a boundary error intermediate event. This behavior is fixed and cannot be modified by the user.

If a boundary intermediate error event is not present, the step transitions using the process error handler step mechanism. For more information about the error handler step, see [“Advanced Process Properties” on page 61](#).

There are no advanced properties for a boundary error intermediate event.

About Boundary Message Intermediate Events

Boundary message intermediate events inherit **Integration Server Name** from the step to which they are attached. They can be *interrupting* or *non-interrupting* depending on the activity type. The supported types are described in [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

You can configure the interrupting behavior with the **Allow Boundary Message Event to interrupt activity** check box on the **General** page in the Properties view, except as noted in the supported event types table referenced above.

For a boundary event to catch an externally triggered event from a different track or process, the associated activity must be active.

Important:

Boundary message events cannot be debugged directly for an externally-created message event. When a message event is transmitted from outside a process that is being debugged *and* there is a boundary message event in the debugged process that is configured to catch and correlate

to the transmitted message, that message is *not* recognized by the Process Engine as being associated with the process currently being debugged.

Software AG Designer extends BPMN functionality to support the use of EDA (deprecated) event types and the EDA (deprecated) protocol in BPMN Message Events.

Advanced Boundary Message Intermediate Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Receive Protocol	<p>Identifies the type of receive protocol to use.</p> <p>Select Subscription (For Publishable Documents), JMS (for JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) from the list.</p> <p>Note: Dragging and dropping an EDA Event Type automatically configures the Receive Protocol to use EDA (For EDA Event Triggered Processes)</p>
	Receive Document	<p>Identifies the document type to receive.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <p>Note: If you drop an IS document or event type onto an existing boundary message intermediate step, this field is automatically populated with the IS document name.</p> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable</p>

Properties Page	Property	Description
	Receive EDA Event	<p>content repository support check box and enables E-form properties.</p> <p>For the EDA (deprecated) protocol only. Identifies the document type to receive. The Integration Server document type associated with the EDA (deprecated) event type is displayed when using EDA (For EDA Event Triggered Processes) as the Receive Protocol. This field is not editable.</p>
	Protocol Properties	<p>Available for JMS and EDA (deprecated) protocols only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS for JMS or EventBus for EDA, or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450 and “Configuring an EDA-Triggered Process” on page 452. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p> <p>Important: Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task.</p>

Properties Page	Property	Description
	Content Repository	Select Browse... to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.
	Template Name	Selecting an e-form as the step's Receive Document automatically populates this field. For more information about using e-forms, see “Using E-forms in a Process” on page 69 and the PDF publication <i>Implementing E-form Support for BPM</i> .
	Subscription Filter	The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444 .
Correlation	Correlation	Select Not Used , Field , or Service . For more information, see “Specifying Correlations” on page 165 .

About Boundary Signal Intermediate Events

Boundary signal intermediate events inherit **Integration Server Name** settings from the step to which they are attached. They can be *interrupting* or *non-interrupting* depending on the activity type. The supported types are described in [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#).

You can configure the interrupting behavior with the **Allow Boundary Signal Event to interrupt activity** check box on the **General** page in the Properties view, except as noted in the supported event types table referenced above.

For a boundary event to catch an externally triggered event from a different track or process, the associated activity must be active.

Important:

Boundary signal events cannot be debugged directly for an externally-created signal event. When a signal event is transmitted from outside a process that is being debugged *and* there is a boundary signal event in the debugged process that is configured to receive and correlate to the transmitted signal, that signal is *not* recognized by the Process Engine as being associated with the process currently being debugged.

Advanced Boundary Signal Intermediate Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Receive Protocol	Type of receive protocol to use. Select Subscription (For Publishable Documents) or JMS (for JMS Triggered Processes) (default) from the list.
	Receive Document	<p>The document type to receive.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> </div>
	Protocol Properties	<p>Available for the JMS protocol only.</p> <ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRANSACTIONAL_ALIAS or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
E-form	<p>Selecting an e-form as the step's Receive Document automatically selects the Enable content repository support check box and enables E-form properties.</p> <p>Receive tasks that start processes and those that do not start processes can use e-forms.</p>	

Properties Page	Property	Description
		<p>For more information on working with e-forms, see “Selecting an E-form Content Repository” on page 70 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
	Content Repository	<p>Select Browse... to specify a My webMethods Server content repository where e-form instances are monitored by the Process Engine listener.</p>
	Template Name	<p>Selecting an e-form as the step's Receive Document automatically populates this field.</p> <p>For more information about using e-forms, see “Using E-forms in a Process” on page 69 and the PDF publication <i>Implementing E-form Support for BPM</i>.</p>
	Subscription Filter	<p>The instances of a subscription document that can trigger the process. See “Working with Subscription Filters” on page 444.</p>
Correlation	Correlation	<p>Select Not Used, Field, or Service. For more information, see “Specifying Correlations” on page 165.</p>

Important:

Incoming e-form instances must have the correct file name extension (for example, .xml for InfoPath forms, .xdp for LiveCycle forms). Otherwise, they will not trigger the receive task.

Basic Boundary Intermediate Event Properties

The following table describes these properties.

Properties Page	Property	Description
General	Type	Select from the list. Available types are Timer , Error , Message , and Signal .
	Allow Boundary [type] Event to interrupt step	When selected, this property applies interrupting behavior to the event. If the interrupting behavior of the event cannot be

Properties Page	Property	Description
		changed, this property is either not present or is not selectable.
	Restore starting pipeline	Available only for an intermediate boundary error event on a subprocess. You can save the pipeline upon entering the subprocess. If a subprocess error occurs, the contents of the pipeline pass to the following step. Disabled by default.
	Label	Step name. Default is Message1 , Signal1 , Timer1 , or Error1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name , Font Size , Bold , Italic , and Font Color . Default settings are Tahoma 8 point black, with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned Documentation Field
Timer Condition		<p>For intermediate boundary timer events only. For more information about this event type, See “About Boundary Intermediate Timer Events” on page 291.</p> <p>Use this page to specify the value that defines the timer condition. The source of this timer value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds.

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Boundary Timer Event Timer Conditions” on page 143</p>

About End Events

You use an end event to end a process flow, either a single track within a process, or the entire process itself.

- You end a single track with an error end event, an message end event, or an signal end event. They throw errors, messages, and signals based on their event type.
- You end an entire process and all its tracks with a terminate end event. A terminate end event also has extensions that describe the process state upon termination: **Failed**, **Canceled**, or **Completed** (default). For more information, see [“Configuring Terminate End Event Status”](#) on page 310.

Configuring an End Event

You configure the behavior of an end event on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes”](#) on page 52.

> To configure a BPMN end event

1. In an open process, click the event you want to configure to select it.
2. In the Properties view, do one of the following:
 - Configure the basic event properties described in [“Basic Event Properties”](#) on page 310.
 - Configure the advanced end event properties described in one of the following topics:
 - [“Advanced None End Event Properties”](#) on page 299.
 - [“Advanced Error End Event Properties”](#) on page 300.
 - [“Advanced Message End Event Properties”](#) on page 302.
 - [“Advanced Signal End Event Properties”](#) on page 305.

- [“Advanced Terminate End Event Properties” on page 308.](#)

3. Save the process.

About None End Events

A none end event ends a track in a process. It allows input transitions, but no output transitions. As other none events (start and end), it has no generated wrapper service.

Advanced None End Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>“About Join Timeouts” on page 141</p>

Properties Page	Property	Description
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Error End Events

An error end event ends a track in a process and throws an error. It supports input transitions but no output transitions.

If you drag an error end event onto an abstract, service, user, or rule task activity, or onto a call activity, it is added to the activity as an interrupting boundary error intermediate event.

Advanced Error End Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.

Properties Page	Property	Description
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>“About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see</p>

Properties Page	Property	Description
		“Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
	Ignore dead path notification	This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.

About Message End Events

A message end event ends a track in a process and throws a message. Message end events have inputs but no outputs.

If you drag a message end event onto an abstract, service, user, or rule task activity, or onto a call activity, it is added to the activity as a non-interrupting message boundary intermediate event.

Software AG Designer extends BPMN functionality to support the use of EDA (deprecated) event types and the EDA (deprecated) protocol in BPMN Message Events.

Note:

When creating a callable process that is to be called by a call activity step, always implement a send task in the callable process to return to the parent process. Message end events and message intermediate events will only publish a document and will not return it to the parent.

Advanced Message End Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Send Protocol	The messaging protocol used to send the document. Select Subscription (For Publishable Documents) , JMS (For JMS Triggered Processes) (default), or EDA (For EDA Event Triggered Processes) .
	Send Document	The document type to send. <ul style="list-style-type: none"> Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite.

Properties Page	Property	Description
		<ul style="list-style-type: none"> Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field.
		<p>Note: If you drop an IS document or event type onto an existing message end step, this field is automatically populated with the IS document name.</p>
	Send EDA Event	For the EDA (deprecated) protocol only. The Integration Server document type associated with the EDA (deprecated) event type is displayed when using EDA (For EDA Event Triggered Processes) as the Send Protocol . This field is not editable.
	Protocol Properties	Available for JMS and EDA (deprecated) protocols only. <ul style="list-style-type: none"> Connection Alias. Accept the default PE_NONTRASACTIONAL_ALIAS for JMS or EventBus for EDA, or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450 and “Configuring an EDA-Triggered Process” on page 452. Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Allow Parallel Execution	Lock the step at run time to allow it to be executed by multiple threads.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming

Properties Page	Property	Description
		<p>transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see</p>

Properties Page	Property	Description
		“Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
	Ignore dead path notification	<p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Signal End Events

An signal end event ends a track in a process and throws a signal. Signal end events have inputs, but no outputs.

If you drag a signal end event onto an abstract, service, user, or rule task activity, or onto a call activity, it is added to the activity as a non-interrupting boundary signal intermediate event.

Advanced Signal End Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Send Protocol	The messaging protocol used to send the signal. Select Subscription (For Publishable Documents) or JMS (For JMS Triggered Processes) (default).
	Send Document	<p>The document type to send.</p> <ul style="list-style-type: none"> ■ Click Browse to open the Choose Document window and locate a service on an Integration Server or in CentraSite. ■ Click New to open the Create a New Document Type window and create a new document on a configured Integration Server. ■ Click View to open the selected document in a document editor. The View button is available only when a document is specified in the Document field.
	Protocol Properties	Available for the JMS protocol only.

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ Connection Alias. Accept the default PE_NONTRASACTIONAL_ALIAS or click Browse to select another alias (an Integration Server connection is required). For more information, see “Configuring a JMS-Triggered Process” on page 450. ■ Destination Name. Required. Defined by default as: <i>projectName_ProcessName_SUBQUEUE</i>. To specify another destination name, click Browse (an Integration Server connection is required).
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads.
Joins	Join Type	<p>A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR, AND, COMPLEX, or Unsynchronized OR.</p> <p>See “About Joins” on page 131.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Joins page is displayed only when a step has more than one incoming transition.</p> </div>
	Join Timeout	<p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Join Timeouts” on page 141.</p>

Properties Page	Property	Description
	Join Condition	<p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p>

About Terminate End Events

A terminate end event ends a process, including all its tracks. A terminate end event can also be used in a subprocess, but behavior varies. For more information, see [“About Terminate End Event Behavior in a Subprocess” on page 307](#).

When a terminate end event ends a process, the process termination status appears in webMethods Monitor for business activity monitoring (BAM) purposes. The status also appears on the event icon in the process editor canvas.

You can configure the termination status applied by the terminate end event in processes and webMethods subprocesses, as described in [“Configuring Terminate End Event Status” on page 310](#).

Note: **Process status upon termination** options are not BPMN constructs; they are webMethods constructs used in a BPMN extension.

About Terminate End Event Behavior in a Subprocess

The behavior of a terminate end event in a process varies depending on the type of subprocess that contains the terminate end event:

- **BPMN subprocess.** When an terminate end event is located in a BPMN subprocess, the step *always* terminates the subprocess with no direct effect on the parent process. When the subprocess is terminated in this way, no other new activities in the subprocess are executed. The subprocess status changes to Complete and the output transitions from the subprocess are taken as if the subprocess completed normally.
- **webMethods subprocess.** When an terminate end event is located in a webMethods subprocess, the step *always* terminates the parent process and applies a configurable status of Completed, Canceled, or Failed. You set the status configuration in the advanced properties for the terminate end event. See [“Configuring an End Event” on page 298](#) and [“Advanced Terminate End Event Properties” on page 308](#). The subprocess does not take any of its output transitions.

Advanced Terminate End Event Properties

The following table describes these properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name .
	Process status upon termination	Status displayed in webMethods Monitor: Completed (default), Canceled , or Failed . Note: This control is unavailable for terminate end events within a BPMN subprocess. For more information, see “About Terminate End Event Behavior in a Subprocess” on page 307 and “Configuring Terminate End Event Status” on page 310 .
	Escalate failed process status to parent	Available for the status of Failed only. Alert parent process of the Failed status. Selected by default.
	Allow Parallel Execution	Lock the step at run time to allow it to be executed by multiple threads.
Joins	Join Type	A join type defines the logic the process follows when a step has multiple incoming transitions. Options: OR , AND , COMPLEX , or Unsynchronized OR . See “About Joins” on page 131 . Note:

Properties Page	Property	Description
	Join Timeout	<p>The Joins page is displayed only when a step has more than one incoming transition.</p> <p>The timeout duration value, after which this join fails. This option is available for AND and COMPLEX joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value.
	Join Condition	<p>For more information, see “About Join Timeouts” on page 141.</p> <p>Use the join condition editor to define a condition for the join. This option is available only for a complex join.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138.</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are not available for unsynchronized OR joins.</p> <p>Suppress Join Failure</p> <p>This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.</p> <p>Ignore dead path notification</p> <p>This option is not recommended except in very isolated cases. For more information, see</p>

Properties Page	Property	Description
		“Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.

Configuring Terminate End Event Status

You configure the behavior of a terminate end event on the pages of the Properties view.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see [“About Process Development Modes” on page 52.](#)

When you add an terminate end event to a process or to a webMethods subprocess, you can specify a status to be applied to the process by the terminate end event. For more information about subprocess behavior, see [“About Terminate End Event Behavior in a Subprocess” on page 307.](#) The following status selections are available:

- **Completed.** This indicates the process successfully ran to completion. In this case, Designer applies the standard terminate end event icon.
- **Canceled.** This indicates the process was canceled. In this case, Designer applies an exclamation point (!) inside a triangle to the terminate end event icon.
- **Failed.** This indicates the process failed. In this case, Designer applies a circle with a diagonal line through it from top left to bottom right to the terminate end event icon.

In all cases, the respective process status also appears in webMethods Monitor.

➤ To configure the process status upon termination of a terminate end event

1. In an open process, click the terminate end event you want to work with to select it.
2. In the Properties view, click the **Implementation** page.
3. Select one of the following **Process status upon termination** values:
 - Completed (default)
 - Canceled
 - Failed
4. Save the process.

Basic Event Properties

Note:

Some events do not provide all of the property pages listed below, depending on the event behavior. For example, a terminate end event displays only the **General** and **Documentation** pages.

The following table describes the Basic Event properties.

Properties Page	Property	Description
General	Type	The event type. Make a selection in this drop-down to change the event type. Choices are limited to the secondary types of the current primary event type. To change the primary event type, use the speed buttons as described in “Working with Speed Buttons” on page 80.
	Throw Event	<i>Intermediate message and signal events (NOT including boundary events) only.</i> Select this option to implement throwing (sending) behavior.
	Catch Event	<i>Intermediate message and signal events (NOT including boundary events) only.</i> Select this option to indicate catching (receiving) behavior. Selected by default.
	Allow [event type] to interrupt step	<i>Intermediate boundary timer, message, and signal events only.</i> Select the check box to enable interruption. Clear the check box to disable interruption.
	Label	Step name. Default is [EventType]1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name , Font Size , Bold , Italic , and Font Color . Default settings are Tahoma 8 point black, with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
	Documentation	Documentation Fields

Properties Page	Property	Description
		Documentation fields are searchable. For more information, see “About Process Documentation” on page 380.
	Documentation Field Value	The text you type that will appear within the Documentation Field .
KPIs	KPIs for <Step Name>	Step-level KPI (Key Performance Indicator) definitions. For more information about the fields on this page and how to work with them, see “About KPIs” on page 372.
	KPI Properties	Information used to describe a KPI: Name, Description, Unit of Measure, Associated Field, Aggregation Type , and Dimensions . See “About Step-Level KPIs” on page 375.
	Name	Name of the KPI. Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names on the target server. However, if you duplicate a KPI name in a process, generation of the process produces a warning.
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.
	Associated Field	Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used. Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.
	Aggregation Type	Aggregation type can be set to SUM , AVERAGE , or LAST VALUE

Properties Page	Property	Description
	Dimensions	<p>Use this area to add, delete, and manage data dimensions for a selected KPI. You can associate a field from the step output with one or more dimensions.</p> <p>Note: Dimension names must be unique on a given Optimize server. Designer cannot, at design time, detect all the dimension names on the target server. However, if you duplicate a dimension label in a process, generation of the process produces a warning.</p> <p>For more information, see “About KPIs” on page 372.</p>
Inputs / Outputs	Inputs	Data flowing into the step. You can add and remove inputs manually, as well as add or update from the service signature. For more information, see “About Inputs and Outputs” on page 90 .
	Outputs	Data flowing out of the step. You can add and remove outputs manually, as well as add or update from the service signature.
	Edit Data Mapping (link)	<p>Opens the Data Mapping editor for the input and output of the step. An Integration Server connection is required.</p> <p>For more information about the fields on this page and how to work with them, see “About Inputs and Outputs” on page 90.</p>
Transitions	Transition Type	<p>Defines the transition behavior. Options are If Condition, Default, Join Timeout, Iterations Exceeded, and Unsatisfied Join.</p> <p>Note: Some steps do not support certain transition types. A warning message appears if you select a transition type that is not supported by the step.</p> <p>For more information about these transition types and their configuration fields, see:</p>
Available only when the event has a transition line drawn to one or more steps.	For information about transitions in general, see “About Transitions” on page 117 .	

Properties Page	Property	Description
		<ul style="list-style-type: none"> ■ “Configuring Transition Behavior” on page 125. ■ “About Transition Type Behavior” on page 118. ■ “About Transition Types” on page 118. ■ “How Incomplete Transitions Affect Join Steps and Gateways” on page 132.
Timer Condition	This page is available only for boundary intermediate timer events.	<p>The timer duration value to apply to the intermediate timer event. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>For more information, see “About Boundary Timer Event Timer Conditions” on page 143.</p>

14 EDA (Deprecated) Event Types

■ About EDA Event Types	316
■ About EDA Event Type Storage	317
■ Adding a Custom EDA Event Type to a Message Step	318
■ Adding an EDA Event Type to an Activity Step	320
■ Enabling and Disabling Predefined EDA Event Emission for a Process Model	321
■ About EDA Event Types and Integration Server Document Types	322
■ About EDA Event Types and Logged Fields	323
■ Importing the EDA Predefined Event Type Project	324
■ About EDA Predefined Process Event Types	325

About EDA Event Types

The webMethods product suite provides an Event Driven Architecture (EDA) that enables suite applications to both emit and consume EDA events that occur within the suite run time. An EDA *event type* is a description of an event, expressed in XML format. The schema of an EDA event type is represented in the form of an XSD file.

The Process Engine is capable of emitting the following EDA events:

- Process-specific predefined EDA event types. You must manually enable each individual process model to emit these events, as described in [“Enabling and Disabling Predefined EDA Event Emission for a Process Model” on page 321](#).
- Custom EDA event types that you develop using the Software AG Designer EDA event type editor in the Events Development perspective. These events are emitted automatically after being added to a process model.

Note:

EDA Event Types are converted to IS Doc Types in Integration Server. The Process Engine emits an instance of this IS Doc Type at run time. As with IS Doc Types, EDA Event Types cannot be recursive.

If you want to view a predefined EDA event type, or use it as a template for creating your own EDA event type, you can import all predefined event types into Software AG Designer, as described in [“Importing the EDA Predefined Event Type Project” on page 324](#).

For more information about working with EDA event types in Software AG Designer, see the *webMethods Event Processing Help*. For more information about how EDA integrates into the webMethods product suite, how to deploy EDA assets, how to create event-enabled applications, and other topics, see the PDF publication *Implementing Event-Driven Architecture with webMethods Products*.

Working with EDA Event Types in BPMN Message Events

Software AG Designer extends BPMN functionality to support the use of EDA event types in the following BPMN message events:

- Message start events
- Catching and throwing message intermediate events
- Catching and throwing boundary message intermediate events
- Message end events

To catch or throw a message that contains an EDA event type, you must use the **EDA (For Event Triggered Processes)** protocol.

- In a catching message event, this is designated as the **Receive Protocol**.
- In a throwing message event, this is designated as the **Send Protocol**.

In the Events Development perspective, you can:

- Drag an EDA event type from the Project Explorer view onto the process canvas to create a message start event.
- Drag an EDA event type onto an existing message start event to reconfigure the step to use the new EDA event type.

Tip:

When you are configuring BPMN message event step properties on the Implementation page in the Properties view, you can also access EDA event types in the local event type repository using the browse button.

Note:

When you drag an EDA event type from the Project Explorer view onto a process model in the process editor, the Integration Server Name for the step is always set to Default. This is because no Integration Server connection is required to work with EDA event types. If you want to change the Integration Server Name, you must do it manually when connected to the Integration Server.

For more information about the local event type repository and the Event Type Store, see [“About EDA Event Type Storage” on page 317](#).

About EDA Event Type Storage

Event types are stored in and available from two separate locations:

- The Event Type Store is a shared run-time repository of predefined (by Software AG) event types and all deployed user-defined (custom) EDA event types. This shared location is used by all EDA participants (including the Process Engine) at run time to retrieve deployed custom EDA event types. Custom event types that are created during design time in Designer are added to the Event Type Store automatically upon deployment of the process model that contains them.
- A local EDA event type repository is created in your Designer workspace when you create an event type with the Events Development perspective, or when you import predefined event types from the Event Type Store. This repository is visible in the Project Explorer view, where you can browse through event type projects. The repository is also available with the **Browse** button for the **Send** or **Receive Document** field on the Implementation page in the Properties view.

The Project Explorer view is included by default in the Events Development perspective, and you can add it to the Process Development perspective, or any other perspective. You can also use standard Eclipse functionality to open the Project Explorer as a detached view.

For more information about deploying EDA assets to the Event Type Store, see the PDF publication *Implementing Event-Driven Architecture with webMethods Products*.

Adding a Custom EDA Event Type to a Message Step

You can use a custom EDA event type in start message, intermediate message, or end message steps in a process. At run time, the message step will automatically emit an EDA event based on the Event Type schema XSD file. You need only to specify an EDA event type.

Note:

The Process Engine also generates and emits predefined events for the process model instance and for process model instance steps. For more information, see [“About EDA Predefined Process Event Types”](#) on page 325.

Important:

To fully complete the following procedure, you must have an active connection to an Integration Server when you add an EDA event type to a process. If you do not, you are prompted to open a connection, or to work offline, when you add the event type. For more information, see [“About EDA Event Types and Integration Server Document Types”](#) on page 322.

➤ To add a custom EDA event type to a message step

1. To add an EDA event type to a message step, do one of the following:

- To create a new start message step that uses an EDA event type, drag an EDA event type from the Project Explorer view onto the process canvas.
- To add an EDA event to an existing start message, intermediate message, or end message step, drag an EDA event from the Project Explorer view onto the existing step.

Integration Server document type for the event type will be created. If you have only one default Integration Server

In addition, the **Receive Document** and **Receive EDA Event** fields are automatically configured for you. The **Protocol Properties** fields are partially configured, but you must manually specify a **Destination Name**.

Note:

You can also add an EDA event type by dragging an Integration Server document type that represents an event type from the Package Navigator view.

- To manually create a start, intermediate, or end message step:
 1. Add the step to the canvas from the Palette and click it to select it.
 2. On the **Implementation** page in the Properties view, select the **EDA (For Event Triggered Processes) Receive Protocol** or **Send Protocol**.
 3. Click the **Browse** button next to the **Receive Document** or **Send Document** field.
 4. Click the **Search EDA Event Type Store** tab and select the event type you want to work with. You can also click the **Browse Integration Server** tab and select an IS document type that represents an EDA event type.

5. Verify the **Connection Alias** in use and specify a **Destination Name**.
2. Provide any additional configuration to the message step as described in “[Configuring a Start Event](#)” on page 267, “[Configuring an Intermediate Event](#)” on page 275, and “[Configuring an End Event](#)” on page 298.
3. Save the process.

Publishing an EDA Event from a Message Step

This procedure applies to BPMN throwing message intermediate events and message end events.

➤ To configure a message step to publish an EDA event

1. Select the message step you want to work with on the process canvas.
2. On the **Implementation** page in the Properties view, set the **Send Protocol** to **EDA (For Event Triggered Processes)**.
3. Click the **Browse** button next to the **Send Document** field.
4. In the Choose Document dialog box, click the **Search EDA Event Type Store** tab, select the event type you want to use, and then click **OK**. You can also click the **Browse Integration Server** tab and select an IS document type that represents an EDA event type.
5. Verify the **Connection Alias** in use and specify a **Destination Name**.
6. Save the process.

Listening for an EDA Event in Message Step

This procedure applies to BPMN catching message intermediate events and message start events.

➤ To configure a message step to listen for an EDA event

1. Select the message step you want to work with on the process canvas.
2. On the **Implementation** page in the Properties view, set the **Receive Protocol** to **EDA (For Event Triggered Processes)**.
3. Click the **Browse** button next to the **Receive Document** field.
4. In the Choose Document dialog box, click the **Search EDA Event Type Store** tab, select the event type you want to use, and then click **OK**. You can also click the **Browse Integration Server** tab and select an IS document type that represents an EDA event type.

5. Verify the **Connection Alias** in use and specify a **Destination Name**.
6. Save the process.

Adding an EDA Event Type to an Activity Step

You can add a custom EDA event type to any activity step that supports a BPMN boundary message intermediate event. For a table of which activity steps are supported, see [“About Interrupting Behavior for Boundary Intermediate Events” on page 288](#). At run time, the boundary message intermediate event will automatically emit an EDA event based on the Event Type schema XSD file. You need only to specify an EDA event type.

Note:

The Process Engine also generates and emits predefined EDA events for the process model instance and for process model instance steps. For more information, see [“About EDA Predefined Process Event Types” on page 325](#).

Important:

To fully complete the following procedure, you must have an active connection to an Integration Server when you add an EDA event type to a process. If you do not, you are prompted to open a connection, or to work offline, when you add the event type. For more information, see [“About EDA Event Types and Integration Server Document Types” on page 322](#).

> To add a custom EDA event type to an activity step as a BPMN boundary message intermediate event

1. Do one of the following:
 - Drag an EDA event type from the Project Explorer view onto the activity step. A boundary message intermediate event is created on the activity automatically.
 - Drag an EDA event from the Project Explorer view onto an existing boundary message intermediate event on an activity.

In both of the above cases, the **Receive Document** and **Receive EDA Event** fields are automatically configured for you. The **Protocol Properties** fields are partially configured, but you must manually specify a **Destination Name**.

Note:

You can also add an EDA event type by dragging an Integration Server document type that represents an EDA event type from the Package Navigator view.

- To manually configure an existing boundary message intermediate event to use the selected EDA event type:
 1. Click the boundary message intermediate event you want to work with to select it.
 2. On the Implementation page of the Properties view, select the **EDA (For Event Triggered Processes) Receive Protocol**.

3. Click the **Browse** button next to the **Receive Document** field.
 4. Click the **Search EDA Event Type Store** tab and select the EDA event type you want to work with. You can also click the **Browse Integration Server** tab and select an IS document type that represents an EDA event type.
 5. Verify the **Connection Alias** in use and specify a **Destination Name**.
2. Provide any additional configuration to the boundary message intermediate event as described in [“Configuring a Boundary Intermediate Event” on page 290](#).
 3. Save the process.

Enabling and Disabling Predefined EDA Event Emission for a Process Model

You can enable and disable the emission of predefined Process Engine EDA events for individual process models on an event-by-event basis. When event type emission is disabled, no predefined events are emitted by the model for the disabled event type.

EDA event emission for all predefined event types is disabled by default. To enable event emission, you must manually enable the predefined EDA events you want to emit for each individual process model. The event emission enablement settings are maintained with the process through the build and upload procedure. They are part of the process asset description, and are sent with the process when deployed with webMethods Deployer.

Note:

Disabling Process Engine EDA events has no effect on custom event types applied to the process model or steps within it.

You can also enable or disable event emission in the run time with webMethods Monitor, on a process model's Edit Process page. For more information, see the PDF publication *webMethods Monitor User's Guide*.

Important:

Before you make modifications to EDA event emission settings, be aware of the interaction of these settings between Designer and webMethods Monitor. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor” on page 66](#).

➤ To enable or disable EDA event emission for a process model

1. Click the process editor's canvas to select a process.
2. In the Properties view, click the **Run Time** tab.
3. On the **Run Time** page, in the **Runtime Editable Properties** section, do the following in the **Emit Process-specific Predefined EDA Events** area:

- Click **Synchronize** to retrieve the current webMethods Monitor settings from the Process Audit database and apply them to EDA events and step enablement areas. A confirmation dialog box appears to inform you either that the settings are identical and no changes are needed, or that the settings are different, in which case you can choose to synchronize the settings or cancel.
- Click **Select All/Clear All** to select or clear all EDA events.
- Select a predefined event type check box to enable emission of the predefined event for this model.
- Clear a predefined event type check box to disable emission of the predefined event for this model.

4. Save the process.

You must build and upload the process model to apply the new settings to the runtime environment. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor”](#) on page 66.

About EDA Event Types and Integration Server Document Types

In general, you can work directly with EDA event types available from the Package Explorer view when you are designing a business process. However, you can also add an EDA event type to a process from the Package Navigator view as an Integration Server (IS) document type that represents the EDA event type.

Each time you add an EDA event type to a process model in the process editor, Designer determines if there is a corresponding EDA event type in a package on the connected Integration Server. The package represents the EDA event type project and it contains an IS document type that represents the event type.

If no corresponding EDA event type package and EDA event type document type exists on the Integration Server, Designer creates one. If an EDA event type project contains more than one event type, only the event type you are working with is added to the Integration Server package as an IS document type. You might have to refresh the Package Navigator view to see the newly added package or document type.

The result is that after you add an EDA event type to a process, an IS document type is available for use in your process models. You can create an EDA event type-enabled step with an IS document using the same procedures described in the following topics, except that you drag or select the IS document type instead of the actual EDA event type:

- [“Adding a Custom EDA Event Type to a Message Step”](#) on page 318
- [“Publishing an EDA Event from a Message Step”](#) on page 319
- [“Listening for an EDA Event in Message Step”](#) on page 319

Important:

If you choose to work offline without an active Integration Server connection, you can add the EDA event type to the process, but when you attempt to build and upload the process, the build

and upload may fail with an error that the EDA event type package does not exist on the Integration Server. This will occur if the EDA event type project you are working with has not been previously created as a package on the Integration Server.

If this occurs, the simplest way to resolve the problem is to delete the EDA event type step that is causing the error from the process, establish a connection to Integration Server, and then add the EDA event type step to the process again. This will create the required package for the EDA event type on the Integration Server.

About EDA Event Types and Logged Fields

By default, a process instance step emits a *process step instance change event* each time the step changes status, as described in [“About EDA Predefined Process Event Types” on page 325](#). If you want the emitted EDA event to contain business data from within the step, you must designate logged fields for the step:

- All designated logged fields are automatically included in the payload of the emitted event. For more information, see [“How Logged Fields are Packaged in EDA Process Events” on page 323](#).
- If the step has no logged fields, no business data is included.
- Input logged fields are emitted in the process step instance change event for Started status.
- Output logged fields are emitted in the process step instance change event for Completed status.

For information about the location and contents of the XSD schema that contains logged fields as business data, see [“About Process Event Schemas” on page 327](#).

For information about how to designate logged fields, see [“Log Inputs and Outputs” on page 93](#).

How Logged Fields are Packaged in EDA Process Events

When you build and upload a process model that contain documents with logged fields, those fields marked for logging are logged to the audit database as described in [“Log Inputs and Outputs” on page 93](#). In addition, the build and upload process generates an event type for each of these fields, and, if EDA event emission is enabled for the model, values for these fields are included as part of the payload of the EDA process events that are emitted at run time.

These fields are processed as follows:

- If the logged field is part of an IS document that originated as an EDA event type (for example, the event type was dragged into the process), then the process stores a reference to the original event type.
- If the logged field is part of an IS document that was created somewhere in the process pipeline, the build and upload process creates an EDA event type in the local Designer event store corresponding to that entire document (that is, not just the selected fields).

For example:

```

-source-
type           : ISDocType
name           : <ISProject>/<ISfolder>/<ISDocType>

-target-
namespace     : http://namespaces.softwareag.com/EDA
path          : <ISProject>/Event Types/<ISfolder>/<ISDocTypeName>.xsd

--Example--
As IS Doc Type      : MyProject/OrderMgmt/OrderShipment
As EDA Asset        : MyProject/Event Types/OrderMgmt/OrderShipment.xsd
As EDA Event Type ID : {http://namespaces.softwareag.com/EDA
                        /OrderMgmt}OrderShipment

```

- If the logged field represents the scalar output of a step or underlying service with no document type, then the build and upload process creates a new event type for each step following this naming convention: ProcessA.StepB.Direction.ScalarData fields (for example field1, field2, field3).

For example:

```

-source-
type           : logged field names is a process step - Input and Output

-target-
namespace     : http://namespaces.softwareag.com/EDA/Process
path          : <ISProject>/Event Types/Process/<processName>
                </stepName>/<direction>/ScalarData.xsd

--Example--
As Asset       : MyProject/Event Types/Process/Finance/OrderMgmt
                /OrderProcessing/S33/Input/ScalarData.xsd
As Event Type ID : {http://namespaces.softwareag.com/EDA/Process/Finance
                    /OrderMgmt/OrderProcessing/CheckSupplierInventory
                    /Input}ScalarData

```

At run time, the Process Engine:

- Creates the appropriate EDA event of the associated event type.
- Populates the logged fields into this corresponding event.
- Wraps this event in the ProcessStepInstanceChange event and emits it.

Importing the EDA Predefined Event Type Project

Your Software AG installation contains all the predefined event types that have been defined for Software AG products, as well as some sample EDA event types. You can import these event types into Designer as an EDA event type project in your local event type repository.

After you import the project, it appears in the Package Explorer view, available by default on the Designer Events Development perspective. You can open these EDA event types in the event editor to see how they are constructed, and you can use them as templates for other events.

➤ **To import the EDA predefined event type package into your local event type repository**

1. In Designer: **File > Import**.
2. In the Import dialog box, click **General** and then **Existing Projects into Workspace**. Then click **Next**.
3. Click **Browse** and navigate to this directory in your Software AG installation
Software AG_directory \common\PredefinedEventTypes.
4. Click **OK** to accept the directory and place it in the **Select root directory** field. If the PredefinedEventTypes project is not already selected in the **Projects** list, select it.
5. Click **Finish**.

After you import the PredefinedEventTypes project, you can access the events in it from the Project Explorer view, available in the Events Development perspective.

About EDA Predefined Process Event Types

As part of its integration with Event Driven Architecture (EDA), Process Engine emits specific predefined events for each process model instance and for process model instance steps. The ability to emit these events can be enabled and disabled on a model-by-model basis. For more information, see [“Enabling and Disabling Predefined EDA Event Emission for a Process Model” on page 321](#).

Each EDA event represents a change to the status of the process instance or process step, and it also contains pertinent information such as the process ID, step ID, time stamp, and so on. The content of each EDA event is defined by the event type XSD schema files installed with Process Engine. For specific information, see [“EDA Process Event Element Definitions” on page 327](#).

Note:

Process instance change and process step instance change events are used to trigger process stage event emission by the Process Tracker in webMethods Optimize. For more information about process stages, see [“About Process Stages and Milestones” on page 352](#).

The following table lists the available EDA predefined process event types:

Process Event Type	Emitted when
Process instance change	The status of the process instance changes. See “Process Instance Change” on page 328 for more information.
Process instance log message	The service pub.prt.log:logActivityMessages is invoked in a process model step. See “Process Instance Log Message” on page 330 for more information.
Process instance log custom ID	The service pub.prt.log:logCustomId is invoked in a process model step. See “Process Instance Log Custom ID” on page 329 for more information.

Process Event Type	Emitted when
Process instance error	A process error occurs in a process instance. See “Process Instance Error” on page 329 for more information.
Process step instance change	The status of a step instance changes. See “Process Step Instance Change” on page 330 for more information.
Process step loop instance change	A step loop starts or completes. This applies to steps that are configured for BPMN standard looping. See “Process Step Loop Instance Change” on page 332 .
Process step instance transition	A transition from one step to another occurs. See “Process Step Instance Transition” on page 332 for more information.
Process step instance error	A step error occurs. See “Process Step Instance Error” on page 331 for more information.

For information about process instance and step statuses, see Chapter 5, *Process Monitoring*, in the PDF publication *webMethods Monitor User’s Guide*.

In addition to the above EDA events, Process Engine also emits the stage events listed in the following table:

Process Event Type	Emitted when
Process stage breached	A process stage fails to complete within the defined stage condition. Schema: ProcessStageBreached.xsd
Process stage completed	A process stage reaches completion within the terms of the defined stage condition. Schema: ProcessStageCompleted.xsd
Process stage definitions change	One or more values that define a process stage change. Schema: ProcessStageDefinitionsChange.xsd
Process stage instance	Supporting schema. Contains the elements of a process stage instance. Schema: ProcessStageInstance.xsd
Process stage model	Supporting schema. Contains the elements of a process stage model. Schema: ProcessStageModel.xsd

Process Event Type	Emitted when
Process stage started	A process stage begins executing. Schema: ProcessStageStarted.xsd

These events are primarily internal in nature, and are used to deliver changes in the stage life cycle to the webMethods Process Tracker. Should you be interested in learning more about their contents, refer to the schema files mentioned above in this location: *Software AG_directory* \common\EventTypeStore\WebM\Process\2.0.

Note:

These stage-related EDA events are always emitted by the Process Tracker and cannot be disabled.

About Process Event Schemas

The Process Engine installation contains an XSD schema file for each of the predefined process events.

- To view these XSD files, you must first import the predefined event types as described in [“Importing the EDA Predefined Event Type Project” on page 324](#).
- You can browse to the schemas in the Designer Project Explorer view by expanding these nodes: **PredefinedEventTypes > Event Types > WebM > Process > 2.0**.
- You can also access the schema files in your file system at *Software AG_directory* \common\EventTypeStore\WebM\Process\2.0.
- For detailed information about the elements found in each of the schema files, see [“EDA Process Event Element Definitions” on page 327](#).

Note:

Event type schemas are versioned, and future developments may result in later version numbers. Be sure to check the directory location mentioned above for the latest available version. In general, only the latest version is emitted.

EDA Process Event Element Definitions

You can find detailed information on the elements that make up the various process events in these topics:

- [“EDA Predefined Process Event Types” on page 327](#)
- [“Key Supporting Schemas” on page 333](#)

EDA Predefined Process Event Types

These process event schemas can be found in the folder *Software AG_directory* \common\EventTypeStore\WebM\Process\2.0

The following table lists the available predefined process event types:

Event Type	Description
Process Instance Change	Emitted when changes are applied to a process instance and defines the resulting status.
Process Instance Error	Emitted when a process instance error condition occurs.
Process Instance Log Custom ID	Emitted when the service pub.prt.log:logCustomId is invoked in a process model step.
Process Instance Log Message	Emitted when the service pub.prt.log:logActivityMessages is invoked in a process model step.
Process Step Instance Change	Emitted when a change occurs to the status of a step instance.
Process Step Instance Error	Emitted when a step error occurs in a process instance.
Process Step Instance Transition	Emitted when a transition from one step to another occurs.
Process Step Loop Instance Change	Emitted when a step loop starts or completes. This applies to steps that are configured for BPMN standard looping.

Process Instance Change

Emitted when changes are applied to a process instance and defines the resulting status.

Schema file: ProcessInstanceChange.xsd

The following table describes the process instance change elements.

Element	Description
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID. For more information, see “Process Instance” on page 335 .
Status	One of the following values: <ul style="list-style-type: none"> ■ Completed ■ Expired

Element	Description
	<ul style="list-style-type: none"> ■ Failed ■ Failed (Escalated) ■ Resumed ■ Revised ■ Started ■ Stopped ■ Suspended ■ Interrupted

Process Instance Error

Emitted when a process instance error condition occurs.

Schema file: ProcessInstanceError.xsd

The following table describes the process instance error elements.

Element	Description
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the error occurred. For more information, see “Process Instance” on page 335 .
ProcessError	Information about the error that occurred. For more information, see “Process Error” on page 335 .

Process Instance Log Custom ID

Emitted when the service `pub.prt.log:logCustomId` is invoked in a process model step.

Schema file: ProcessInstanceLogCustomId.xsd

The following table defines the process instance log custom ID elements.

Element	Definition
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .

Element	Definition
ProcessInstance	The process instance ID where the service <code>pub.prt.log:logCustomID</code> was invoked. For more information, see “Process Instance” on page 335 .
ProcessCustomId	The custom ID (also known as a “friendly name”) created for the process by the public service <code>pub.prt.log:logCustomID</code> . For more information, see “Process Custom ID” on page 334 .

Process Instance Log Message

Emitted when the service `pub.prt.log:logActivityMessages` is invoked in a process model step.

Schema file: `ProcessInstanceLogMessage.xsd`

The following table defines the process instance log message elements.

Element	Definition
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the service <code>pub.prt.log:logActivityMessages</code> was invoked. For more information, see “Process Instance” on page 335 .
ProcessStepModel	Information about the process model step from which the log message originated. For more information, see “Process Step Model” on page 337 .
ProcessStep Instance	Information about the process step instance from which the log message originated. For more information, see “Process Step Instance” on page 337 .
ProcessActivity	Information about the log message. For more information, see “Process Activity” on page 334 .

Process Step Instance Change

Emitted when a change occurs to the status of a step instance.

Schema file: `ProcessStepInstanceChange.xsd`

The following table describes the process step instance change elements.

Element	Description
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the change occurred. For more information, see “Process Instance” on page 335 .

Element	Description
ProcessStepModel	Information about the process model step where the change occurred. For more information, see “Process Step Model” on page 337 .
ProcessStepInstance	Information about the process step instance where the change occurred. For more information, see “Process Step Instance” on page 337 .
BusinessData	<p>Any logged fields specified for the step are included in the BusinessData element.</p> <ul style="list-style-type: none"> ■ Fields that are logged on input are included when the event status is "Started". ■ Fields that are logged on output are included when the event status is "Completed". <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The document list data selected on the Logged Fields tab of the step will not appear in the BusinessData section of the event. The document list data will appear in the audit data displayed by Monitor.</p> </div>
Status	<p>One of the following values:</p> <ul style="list-style-type: none"> ■ Completed ■ Expired ■ Failed ■ Retries Exceeded ■ Unsatisfied join ■ Started ■ Stopped ■ Waiting ■ Interrupted

Process Step Instance Error

Emitted when a step error occurs in a process instance.

Schema file: ProcessStepInstanceError.xsd

The following table defines the process step instance error elements.

Element	Definition
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the error occurred. For more information, see “Process Instance” on page 335 .
ProcessStepModel	Information about the process model step where the error occurred. For more information, see “Process Step Model” on page 337 .
ProcessStepInstance	Information about the process step instance where the error occurred. For more information, see “Process Step Instance” on page 337 .
ProcessError	Information about the error that occurred. For more information, see “Process Error” on page 335 .

Process Step Instance Transition

Emitted when a transition from one step to another occurs.

Schema file: ProcessStepInstanceTransition.xsd

The following table defines the process step instance transition elements.

Element	Definition
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the transition occurred. For more information, see “Process Instance” on page 335 .
ProcessSource StepModel	Information about the process model step where the transition starts. For more information, see “Process Step Model” on page 337 .
ProcessSource StepInstance	Information about the process step instance where the transition starts. For more information, see “Process Step Instance” on page 337 .
ProcessTarget StepModel	Information about the process model step where the transition ends. For more information, see “Process Step Model” on page 337 .

Process Step Loop Instance Change

Emitted when a step loop starts or completes. This applies to steps that are configured for BPMN standard looping.

Schema file: ProcessStepLoopInstanceChange.xsd

The following table defines the process step loop instance change elements.

Element	Definition
ProcessModel	The process model ID of the model that the referenced instance was started from. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance ID where the change occurred. For more information, see “Process Instance” on page 335 .
ProcessStep Model	Information about the process model step where the change occurred. For more information, see “Process Step Model” on page 337 .
ProcessStep Instance	Information about the process step instance where the change occurred. For more information, see “Process Step Instance” on page 337 .
ProcessStepLoop Instance	Information about the process step loop instance. For more information, see “Process Step Loop Instance” on page 337 .
BusinessData	Any logged fields specified for the step are included in the BusinessData element. <ul style="list-style-type: none"> ■ Fields that are logged on input are included when the event status is "Started". ■ Fields that are logged on output are included when the event status is "Completed".
Status	One of the following values: <ul style="list-style-type: none"> ■ Started ■ Completed

Process Stage Events

These events are emitted when a process stage starts, completes, breaches its defined condition, and when its definition values are changed. The events are primarily internal in nature, and are used to deliver changes in the stage life cycle to the webMethods Process Tracker.

For more information about stage events, see [“About EDA Predefined Process Event Types” on page 325](#)

Key Supporting Schemas

The schemas listed in this section provide support for the predefined EDA process events. These, as well as other supporting event schemas, can be found in the folder *Software AG_directory \common \EventTypeStore \WebM \Process \2.0*, along with the EDA process event schemas.

The table below lists and defines the available supporting schemas for predefined process event types.

Supporting Schema	Definition
Process Activity	Provides information about specific process activity.
Process Custom ID	Provides information about a process custom ID.
Process Error	Provides information about a process error.
Process Instance	Provides information about a process instance.
Process Instance with Parent	Provides information about a process instance parent.
Process Model	Provides information about a process model.
Process Parent	Provides information about a parent process.
Process Step Model	Provides information about a process model step.
Process Step Instance	Provides information about a process instance step.
Process Step Loop Instance	Provides information about a process step loop.

Process Activity

Provides information about specific process activity.

Schema file: ProcessActivity.xsd

The following table describes the process activity elements.

Element	Description
BriefMessage	String. Optional. A short text description of the process activity. Specified in the <i>BriefMessage</i> parameter of the public service <code>pub.prt.log:logActivityMessages</code> .
FullMessage	String. Optional. A complete description of the process activity. Specified in the <i>FullMessage</i> parameter of the public service <code>pub.prt.log:logActivityMessages</code> .
Type	One of the following values, as specified in the <i>EntryType</i> parameter of the public service <code>pub.prt.log:logActivityMessages</code> : <ul style="list-style-type: none">■ Error■ Warning■ Message

Process Custom ID

Provides information about a process custom ID.

Schema file: ProcessCustomId.xsd

The following table defines the process custom ID elements.

Element	Definition
CustomId	String. Optional. The custom ID (“friendly name”) given to the process model, specified in the <i>customID</i> parameter of the public service <code>pub.prt.log:logCustomId</code> .

Process Error

Provides information about a process error.

Schema file: `ProcessError.xsd`

The following table defines the process error elements.

Element	Definition
ErrorMessage	String. Optional. The error message text.
ErrorStackTrace	String. Optional. The stack trace associated with the error.
ServiceName	String. Optional. The name of the service involved in the error.

Process Instance

Provides information about a process instance.

Schema file: `ProcessInstance.xsd`

The following table defines the process instance elements.

Element	Definition
ID	String. The process instance ID.
Iteration	Integer. Optional. The iteration number of the process instance.
DisplayName	String. Optional. The display (“friendly”) name of the process instance.

Process Instance with Parent

Provides information about a process instance parent.

Schema file: `ProcessInstanceWithParent.xsd`

The following table defines the process instance parent elements.

Element	Definition
ID	String. The process instance ID.

Element	Definition
Iteration	Integer. Optional. The iteration number of the process instance.
Parent	Optional. The parent process from which this process instance was started. For more information, see “Process Parent” on page 336 .

Process Model

Provides information about a process model.

Schema file: ProcessModel.xsd

The following table defines the process model elements.

Element	Definition
ID	String. The process model ID.
DisplayName	String. Optional. The display (“friendly”) name of the process model.
Version	String. Optional. The version of the process model.

Process Parent

Provides information about a parent process.

Schema file: ProcessParent.xsd

The following table defines the process parent elements.

Element	Definition
ProcessModel	Optional. The process model that started the child process instance. For more information, see “Process Model” on page 336 .
ProcessInstance	The process instance that started the child process instance. For more information, see “Process Instance” on page 335 .
ProcessStepModel	Optional. Information about the process model step that started the child process instance. For more information, see “Process Step Model” on page 337 .
ProcessStepInstance	Optional. Information about the process step instance that started the child process instance. For more information, see “Process Step Instance” on page 337 .
ProcessStepLoopInstance	Optional. The iteration number of the process step loop instance that started the child process instance.

Process Step Model

Provides information about a process model step.

Schema file: ProcessStepModel.xsd

The following table defines the process step model elements.

Element	Definition
ID	String. The ID of the process model step.
DisplayName	String. Optional. The display (“friendly”) name of the process model step.
StepType	String. Optional. The type of step.

Process Step Instance

Provides information about a process step instance.

Schema file: ProcessStepInstance.xsd

The following table defines the process step instance elements.

Element	Definition
Iteration	Integer. Optional. The iteration number of the process step instance.

Process Step Loop Instance

Provides information about a process instance step loop.

Schema file: ProcessStepLoopInstance.xsd

The following table defines the process step loop instance elements.

Element	Definition
Iteration	Integer. Optional. The iteration number of the process step loop instance.

15 Gateway Steps

■ About Gateways	340
■ About Gateway Types	341
■ Adding a Gateway	343
■ Configuring a Gateway	344
■ Changing the Gateway Type	344
■ Removing a Gateway	344
■ Basic Gateway Properties	345
■ Advanced Gateway Properties	347

About Gateways

Gateways split and join data paths in a process flow. No service is associated with a gateway step, and other than determining direction of flow, they have no effect on the process data going through the gateway. Use a gateway only when you need to merge or split a data flow.

A single gateway can have multiple inputs and multiple outputs flows, and can be configured to perform both a merge (or join) and a split in a single step. However, BPMN best practices recommend that you configure any single gateway to perform only one of these functions. In this case, you would create two sequential gateways, the first one to merge and the second one to split the data flow.

There are four different gateway types, as described in “[About Gateway Types](#)” on page 341. For detailed information about gateway behavior, see the BPMN 2.0 specification at <http://www.omg.org/spec/BPMN/2.0>.

Parallel and complex gateways support join timeouts, inclusive and exclusive gateways do not support join timeouts. As of version 9.7, Parallel gateway steps support timeout and unsatisfied join transitions. Exclusive gateway does not have timeout or unsatisfied join transition conditions.

Importing Gateways from Earlier Versions

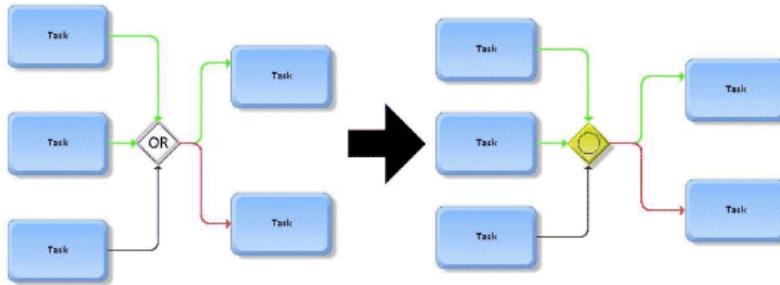
Versions 8.0 and 8.1 of Designer supported *webMethods gateways*, which do not provide all the functionality of the BPMN 2.0 gateways available in version 8.2 and later. With Designer 8.2 and later, the webMethods gateway step is no longer supported. If you are importing process models from those earlier versions:

- webMethods gateways imported into Designer 8.2 and later from 8.0 and 8.1 are converted to BPMN complex gateways, although their underlying implementation does not change. For example, join condition expressions and If Condition transition terms remain as part of the gateway after importing.
- Gateway join and dead path behavior are unchanged from the previous versions of Designer.
- The join behavior of a webMethods gateway step is configurable. With the exception of the complex gateway, BPMN gateway types in Designer 8.2 and later have fixed join behavior, depending on the gateway type.

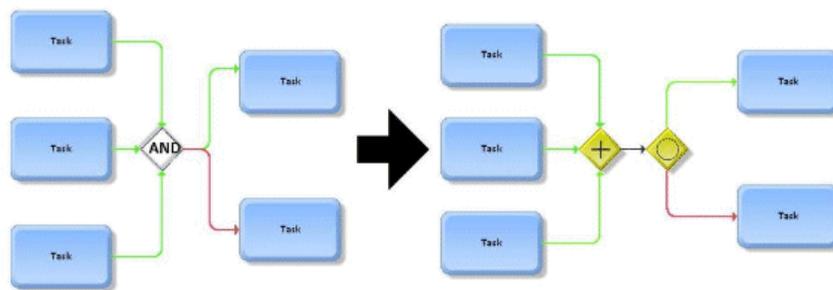
After you import any 8.0 or 8.1 process models with webMethods gateways, you are advised to restructure the resulting complex gateways to more accurately display the data flow, and to conform with BPMN practices.

For example, the following drawings indicate how some webMethods gateway scenarios could be restructured.

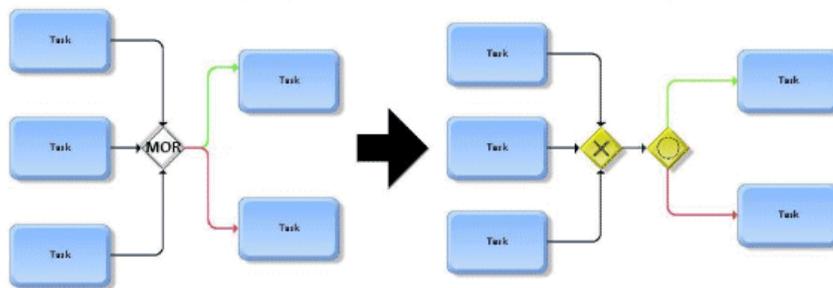
This is a webMethods gateway with an OR join and a split output, restructured to use an inclusive gateway:



This is a webMethods gateway with an AND join and a split output, restructured to use a parallel gateway and an inclusive gateway:



This is a webMethods gateway with a multiple OR (MOR) join and a split output, restructured to use an exclusive gateway followed by an inclusive gateway:



Note:

When you import a model that contains steps with internal AND, OR, MOR, or COMPLEX joins, you also have the option of restructuring them by moving the join behavior to a preceding parallel, inclusive, exclusive, or complex gateway step.

About Gateway Types

Designer represents a gateway as a yellow diamond shape. The following table lists the four different icons which represent the four gateway types:

Gateway	Behavior
 Exclusive (XOR, or Unsynchronized OR)	<p>Diverging behavior:</p> <ul style="list-style-type: none"> ■ Only one of the available output transition paths can be taken. ■ The default transition path (optional) is taken if none of the conditions on the other transition paths are true. ■ If no transition path is true and a default path is not available, a run-time exception occurs. <p>Converging behavior:</p> <ul style="list-style-type: none"> ■ Each input transition path is routed to the output transition path without synchronization (an Unsynchronized OR join).
 Inclusive (OR, or Synchronized OR)	<p>Diverging behavior:</p> <ul style="list-style-type: none"> ■ All of the available output transition paths that evaluate to true are taken. That is, parallel output transition paths are possible. ■ The default transition path (optional) is taken if none of the conditions on the other transition paths are true. ■ If no transition path is true and a default path is not available, a run-time exception occurs. <p>Converging behavior:</p> <ul style="list-style-type: none"> ■ All input transition paths are merged in a synchronized OR join.
 Complex (webMethods)	<p>Diverging behavior is based on output transition conditions. Converging behavior supports join types: AND, OR, and unsynchronized OR. A join timeout is supported.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Current webMethods gateways migrate to complex gateways visually only (no generation or model changes).</p> </div>
 Parallel (AND)	<p>Diverging behavior:</p> <ul style="list-style-type: none"> ■ All of the available output transition paths that evaluate to true are taken. That is, parallel output transition paths are possible. ■ The default transition path (optional) is taken if none of the conditions on the other transition paths are true. ■ If no transition path is true and a default path is not available, a run-time exception occurs. <p>Converging behavior:</p>

Gateway	Behavior
	<ul style="list-style-type: none"> ■ All input transition paths are merged in an AND join (waits for all inbound paths). A join timeout is supported.

Always be sure you are adding the correct gateway type to your process. Incorrect usage or configuration can create unexpected run-time results.

For example, you can configure an exclusive gateway with two separate output transition with two separate but identical conditions, expecting that each path will be taken when the conditions are matched. However, only one of the paths will be randomly selected, because single-path operation is the defined behavior of an exclusive gateway. For this scenario, the best choice is an inclusive gateway.

Adding a Gateway

> To add a gateway to a process

1. Open the process model you want to work with in the process editor and do one of the following:
 - Drag the gateway you want to add from the palette to the design canvas, or, in the process editor palette, click the gateway you want to add, then click in the design canvas where you want to place it. You must use one of these methods the first time you add a gateway to an empty canvas.
 - Hover the cursor over an existing step on the canvas to display the speed buttons. Click the gateway speed button and then click the canvas at the location where you want to add the task activity. A transition line is created automatically from the source step to the gateway.

Tip:
If the gateway is not the one you want, right-click the gateway and click **Change Gateway Type** to specify a different gateway.

Tip:
You can set the label of the gateway to be empty, or to have the same name as another step in the process. You can also change the placement of the label. For more information, see [“About Step Labels” on page 72](#).
2. Add transition lines if needed and configure the gateway as necessary in the Properties view.
3. Save the process.

Configuring a Gateway

Gateways are configured on the pages of the Properties view. For more information about gateways, see “[About Gateways](#)” on page 340.

Note:

You must be working in Process Developer mode to view and work with advanced properties. To learn more, see “[About Process Development Modes](#)” on page 52.

➤ To configure a gateway

1. In an open process, click the gateway you want to configure to select it.
2. In the Properties view, do one or both of the following:
 - Configure the basic gateway properties described in “[Basic Gateway Properties](#)” on page 345.
 - Configure the advanced gateway properties described in “[Advanced Gateway Properties](#)” on page 347.
3. Save the process.

Changing the Gateway Type

While working with gateways in a process in Designer, you can change the gateway type of any gateway on the canvas. For example, if you created a gateway step as a parallel gateway but later determine that it should be a complex gateway, you can easily do so.

Existing process documentation is unaffected when you change the gateway type.

➤ To change a gateway to a different type

1. In an open process, right-click the gateway you want to change, then click **Change Gateway type**.
2. From the list, click the gateway type you want to change to.

Note:

You can also change the gateway type on the step's **Implementation** page in the Properties view, in the **Type** list.

3. Save the process.

Removing a Gateway

You can remove a gateway from the process editor's canvas.

➤ To remove a gateway from a process

1. In an open process, do one of the following:
 - Right-click the gateway you want to remove, then click **Delete**.
 - Click the gateway to select it and press the Delete key on your keyboard.
2. Save the process.

Basic Gateway Properties

The following table describes the basic gateway properties.

Properties Page	Property	Description
General	Type	The gateway type. You can change the gateway type by clicking another value in this list.
	Label	Step name. Default is Gateway1 .
	Step ID	Automatically generated identifier, unique within the process. Not editable.
	Font Style	Format of the Label text in the process editor and in the HTML Documentation Report. You can select the Font Name , Font Size , Bold , Italic , and Font Color . Default settings are Tahoma 8 point black, with no bold or italic.
	Description	Descriptive information about the step, for documentation purposes only. Text in step descriptions is searchable.
	Reset Default Size (button)	If the step has been manually resized, click this button to return the step to its default size.
Documentation	Documentation Fields	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned Documentation Field .
KPIs	KPIs for <Step Name>	Step-level KPIs.
	KPI Properties	Information used to describe a KPI: Name , Description , Unit of Measure , Associated Field , Aggregation Type , and Dimensions . See “About Step-Level KPIs” on page 375 .

Properties Page	Property	Description
	Name	<p>Name of the KPI</p> <p>Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names and dimension labels on the target server. However, if you duplicate a KPI name or dimension label in a process, generation of the process produces a warning.</p>
	Description	Description of the KPI.
	Unit of Measure	How the KPI is measured.
	Associated Field	<p>Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used.</p> <p>Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.</p>
	Aggregation Type	Aggregation type can be set to SUM , AVERAGE , or LAST VALUE .
	Dimensions	<p>Name of the KPI.</p> <p>Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names and dimension labels on the target server. However, if you duplicate a KPI name or dimension label in a process, generation of the process produces a warning.</p>
Inputs / Outputs	Inputs	Data flowing into the step. You can add and remove inputs manually. For more information, see “Create Inputs and Outputs” on page 91 and “Remove Inputs and Outputs” on page 92 .

Properties Page	Property	Description
	Outputs	Data flowing out of the step. You can add and remove outputs manually.
Transitions (for information about transitions in general, see “About Transitions” on page 117)	Transition Type	<p>Defines the transition behavior. Options are If Condition, Default, Join Timeout, Iterations Exceeded, and Unsatisfied Join.</p> <p>For more information about these transition types and their configuration fields, see:</p> <ul style="list-style-type: none"> ■ “Configuring Transition Behavior” on page 125. ■ “About Transition Type Behavior” on page 118. ■ “About Transition Types” on page 118. ■ “How Incomplete Transitions Affect Join Steps and Gateways” on page 132.

Advanced Gateway Properties

The following table describes the advanced gateway properties.

Properties Page	Property	Description
Implementation	Integration Server Name	Assigned Integration Server Name
	Allow Parallel Execution	Lock the step at run-time to allow it to be executed by multiple threads. For more information, see “Parallel Execution (Step Locking)” on page 117.
Logged Fields	Inputs	The input fields you want to log. Inputs are logged before the step executes. For more information, see “Log Inputs and Outputs” on page 93.
	Outputs	The output fields you want to log. Outputs are logged after the step executes.
Joins	Join Type	<p>Defines the logic the process follows when a step has two or more incoming transitions.</p> <p>This field is read-only for the following gateway types:</p> <ul style="list-style-type: none"> ■ An inclusive gateway (OR join).

Note:
The **Joins** page is displayed only when a step has

Properties Page	Property	Description
more than one incoming transition.	Join Timeout	<ul style="list-style-type: none"> ■ An exclusive gateway (unsynchronized OR join). ■ A parallel gateway (AND join). <p>A complex gateway can be configured to use an AND, OR, Complex, or Unsynchronized OR join.</p> <p>The timeout value that defines how long the gateway is to wait for the join to complete. Available for parallel and complex joins only. The source of this value can be:</p> <ul style="list-style-type: none"> ■ A static value that you define. ■ A value from a field in the process pipeline. The field value is interpreted in milliseconds. ■ A value based on a business calendar in My webMethods Server. You can specify a static number of days, hours, and minutes, or specify a pipeline field to set the day, hours, or minutes value. <p>This option is available for all join types for complex gateways, and AND joins for parallel gateways.</p> <p>For more information, see “About Join Timeouts” on page 141</p>
	Join Condition	<p>This option is available only for a complex join in a complex gateway.</p> <p>When the terms of this condition are met (that is, the condition is true), the join is considered satisfied.</p> <p>For more information about using the condition editor, see “About Complex Join Expressions” on page 137 and “Defining a Complex Join Expression” on page 138</p>
	Deprecated properties (not recommended)	<p>These deprecated properties are available only for OR, AND, and COMPLEX joins for complex gateways, for OR joins in inclusive</p>

Properties Page	Property	Description
		gateways, and for AND joins in parallel gateways.
		Suppress Join Failure
		This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.
		Ignore dead path notification
		This option is not recommended except in very isolated cases. For more information, see “Migrating Process Models with Join Steps to Version 9.7 and Later” on page 134.

16 Stages and Milestones

■ About Process Stages and Milestones	352
■ About Milestones	352
■ Working with Stages	353

About Process Stages and Milestones

Software AG Designer enables you to monitor the behavior of your process instances by defining *stages* within your process model. A stage spans one or more steps within a process model, and you can define as many stages as you need to monitor your process behavior.

Each stage starts and ends with a *milestone*. Each time you add a step to a process model, it is created with a start and complete milestone as part of its configuration. You can define a stage by specifying a start milestone and a complete milestone from any step in the process. You then specify a period of time for the stage to complete.

For example, you might define a stage to monitor the process flow between the milestone at the start of step “Receive Purchase Order” and the milestone at the completion of step “Ship Product”. You can define a conditional time period for the stage to complete (for example, less than 3 days), and then take advantage of EDA (deprecated) event notifications that result from the stage behavior.

Stage behavior is evaluated by Process Tracker, a component of the webMethods Optimize analytic engine. Process Tracker evaluates stage behavior and emits the following process stage events depending on the stage behavior:

- `ProcessStageStarted`. Emitted when the stage begins.
- `ProcessStageCompleted`. Emitted when the stage completes.
- `ProcessStageBreached`. Emitted when the stage fails to meet the conditional period of time defined for the stage.

About Milestones

Software AG Designer predefines milestones at the beginning and end of a process model, as well as at the beginning and end of each step in the process. Milestones are represented in the Process Editor as *pins*, such as you might see on a map. After you create a stage, the milestones for that stage are displayed in the process editor when you select the stage in the **Stages** tab in the Properties view.

Note:

Milestone pins are not displayed for the start of process and end of process milestones.

A start stage milestone pin appears as  and an end stage milestone pin appears as . A step beginning milestone pin appears above the step on the left side, and a step completion milestone pin appears above the step on the right side. If a stage starts and ends on the same step (from step started to step completed), both milestone pins are displayed centered above the step, like this:



You can use a single milestone in multiple stage definitions. All milestone and stage information is saved with the process and stored in the Process Audit database when you build and upload the process model. For more information, see [“About Process Generation and Stage Status Display” on page 440](#) and [“About Synchronizing Process Runtime Settings with webMethods Monitor ” on page 66](#).

Working with Stages

webMethods Process Tracker emits events for stage started, stage breached, and stage completed activity. For example, you can define a condition that a stage must run to completion in more or less than a specified amount of time. If the stage runs as described, the stage is completed. If the stage execution time falls outside of the defined condition, the stage is *breached*.

If a particular stage in a process instance is breached, you may want to stop tracking of stages for the process instance. You can configure an option on any stage to stop tracking for all stages in the process if that stage is breached. This prevents unneeded processing and notifications. For more information, see [“Adding a Stage” on page 353](#).

On the **Stages** page in the Properties view of a process, you can do the following:

- [“Adding a Stage” on page 353](#)
- [“Selecting a Stage” on page 355](#)
- [“Editing a Stage” on page 355](#)
- [“Deleting a Stage” on page 356](#)

Important:

When you work with stages in Designer, you must be aware of the interaction of stage settings between Designer and webMethods Monitor. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor” on page 66](#).

Adding a Stage

Note:

Before you make modifications to process model stage settings, be aware of the interaction of these settings between Designer and webMethods Monitor. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor” on page 66](#).

➤ To add a process stage

1. Click the process editor’s canvas to select the process.
2. Click the **Stages** tab in the Properties view.
3. On the **Stages** page, click **Add Stage**.
4. Configure the fields described in the following table to define the stage:

Field	Description
Name	Type a name for the stage.

Field	Description
	<p>Note: The Name is not editable after creation. If you want to rename a stage, you must delete it and then recreate it with the new name.</p> <p>Note: There is an 80-character limit for the stage name when double-byte characters are used in an IBM DB2 database. If you are not using DB2, or if your characters are single byte, then the stage name is limited to 255 characters.</p>
Description	Optional. Type a description of the stage.
<p>🔍 Start Milestone</p>	<p>Click the list and select a milestone. Optionally, you can type characters in the text box to filter the list. When you click in the list, note a small light bulb or error icon immediately next to the list. Double-click the icon for further information. Right click the error icon for a quick-fix menu.</p> <p>Milestones are listed alphabetically. There is no default selection. Start Milestone and End Milestone selections must be different.</p>
<p>📍 End Milestone</p>	<p>Click the list and select a milestone. Optionally, you can type characters in the text box to filter the list. When you click in the list, note a small light bulb or error icon immediately next to the list. Double-click the icon for further information. Right click the error icon for a quick-fix menu.</p> <p>Milestones are listed alphabetically. There is no default selection. Start Milestone and End Milestone selections must be different.</p>
Condition	<ul style="list-style-type: none"> ■ Select < (less than) or > (greater than). Default is <. ■ Enter a positive whole number. The maximum supported values are as follows: <ul style="list-style-type: none"> ■ 2,777,777 hours ■ 166,666,666 minutes ■ 9,999,999,999 seconds ■ 9,999,999,999,999 milliseconds ■ Default is 1. ■ Select hours, minutes, seconds, or milliseconds. Default is hours. <p>The result is a condition. If the condition specifies <, then the stage is breached when the cycle time exceeds the specified time period. If the condition specifies >, then the stage is breached when the cycle time is less than the specified time period. For example:</p>

Field	Description
	< 1 hours means that the stage must complete in less than 1 hour or a ProcessStageBreached event will be emitted.
Stop Tracking On Breach	Stops stage processing for all remaining stages in the process instance when a stage breach occurs in this stage, and only one stage breached EDA (deprecated) event is emitted. Remaining stages are not tracked and will be shown as Incomplete in Monitor. The check box is cleared by default.

5. Save the process.

Selecting a Stage

> To select a process stage

1. Click the process editor's canvas to select the process.
2. Click the **Stages** tab in the Properties view.
3. On the **Stages** page, click to select a stage in the **Stages** list.

The **Stage Details** section on the **Stages** page displays the configured fields for the stage.

- The start and end steps are preselected on the model canvas, so they can be repositioned simultaneously.
- Milestone markers are displayed on the start and end stage steps.

Note:

Milestone markers are never displayed for the start of process and end of process milestones, and they are not outlined or preselected.

Editing a Stage

You can edit all stage fields except **Name**. If you want to rename a stage, you must first delete it and then recreate it with the new name.

Filtering support is provided for the **Start Milestone** and **End Milestone** fields. For more information about the stage fields, see [“Adding a Stage” on page 353](#).

Note:

Before you make modifications to process model stage settings, be aware of the interaction of these settings between Designer and webMethods Monitor. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor ” on page 66](#).

➤ To edit a process stage

1. Click the process editor's canvas to select the process.
2. Click the **Stages** tab in the Properties view.
3. On the **Stages** page, click to select a stage in the **Stages** list.
4. In the **Stage Details** section, edit any field or fields except **Name**.
5. Save the process.

Deleting a Stage

In addition to the procedure below, a stage is automatically deleted when you delete a process step that contains a stage's start milestone, or a step with a stage's end milestone.

Note:

Before you make modifications to process model stage settings, be aware of the interaction of these settings between Designer and webMethods Monitor. For more information, see ["About Synchronizing Process Runtime Settings with webMethods Monitor"](#) on page 66.

➤ To delete a process stage

1. Click the process editor's canvas to select a process.
2. Click the **Stages** tab in the Properties view.
3. On the **Stages** page, click the stage you want to delete in the **Stages** list.
4. Click **Delete Stage**.
5. Save the process.

Synchronizing Stage Settings

As a best practice, you should ensure that your process model stage settings are always synchronized between Designer and webMethods Monitor. For more information, see ["About Synchronizing Process Runtime Settings with webMethods Monitor"](#) on page 66.

1. With the process model you want to work with open in the process editor, click the design canvas to select the entire process.
2. In the Properties view, click the **Stages** tab.

3. Click **Synchronize**. Designer retrieves the stage settings from the Process Audit database and applies them to the process model.
4. Click **Save** to save the stage settings to the local workspace.

To make your changes available to the run time, you must build and upload the process model. This will overwrite the existing stage settings in the Process Audit database. For more information, see [“About Synchronizing Process Runtime Settings with webMethods Monitor ”](#) on page 66.

17 Pools and Swimlanes

■ About Pools	360
■ About Swimlanes	362

About Pools

Pools are constructs that help organize your process project. You can create pools that represent different organizations, or different parts of a single organization, such as departments.

A Designer process can have one internal pool and unlimited external pools. The process editor's is, by default, an internal pool. If you choose to create a new internal pool, the canvas pool becomes an external pool. Only steps in internal pools are generated and executed.

Note: Designer process validation ignores steps in external pools. Steps within each external pools are validated against duplicate names.

Each pool, whether internal or external, can have unlimited swimlanes, which further subdivide the organizational structure you employ.

You can choose to create horizontal or vertical pools, and you can change the orientation of an existing pool. You can set the default pool orientation in **Window > Preferences > Software AG > Process Development > Appearance**, and the default pool label color in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**. You can right-click a pool and then select **Choose Pool Label Color** to change its pool label color.

Message flow lines indicate that the process needs to send or receive information from outside the pool. Control flow lines (transitions between steps in the same pool) indicate the flow of control throughout the process.

Adding Pools

> To add a pool to the process editor's canvas

1. Click  **Pool** on the Palette, and then click the process editor's canvas to place it.
2. Select the pool type from the Set Pool Type window. The default pool type is **External Organization**.

Configuring Pools

You can configure a pool's label, description, orientation, and type in the Properties view.

You can set the default pool orientation in **Window > Preferences > Software AG > Process Development > Appearance**, and the default pool label color in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**. You can right-click a pool and then click **Choose Pool Label Color** to change its pool label color.

> To configure the properties of a pool

1. Select the title bar area of the pool.

2. On the General page in the Properties view, enter or edit descriptive information about the pool using the following table:

Property Name	Description
External	Select the check box to designate the pool as external. A process can have one internal pool and unlimited external pools.
Label	Name of the pool. Default is Pool1 .
Description	Description of the pool, for documentation purposes. Text in pool descriptions is searchable.
Orientation	Select Horizontal or Vertical from the list. The default is set in Window > Preferences > Software AG > Process Development > Appearance .

Note:

A process can have only one internal pool. If you change the type of a pool from **External** to **Internal**, the default internal pool (the process editor's canvas) will become an external pool. Only steps in internal pools are generated and executed.

Renaming Pools

You can rename pools using their shortcut menus or by selecting the pool name and editing it.

> To rename a pool

Do one of the following:

- Select the pool name and edit the highlighted text.
- Right-click the pool to open its context menu and select **Rename <current pool name>**.

Resizing Pools

You can resize pools from their bottom and right edges and bottom right corners. Pools must always be at least as large as their contents, including the swimlanes. If you resize a swimlane to be smaller, by dragging its bottom edge upward, the swimlanes below it will resize only as much as their contents are accommodated by their swimlanes.

> To resize a pool

1. Select the title bar area of the pool.
2. Click and drag one of the resize handles of the pool to resize it.

Note:

If there are steps in the pool, Designer will not allow you to resize the pool to a size at which the steps are not visible.

Removing Pools

> To remove a pool from the process editor

1. Select the title bar area of the pool.
2. Press **Delete**.
3. If the pool you are removing contains process steps, click **OK** in the Warning window that advises you that all process steps in the pool are removed when the pool is removed, and that you can use Undo (CTRL+Z) to retrieve the pool and its steps.

About Swimlanes

Swimlanes are subdivisions of pools. Where a pool typically represents a single process, swimlanes typically serve to further subdivide a process, such as by department. Each swimlane designates an actor, which becomes a property inherited by all steps in the swimlane. A pool can have an unlimited number of swimlanes, which you can organize by color.

A pool initially contains one swimlane, and the **Actor** property shares the name of the pool (by default, Pool). Additional swimlanes use the Swimlane actor by default. You can edit the actors of all swimlanes in a pool.

You can right-click a pool to move swimlanes up and down in a horizontal pool, or left and right in a vertical pool. Steps within the swimlane move with it.

You can set the default swimlane and swimlane label colors in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**. You can right-click a swimlane and then click **Choose Swimlane Color** to change the swimlane color. You can also right-click a swimlane and then click **Choose Swimlane Label Color** to change the swimlane label color.

Adding Swimlanes

> To add a swimlane to a pool

1. Right-click the pool. The location where you click will be relative to the placement of the new swimlane.
2. In a horizontal pool, select either **Add Swimlane Above** or **Add Swimlane Below** from the context menu.

3. In a vertical pool, select either **Add Swimlane Left** or **Add Swimlane Right** from the context menu.

Adding Steps to Swimlanes

You can drag and drop existing process steps from anywhere on the process editor's canvas into a swimlane. You can create new steps by dragging and dropping a document or service from the Package Navigator. You can also add a new step to a process and place it directly into a swimlane, or paste one or more copied steps into a swimlane.

Tip:

Pools are designed to expand automatically in order to accommodate steps being moved beyond current pool bounds. If you move a step past a certain (Eclipse-determined) point, it will land outside of the pool when you drop it. Expanded subprocesses are especially susceptible to this situation. Designer provides feedback cues to help you determine where a step will land when you drop it. A grey "ghost" image of the new pool bounds is displayed during the drag action when the step is to be dropped inside the pool; if the grey image of the new pool bounds is no longer displayed, your step will land outside the pool.

➤ To add a step to a swimlane

1. Click the button on the Palette that represents the type of step you want to add.
2. Click the spot in the swimlane where you want to place the step.
3. You can also add a step by dragging a document or service from the Package Navigator, or by dragging an existing process or webMethods task from the Solutions view, and dropping it into a swimlane. Finally, you can copy one or more steps and paste them into a swimlane.

Note:

Relative positions of multiple steps is preserved when you paste. If the target process editor's canvas has an asset selected, steps and/or notes and annotations are pasted relative to the selected asset. If there is nothing selected on the target canvas, steps and/or notes and annotations are pasted relative to the top left of the canvas. If you paste a step or a note/annotation very close to the edge of a pool, swimlane, or expanded subprocess, the container expands to accommodate the pasted item.

Configuring Swimlanes

In the Properties view, you can configure the actor and description for a swimlane. You can also configure the fields to include in your HTML Documentation Report.

➤ To configure the properties of a swimlane

1. Select the swimlane.

2. On the General page in the Properties view, edit the following:
 - **Actor:** by default, the **Actor** is Swimlane1.
 - **Description:** description of the swimlane, for documentation purposes. Text in swimlane descriptions is searchable.
3. On the Documentation page in the Properties view, enter or edit **Documentation Fields** and values you want to include in your HTML Documentation Report.

Documentation fields are searchable.

Moving Swimlanes

You can move swimlanes in a pool, and their steps move with them. You can use the context menu or simply drag and drop swimlanes within a pool.

> To move a swimlane

1. Select a swimlane.
2. In a horizontal pool, right-click the swimlane and select **Move Swimlane Down** or **Move Swimlane Up** from the context menu OR drag and drop a swimlane to another spot within the same pool.

Note:

Swimlanes on the upper edge of a pool can only move down, and swimlanes on the bottom edge can only move left.

3. In a vertical pool, right-click the swimlane and select **Move Swimlane Left** or **Move Swimlane Right** from the context menu OR drag and drop a swimlane to another spot within the same pool.

Note:

Swimlanes on the left edge of a pool can only move right, and swimlanes on the right edge can only move left.

Renaming Swimlanes

You can rename a swimlane using its shortcut menu, by selecting the swimlane name and editing it, or by editing the **Actor** field on the General page of the Properties view.

> To rename a swimlane

Do one of the following:

- Select the swimlane name and edit the highlighted text.

- Right-click the swimlane to open its context menu and select **Rename** <current swimlane name> .
- Select the swimlane and edit the **Actor** field on the General page of the Properties view.

Resizing Swimlanes

You can resize swimlanes from their bottom edges, with the exception of the bottommost swimlane in a pool, which cannot be resized. Swimlanes must always be at least as large as their contents.

Important:

A pool's default dimensions are 100 pixels high by 700 pixels wide, horizontally oriented. A pool's minimum dimensions are 100 pixels by 200 pixels. If you attempt to resize a pool to less than its minimum height or width, Designer resizes the pool to its default dimensions.

> To resize a swimlane

1. Select the swimlane.
2. Click and drag one of the resize handles of the swimlane to resize it.

Changing Swimlane Colors

You can set the default swimlane and swimlane label colors in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**.

You can right-click a swimlane and then click **Choose Swimlane Color** to change the swimlane color.

> To change the color of a swimlane

1. Right-click the swimlane.
2. Click **Choose Swimlane Color**.
3. In the Color window, click the color you want the swimlane to have.
4. Click **OK**.

Changing Swimlane Label Colors

You can set the default swimlane and swimlane label colors in **Window > Preferences > Software AG > Process Development > Appearance > Colors And Fonts**.

You can right-click a swimlane and then click **Choose Swimlane Label Color** to change the swimlane label color.

> To change the color of a swimlane label

1. Right-click the swimlane.
2. Click **Choose Swimlane Label Color**.
3. In the Color window, select the color you want to assign to the swimlane label.
4. Click **OK**.

Removing Swimlanes

> To remove a swimlane from a pool

1. Select the swimlane.
2. Press **Delete**.

Note:

Removing a swimlane does not remove the steps in that swimlane. Steps remain in the pool, and become part of the swimlane below (or, in a vertical pool, to the left of) the one removed. If the removed swimlane is on the right or bottom edge of a pool, steps become part of the neighboring swimlane.

18 Notes and Annotations

■ About Notes and Annotations	368
■ Working with Notes	368
■ Working with Annotations	369

About Notes and Annotations

You can add text entries to any process in the form of notes and annotations:

-  A *note* is a free-floating object on the process canvas containing text. A note can be resized and moved as required. Notes are saved with the process, their content is searchable, and they are included in generated HTML Documentation Reports.
-  An *annotation* can be associated with one or more process steps using an annotation link (a dotted line). Like notes, annotations are saved with the process, their content is searchable, and they are included in generated HTML Documentation Reports. Annotations can be moved around the canvas, but they cannot be resized like a step or note. You must edit the text by adding line breaks to change the shape of the text area.

Important:

A step and its connected annotation must be located in the same pool to maintain the association between them.

Tip:

Notes and annotations are not top-level process entities; they are members of their parent entities. For example, if an annotation is in a pool, it is a member of the pool. If an annotation is in a subprocess, it is a member of the subprocess. If you delete the parent, you also delete the annotation.

Working with Notes

Notes and annotations share some configuration features. For example, a default font style preference applies to both notes and annotations. For more information, see [“Configuring Colors And Fonts Preferences” on page 39](#).

> To configure a note

1. To set the default note color, see [“Configuring Colors And Fonts Preferences” on page 39](#).
2. To work with notes on the process canvas, do the listed in the following table:

To	Do this
Add a new note.	Click  Note on the palette, then click the process editor canvas to place the note.
Edit a note.	Click inside a note and begin typing to edit the text directly, or select the note and edit the Note Text property on the General page of the Properties view. You can also specify the font name, font size, bold, italic, and font color on the General page of the Properties view.

To	Do this
Resize a note.	Click and drag a resize handle on the edges of a selected note to make it larger or smaller.
Move a note.	Click and drag the note to another location on the canvas and drop it.
Change a note's color.	Right-click the note and click Choose Color . Select a new color in the Color dialog box. To change the default color for all process model notes, see “Configuring Colors And Fonts Preferences” on page 39 .
Delete a note.	Right-click the note and click Delete .

3. Save the process.

Working with Annotations

Notes and annotations share some configuration features. For example, a default font style preference applies to both notes and annotations. For more information, see [“Configuring Colors And Fonts Preferences” on page 39](#).

Also, the behavior of annotation link lines is subject to the settings defined for transition lines. For more information, see:

- [“About Transition Line Shape” on page 124](#) for annotation line shape information.
- [“Configuring Custom Transition Line Appearance” on page 126](#) for information about setting annotation line appearance for individual annotation links or for all annotation links in a process model.

> To configure annotations

1. To work with annotations on the process canvas, do the listed in the following table:

To	Do this
Add a new text annotation.	Do one of the following: <ul style="list-style-type: none"> ■ Click  Text on the palette, then click the process editor canvas to place the annotation. ■ Hover the cursor over the step you want to annotate to display the speed buttons, click the annotation  speed button, and then click the process editor canvas to place the annotation. See “Connect an annotation to a step,” below.
Edit a text annotation.	Click inside the annotation and begin typing to edit the text directly, or select the annotation and edit the Annotation Text property on the General page of the Properties view.

To	Do this
	You can also specify the font name, font size, bold, italic, and font color on the General page of the Properties view.
Resize an annotation.	Annotations have no resize handles and cannot be manually resized. However, you can change the text line length by inserting a hard (SHIFT) or soft (SHIFT+ENTER) return in the text to narrow the text box.
Connect an annotation to a step.	<p>Right-click the annotation and click Add Link to connect the link to a process step. You can initially connect the link to any of a step's four attachment points. For activity steps, you can select the link and then drag the link endpoint to any location around the activity step's border (this is not possible for event and gateway steps).</p> <p>You can connect an annotation to two or more steps, and you can create two or more association lines from one anchor point on the annotation.</p>
Move an annotation.	Click and drag the annotation to another location on the canvas and drop it. If the annotation is associated with one or more process nodes, the representative connecting lines reposition automatically.
Delete an annotation.	Right-click the annotation and click Delete .

2. Save the process.

19 KPIs

■ About KPIs	372
--------------------	-----

About KPIs

KPIs (Key Performance Indicators) are quantifiable measurements that reflect critical success factors of a company, department, or project. In *webMethods Business Activity Monitoring (BAM)*, KPIs are defined by dimensions, measures, and dimension hierarchies. A KPI is related to a dimension hierarchy, which is related to a group of dimensions, which is related to a measure.

For example, a KPI might be comprised of:

- Order Total (measure) per customer (dimension) per region (dimension hierarchy)

A *Measure* is an integer field or decimal field that contains data such as sales (units sold) and profits. For example, an Order Total is a measure. KPIs compare measures with dimensions.

A *Dimension* is a string field used to categorize data in order to enable users to answer business questions. Commonly used dimensions are customer, product, promotion, channel, and time.

A *Dimension Hierarchy* is a group of dimensions that show relationships among different dimensions. For example, a customer (dimension) is part of a region (dimension).

In Designer, you can:

- Define step-level KPIs
- Define dimensions for step-level and process-level KPIs
- Assign dimensions to process-level and step-level KPIs
- Remove dimensions from KPIs
- Delete dimensions
- Upload KPIs when you build and upload a process for execution
- Upload KPIs explicitly (without building and uploading a process for execution)
- Configure preferences for uploading KPIs
- Configure Optimize Server preferences for uploading KPIs

About Dimensions

A dimension is a string field that is used to categorize data in order to enable users to answer business questions. Commonly used dimensions are customer, product, promotion, channel, and time. For example, each sales channel of a clothing retailer might gather and store data regarding sales and reclamations of their clothing assortment. The retail chain management can analyze the sales of its products across all stores over time and help answer questions such as:

- What is the effect of promoting one product on the sale of a related product that is not promoted?
- What are the sales of a product before and after a promotion?

- How does a promotion affect the various distribution channels?

Defining Dimensions

After you have defined a KPI, you define dimensions to associate with it. In a given process, a dimension can be used only once.

Tip:

Although you select a KPI when you define a dimension, you can use any dimension with any KPI.

> To define a dimension

1. Select a step in the process editor.
2. Click the KPIs page in the Properties view.
3. In the KPIs for *<Step Label>* section, select a KPI.
4. In the Dimensions section, click **Add Dimension**.
5. In the Add Dimension window, enter a Dimension Label.

Note:

Dimension labels must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names and dimension labels on the target server. However, if you duplicate a KPI name or dimension label in a process, generation of the process produces a warning.

6. Click  **Add Association** to browse for an **Associated Field** for the new dimension.
7. In the Choose Field window, click the output field with which you want to associate the dimension, and then click **OK**.

Note:

When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.

Deleting Dimensions

You can delete dimensions you defined. Deleting a dimension removes it from all KPIs that use it.

> To delete a dimension

1. Select a step in the process editor.
2. Click the KPIs page in the Properties view.
3. In the KPIs for *<Step Label>* section, select a KPI.
4. In the Dimensions section, select the **Description** field of a dimension.
5. Click **Delete Dimension**.

Removing Associated Fields from Dimensions

You can remove a dimension's associated field. A dimension without an associated field cannot be used in a KPI.

> To remove an associated field from a dimension

1. Select a step in the process editor.
2. Click the KPIs page in the Properties view.
3. In the KPIs for *<Step Label>* section, select a KPI.
4. In the Dimensions section, click  **Remove Association** next to an **Associated Field** to remove that associated field.

Note:

When you remove an association of an output field from a dimension or a KPI, Designer does not remove the field from the Logged Fields page in the Properties view.

About Process-Level KPIs

Process-level KPIs apply to the process as a whole. Designer includes the following pre-defined process-level KPIs on the Auto Generated tab of the KPIs page in the Properties view of a process:

- **Volume**
- **Cycle Time**
- **Error Count**

For each of these, **Name** is automatically populated and read-only. You can enter a **Description** for documentation purposes. The other fields displayed are not available for process-level KPIs.

About Step-Level KPIs

Step-level KPIs apply to a step in a process. You can define as many step-level KPIs as you want for a given step. KPIs cannot be reused in multiple steps; each step has its own KPIs.

You can view the KPIs included in an open process by using the Outline view. Click a KPI nested under a step and click it. The step that contains the KPI is highlighted in the process editor.

Defining Step-Level KPIs

> To define a step-level KPI

1. In the process editor, click a step for which you want to define a step-level KPI.
2. Click the KPIs page in the Properties view.
3. On the Business KPIs tab in the KPIs for <Step Label> section, select a KPI.
4. Click **Add KPI**.
5. In the Add KPI window, enter a Label for Business KPI, and click **OK**.
6. With the KPI selected, define the properties described in the table below:

Property	Description
Name	Name of the KPI Note: KPI names must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names and dimension labels on the target server. However, if you duplicate a KPI name or dimension label in a process, generation of the process produces a warning.
Description	Description of the KPI
Unit of Measure	How the KPI is measured
Associated Field	Associate the KPI with actual fields in the process. Only fields that are available in the output of the step can be used. Note: When you associate an output field with a KPI or a dimension, Designer automatically adds the field to the Logged Fields page in the Properties view. Designer does not, however, automatically remove an output

Property	Description
	field from the Logged Fields page in the Properties view if you remove it from a KPI or dimension.
Aggregation Type	Aggregation type can be set to SUM , AVERAGE , or LAST VALUE

7. Add dimensions for the KPI (see [“Defining Dimensions” on page 373](#)) or select the check boxes corresponding to existing dimensions. A dimension must have an associated field to be used in a KPI.

Important:

For any one step, all the associated fields for all measures and dimensions must be fields that are contained in the same document. Different steps in a process can use different documents, so long as any one step uses fields from only a single document.

Modifying Step-Level KPIs

Modifying a step-level KPI involves editing its on the KPI page in the Properties view of the step. You can edit fields, add new dimensions, delete existing dimensions, and change the dimensions used to measure the KPI.

Deleting Step-Level KPIs

➤ To delete a step-level KPI

1. In the process editor, select a step that has a step-level KPI you want to delete.
2. Select the KPIs page in the Properties view.
3. On the Business KPIs tab in the KPIs for *<Step Label>* section, select a KPI.
4. Select **Delete KPI**.

Removing Dimensions from Step-Level KPIs

➤ To remove dimensions from a step-level KPI

1. In the process editor, select a step that has a step-level KPI.
2. Select the KPIs page in the Properties view.
3. On the Business KPIs tab in the KPIs for *<Step Label>* section, select the KPI with a dimension you want to remove.

4. In the Dimensions section, clear the check box that corresponds with the dimension you no longer want to associate with the KPI.

Note:

When you remove an association of an output field from a dimension or a KPI, Designer does not remove the field from the Logged Fields page in the Properties view.

About Uploading KPIs to Optimize

You can upload KPIs to Optimize when you build and upload a process for execution. You can also upload KPIs explicitly, without building and uploading the process for execution.

Set the default Designer behavior for KPI upload behavior in Build and Upload preferences. Configure the Optimize server to which you want to upload KPIs in Optimize Server preferences.

When you upload KPIs from Designer to Optimize, you overwrite all KPIs for that process that have already been uploaded. The exception to this is KPIs that have been edited using My webMethods Server. If you want to upload KPIs for a process whose KPIs have been edited in My webMethods Server, you must first use My webMethods Server to delete the KPIs. After you have done this, you can upload KPIs for the process from Designer again.

Important: Designer cannot upload KPIs for steps whose names contain special characters, as determined by the operating system used.

Important:

KPI names and dimension labels must be unique on a given Optimize server. Designer cannot, at design time, detect all the KPI names and dimension labels on the target server. However, if you duplicate a KPI name or dimension label in a process, Designer warns you of this when you upload KPIs, whether when building and uploading a process for execution, or when uploading KPIs explicitly (without building and uploading a process for execution).

Uploading KPIs to Optimize when Building and Uploading a Process for Execution

➤ To upload KPIs to Optimize when building and uploading a process for execution

1. Configure an Optimize server in Optimize Server preferences.
2. Configure the **Automatically Upload KPIs on Build** option in Build and Upload preferences to either **Always** (if you don't want a prompt) or **Prompt** (if you do). The default setting is **Prompt**.
3. Open a process that contains KPIs.
4. Click **File >  Build and upload for execution**

OR

Click  **Build and upload for execution** on the tool bar.

Uploading KPIs to Optimize Explicitly (without Building and Uploading a Process for Execution)

➤ To upload KPIs to Optimize explicitly (without building and uploading a process for execution)

1. Configure an Optimize server in Optimize Server preferences.
2. Configure the **Prompt to Upload KPIs** option in Build and Upload preferences to either **Never** (if you don't want a prompt) or **Always** (if you do). The default setting is **Always**.
3. Open a process that contains KPIs.
4. Click **File >  Upload KPIs**

OR

Click  **Upload KPIs** on the tool bar.

20 Process Documentation

■ About Process Documentation	380
■ Adding Documentation Fields	380
■ Assigning Documentation Field Values	380
■ Removing Documentation Fields	381
■ Generating Documentation Reports	381
■ Configuring Default Documentation Fields Preferences	382

About Process Documentation

Process documentation enhances the collaborative effort inherent and critical to many process projects. For example, the initial process may be designed by a business analyst, who then passes the project on to a developer. Using process documentation, the business analyst can describe elements of the process design, methodology, and intent to the developer. This information is stored with the process project, and can also be collected into a printable HTML report. Information about KPI definitions, step transition conditions, step inputs and outputs, and process annotations are also captured in the HTML Documentation Report. Text in documentation fields is searchable.

Adding Documentation Fields

You can add documentation fields and corresponding values for individual processes, activities, events, gateways, and swimlanes.

> To add a documentation field

1. Select process component to which you want to add a documentation field. Do one of the following:
 - Click the process editor's canvas to select the entire process
 - Click to select an event, activity, gateway, or swimlane
2. On the Documentation page in the Properties view, click  **Add Documentation Field**.
3. Type a name for the new field.
4. Save the process.

Assigning Documentation Field Values

You can add documentation fields and assign corresponding values for individual processes, events, activities, gateways, and swimlanes.

> To assign a documentation field value

1. Select the process component to which you want to assign a documentation field value.
 - Click the process editor's canvas to select the entire process
 - Click to select an event, activity, gateway, or swimlane
2. On the Documentation page in the Properties view, select the field in the **Documentation Fields** list whose value you want to assign. Documentation fields specific to this component as well as default documentation fields are listed.

3. In the **Value for Field: <field name>** box, enter or edit the information about the field or process that you want to store. You can click the field label to show or hide the text box underneath it.

Note:

The description must include only text. Preface a URL with http:// or https:// to create a clickable link in the HTML output; URLs are not clickable in the text box.

Removing Documentation Fields

➤ To remove a documentation field

1. Select the process component from which you want to remove a documentation field.
 - Click the process editor's canvas to select the entire process
 - Click to select an event, activity, gateway, or swimlane
2. On the Documentation page in the Properties view, click **✖ Remove Documentation Field**.

Generating Documentation Reports

You can generate an HTML Documentation Report of your Designer process at any point during process development. For processes that originate in ARIS, you can also generate a Documentation Report from the Solutions view using the **ARIS** context (right-click) menu. This can either be an explicit documentation report upload to ARIS, or as an option when synchronizing with ARIS. For more information, see [“Working with ARIS Processes in the Solutions View” on page 21](#) and [“Synchronizing ARIS Processes” on page 404](#).

➤ To generate a documentation report

1. In Designer: **File > Generate Documentation Report**.
2. In the Generate HTML Documentation Report dialog box, specify the settings described in the following table:

Setting	Description
Report File Name	By default, the name of the process is used, but you can specify a different name. Type the name you want to use.
	Note: If you select an SVG Process Image Type , Designer requires that the Report File Name contain only ASCII characters.

Setting	Description
Process Image Type	<p>When you create a report, you also create an image file of your process. Click the button preceding the type of image you want the report to generate (PNG, JPEG, or SVG).</p> <p>Note: If you select an SVG Process Image Type, Designer requires that the Report File Name contain only ASCII characters.</p>
Sort By	You can choose to list the steps in your process in several ways. Click the button preceding Ascending by Step Name , Descending by Step Name , or Task/Event Order .
CSS Style Sheet	<p>You can format your HTML Documentation Report using a CSS. Click the button preceding Default Style, Classic Style, or Custom CSS.</p> <p>If you choose to use a Custom CSS, your style sheet must be located in the same folder as the Save to Location. Specify the file name in the text box.</p>
Save to Location	Documentation Report files are saved by default to the root of the workspace parent directory with the name of the process project (c:\Documents and Settings\ <user name="">\<process name>.html), but you can specify a name and location of your choice. Click Browse to navigate to and select the directory in which you want to save the report.</user>

Note:

Standard Windows file naming limitations apply. If you enter a file name that is not allowed, Designer prompts you to enter a different name.

- After the report has finished generating, Designer opens it automatically using Eclipse's default browser. The default browser setting is at **Window > Preferences > General > Web Browser**.

Configuring Default Documentation Fields Preferences

You can define default documentation fields in your Designer workspace. Default documentation fields apply to all Designer processes, and are always included in generated HTML Documentation Reports. You can edit these properties; export them to an XML file for use in other Designer installations; and import them from an XML file. You don't have to enter values for all global documentation definition fields, but you can't remove them except from the Default Documentation Fields page in the Preferences window.

When you remove default documentation fields, you remove them from all future Designer projects. Existing projects retain the fields that have values assigned to them, but new ones will not have them at all.

On the Default Documentation Fields page in the Preferences window, you can add and remove default documentation fields for processes, events, activities, gateways, and swimlanes. You can also import and export files that contain these default documentation fields, making sharing default documentation fields among related projects easy.

Default documentation fields are represented by the  **Default Documentation Field** icon, allowing you to distinguish a default field from a local one.

Adding Default Documentation Fields

> To add a default documentation field

1. In Designer: **Window > Preferences > Software AG > Process Development > Default Documentation Fields**
2. Click the type of asset to which you want to add the default documentation field.
3. Click **Add**, and type the name you want to give the field.

Editing Default Documentation Fields

> To edit a default documentation field

1. In Designer: **Window > Preferences > Software AG > Process Development > Default Documentation Fields.**
2. Select the default documentation field you want to edit.
3. Select the name of the field to enable the cursor, and edit the name.

Removing Default Documentation Fields

> To remove a default documentation field

1. In Designer: **Window > Preferences > Software AG > Process Development > Default Documentation Fields.**
2. Click the default documentation field you want to remove, and click **Remove**.

Importing and Exporting Default Documentation Fields

> To import or export default documentation fields

1. In Designer: **Window > Preferences > Software AG > Process Development > Default Documentation Fields.**
2. Click **Import** or **Export**, depending on which you want to do.
3. Browse to select the file to import from or export to (an XML file), and click **OK**.

If you imported, new fields appear in the Default Documentation Fields list.

Note:

If you import default documentation fields you already have, they remain unchanged. Only new fields are added.

21 Working with Escalation Processes

■ Escalation Process Overview	386
■ Importing Closed Loop Analytic Assets into Designer	386
■ Activating the WmClosedLoopAnalytics Package	386
■ Working with the Default Escalation Process	387
■ Working with the Default Escalation Task	388
■ Working with the Default webMethods Rule Projects	388
■ Mapping Data Fields in the Wrapper Services	388
■ Triggering a Test Escalation Process	390

Escalation Process Overview

The Closed Loop Analytics feature of the webMethods product suite enables you to trigger an escalation process when an error or other supported event occurs. Within this escalation process, you can create a task that can be assigned to one or more My webMethods Server users, enabling them to take action in response to the escalated event.

For more information about the predefined assets included in Closed Loop Analytics, see *webMethods Closed Loop Analytics Help*.

Importing Closed Loop Analytic Assets into Designer

Your Closed Loop Analytics installation contains predefined assets that you can import into your Designer workspace and then customize.

➤ **To import the predefined Closed Loop Analytics assets into your workspace**

1. In Designer: **File > Import**.
2. In the Import dialog box, click **General** and then **Existing Projects into Workspace**. Then click **Next**.
3. Click **Browse** and navigate to this directory in your Software AG installation:
\\Solutions\ClosedLoopAnalytics\ProjectSources
4. Click **OK** to accept the directory and place it in the **Select root directory** field. The following projects are displayed in the Projects list:
 - ClosedLoopAnalyticsProcessProject
 - EscalationTaskApplication
 - WmClosedLoopAnalytics (rules project)

If the projects are not already selected in the **Projects** list, select them.

5. Click **Finish**.

The projects appear in the Solutions view, and in the Navigator view, available in the UI Development perspective.

Activating the WmClosedLoopAnalytics Package

Note:

Prior to executing this procedure, you must install the Closed Loop Analytics feature using Software AG Installer (**Solutions > Closed Loop Analytics** in the Installer product selection page).

Your Closed Loop Analytics installation contains predefined Integration Server package that enable you to develop your own customized escalation solutions. You must copy this package to the \packages directory in your Integration Server instance, and then activate the package in Integration Server Administrator.

➤ To activate the predefined WmCloseLoop Analytics package

1. Open this directory in your Software AG installation:
 \Solutions\ClosedLoopAnalytics\ProjectSources\ISPackage
2. Copy the WmClosedLoopAnalytics directory to:
 \IntegrationServer\instances*instanceName*\packages
3. In a browser, open the Integration Server Administrator for the targeted instance.
4. Click **Packages > Management > Activate Inbound Packages**.
5. Make sure WmClosedLoopAnalytics is selected, then click **Activate Package**.
6. In the Package Navigator view in Software AG Designer, make sure the view is connected to the target Integration Server, and then refresh the view to see the WmClosedLoopAnalytics package.

You can expand the package in the Package Navigator view to access its contents, and open the contents in an appropriate editor:

- DocType: wm.closedloopanalytics.docTypes:DueDateOffset
- DocType: wm.closedloopanalytics.docTypes:ProcessEscalation
- Service: wm.closedloopanalytics.service:getOffsetDate
- Service: wm.closedloopanalytics.service:StartEscalationProcess
- Trigger: wm.closedloopanalytics.service:ProcessEscalationJmsTrigger:ProcessEscalationJmsTrigger

Working with the Default Escalation Process

The default process is pre-configured to interact with the other default assets of Closed Loop Analytics. In other words, you do not need to make any changes to the process to obtain the expected run-time behavior. However, in certain cases you may need to provide data mapping, as described in [“Mapping Data Fields in the Wrapper Services” on page 388](#). In addition, a task error handling step is included, but will not provide any action unless you specify an Integration Server service or a web service to process the error.

Also, the process can complete with one of two end message event steps, Task Expiration Reminder and Task Error Message. These steps are configured to publish the pipeline of the service in the form of the ProcessEscalation IS document type. If you want to take further action based on these outputs, you will need to create your own custom applications to handle them.

For more information about the default escalation process workflow and customizing the default escalation process, see *webMethods Closed Loop Analytics Help*.

Working with the Default Escalation Task

The default task is pre-configured to interact with the other default assets of Closed Loop Analytics. In other words, you do not need to make any changes to the task to obtain the expected run-time behavior. It is pre-loaded with the business data from the ProcessEscalation document type, and contains a pre-configured set of task events.

Note:

To be able to be available to the run-time process instance, the task application must be published to My webMethods Server, as described in the *webMethods BPM Task Development Help*. You can publish the application manually if you prefer, as described in the *webMethods BPM Task Development Help*. However, when you build and upload the escalation process, Designer will automatically publish the task application to a connected My webMethods Server if the application does not already exist there. A prompting dialog box is displayed by default, enabling you to cancel the publishing of the task application if you do not want it published at this time.

For more information about customizing the default escalation task, see *webMethods Closed Loop Analytics Help*.

Working with the Default webMethods Rule Projects

You can integrate the imported escalation rules into your escalation process or escalation task to provide a very wide range of processing capability based on the data delivered in the EscalateProcess IS document type (or your own custom document type).

For more information about working with rules, see:

- The *webMethods Closed Loop Analytics Help*.
- The *webMethods BPM Task Development Help* for information about using webMethods business rules with task applications.
- [“About webMethods Business Rules” on page 100](#) for information about using business rules with process models.
- The *webMethods BPM Rules Development Help* for information about working with webMethods business rules in general.

Mapping Data Fields in the Wrapper Services

The solution component ClosedLoopAnalyticsProcessProject contains all of the generated wrapper services for all process steps, and all mapping is present by default, even after a new build and upload of the process project. Manual mapping is necessary only if the process package is not present on the Integration Server, or if you have renamed the process project, which results in a new Integration Server process package with a different name after build and upload, or if you have customized the process by adding new inputs or outputs, or adding new steps.

In any of these cases, you must manually map the following fields using the default input and output mapping services:

- TaskAssignee
- DueDate
- Status

Note:

If you build and upload the process project to a package with a different name, you must do the mapping again.

➤ **To map the default wrapper service**

1. Build and deploy your escalation process project, then execute the process with the ProcessEscalation doc type using **Run As > Publish as JMS Message**.
2. In the process editor, right-click the **Escalation Task** step and click **Edit Input Data Mapping**.
3. From the editor Palette view, add a **Map** step to the **Escalation_Task_InputMapService** tab.
4. With the **Map** step selected, click the Pipeline view
5. Map the **TaskAssignee** field:
 - a. From the **pub.list** section of the Pipeline Palette view, drag an **appendToStringList** element to the Transformers section of the editor.
 - b. In the **Pipeline In** column, expand **ProcessEscalation** and create a mapping line from the **TaskAssignee** field to the **appendToStringList** transformer, selecting **fromList** in the Link To dialog box.
 - c. In the **Pipeline Out** column, expand **EscalationTaskInputEnv** and create a mapping line from the **assignedToList** field to the **appendToStringList** transformer, selecting **toList** in the Link To dialog box.
 - d. Save your changes.
6. Map the **DueDate** field:
 - a. In the **Pipeline In** column, expand **ProcessEscalation** and create a mapping line from the **DueDate** field to the **expireDate** field in **EscalationTaskInputEnv** in the **Pipeline Out** column.
 - b. Save your changes.

7. In the process editor, right-click the **Escalation Task** step and click **Edit Output Data Mapping**.
8. Click the **Map** step in the **Escalation_Task_OutputMapService** tab, and map the **Status** field:
 - a. In the **Pipeline In** column, expand **TaskCompletionInfo** and create a mapping line from the **status** field to the **taskCompletionState** field in the **Pipeline Out** column.
 - b. Save your changes.

Triggering a Test Escalation Process

Prior to placing your escalation process into a production environment, you might want to run in your development environment to test its behavior.

> To trigger a test escalation process

1. Be sure you have completed the data mapping described in [“Mapping Data Fields in the Wrapper Services” on page 388](#), as well as any other modifications you want to include. In addition, make sure the WmClosedLoopAnalytics package is activated on your targeted Integration Server as described in [“Activating the WmClosedLoopAnalytics Package” on page 386](#).
2. Build and upload the process to your targeted Integration Server.
3. In the Package Navigator view, in the WmClosedLoopAnalytics package, right-click **wm > closedloopanalytics > docTypes > ProcessEscalation**.
4. Click **Run As > Publish as JMS Message**.
5. In the Enter Input dialog box:
 - a. In the **JMS connection name alias** field, click the browse button and select `PE_NONTRANSACTIONAL_ALIAS`.
 - b. In the **Destination name** field, click the browse button and select `projectName_processName_SUBQUEUE`
 - c. In the **Destination type** list, make sure **Queue** is selected.
 - d. Select the **Prepare message for BPM** check box.
 - e. In the JMS Message Body section, type some test data to populate the document type, for example, originator, due date, task assignee, and so on.
 - f. Click **OK**.

Successful publication of the JMS message is shown in the Results view. The published document should trigger a new instance of the escalation process.

You can log into to My webMethods Server and browse to **Navigate > Applications > Monitoring > Business > Process Instances** to view the resulting process instance.

You can browse to **Navigate > Applications > Monitoring > Business > Tasks > Task List Management** to view the resulting task instance (assuming you have the correct permissions). You can also view the task in its own task type inbox, as well as in the My Inbox page for any assigned user.

22 Process Integration with ARIS

■ ARIS Process Integration	394
■ Configuring ARIS Server Preferences	397
■ About Importing ARIS Processes as XPDL Files	397
■ About Importing ARIS Solutions	402
■ Synchronizing ARIS Processes	404

ARIS Process Integration

Software AG Designer is integrated with ARIS so that the users of these tools may easily share and collaborate, freely passing process models from ARIS to Designer and back.

You can import ARIS models in the following ways:

- As a process. For more information, see [“Using the Process Import Wizard for ARIS Imports” on page 399](#).
- As part of a solution file. For more information, see [“Importing an ARIS Solution” on page 402](#).

ARIS supports model exchange both with and without CentraSite. By default, models are shared directly, with no use of CentraSite. After a CentraSite configuration is set, it is used instead. This is configured in ARIS when the database is created. For CentraSite support, the **select Model-to-execute-scenario** property must be selected, and the CentraSite connection properties must be specified.

Important:

ARIS requires a CentraSite configuration to exchange Web services in a process model.

Important:

After you set the database to use CentraSite, you cannot change it back.

When the database in ARIS is configured to use CentraSite, it is used as the repository where all its process models are saved.

The ARIS server manages roles and access to the models in the repository, enforcing the process life cycle, whether or not you use CentraSite. It also controls which users belong to which roles, and which roles have access to which models and for what purpose.

To import a model from ARIS, Designer connects to an ARIS server using the credentials on the ARIS Server preferences page. See [“Configuring ARIS Server Preferences” on page 397](#).

When Designer is connected to the ARIS server, refreshing the ARIS Tasks view in the Process Development perspective retrieves the up-to-date list of ARIS tasks. See [“The ARIS Tasks View” on page 396](#). Each ARIS task in the ARIS Tasks view represents a process model. In Designer, you can accept one of these ARIS tasks and then import the associated process model to work on it. See [“Importing an ARIS Process \(XPDL File\)” on page 398](#).

The resulting .process file contains the model as represented by a conversion from BPMN to XPDL to Designer. It includes:

- Pools and swimlanes
- Events (with many restrictions)
- Activities and subprocesses
- Referenced child processes
- Extensions to maintain ARIS ObjectIDs for all of the above when re-exported to ARIS later

- Extra metadata about the model in ARIS
- Server location of the ARIS model
- The model's ObjectID

Note:

The imported model does not contain DataObjects (XSDs as Integration Server document types).

When work on the process model in Software AG Designer is finished, the Designer user can update the original ARIS version by publishing an update to ARIS. ARIS users can then view the process in ARIS. Further development should be done in Designer. See [“Synchronizing ARIS Processes” on page 404](#).

For more information about ARIS-Designer integration and configuration, see ARIS documentation.

ARIS Process Transformation

The first steps of process model sharing are executed in ARIS, where a user creates a process model and transforms it to BPMN. The ARIS user configures the details of the sharing, including what is being shared and who it will be shared with.

On the ARIS server, the administrator configures roles and who is assigned to those roles. When all the details are set and the process model is ready, the ARIS user shares it with IT (Software AG Designer) by clicking **Start “Share with IT” governance flow**.

This creates an XPDL version of the ARIS process and makes it available for import into Designer. The ARIS server queues a task and sends an email notification to all registered ARIS server users belonging to the WM_PROCESS_DEVELOPER role that the process is ready for implementation.

In Software AG Designer, you have access to the shared ARIS process model and its documentation on the **ARIS Model** page in the Properties view of the process. You can open the model in ARIS using the **ARIS Download Client** link, and in browser-based **ARIS Connect**. You can also view the **ARIS Process Documentation**.

Note:

Your editing permissions in ARIS tools are defined by your license as well as by the state of the model in the “Share with IT” governance flow.

ARIS requires a CentraSite configuration to use Web services in a process model. If a Software AG Designer user adds Web services to an ARIS model that does not use CentraSite, those Web services are not displayed when the model is returned to ARIS. They are, however, retained in the model and displayed in Designer.

ARIS Extended XPDL Attributes

ARIS supports some extensions that Software AG Designer uses in its processes to allow a more complete process model exchange. These extensions are preserved as extended attributes in the XPDL during process transformation.

The following table lists the extensions and the ARIS extended attributes.

Extension	ARIS Extended Attribute
Error Handler	ext:errorHandler=" <i>error handler Activity ID</i> "
Cancel Handler	ext:cancelHandler=" <i>cancel handler Activity ID</i> "
Timeout Handler	ext:timeoutHandler=" <i>timeout handler Activity ID</i> " Additional timeoutHandler attributes: <ul style="list-style-type: none"> ■ ext:isStaticTimeout="<i>true or false</i>" ■ ext:timeoutValue="<i>static timeout value in milliseconds (does not apply if the timeout is not static)</i>"

When you import an ARIS model with these extended attributes, Designer recognizes the appropriate handler activities.

```
<WorkflowProcesses ext:cancelHandler="fc42a46c-7138-11e3-6e9c-af8089f2ab89"
ext:errorHandler="fc42a469-7138-11e3-6e9c-af8089f2ab89" ext:isStaticTimeout="true"
ext:timeoutHandler="fc42a47b-7138-11e3-6e9c-af8089f2ab89" ext:timeoutValue="10000">
```

The ARIS Tasks View

The ARIS Tasks view displays tasks in the ARIS system (managed by the ARIS server) that correspond to process models that are ready for implementation in Software AG Designer.

When an ARIS user shares a process, an ARIS task is created, and Software AG Designer displays the task and its associated process model in the ARIS Tasks view. Use this view to import ARIS processes into Designer for implementation or editing.

Click the column headers to sort the list by **Model** or by **Task**. The initial list displayed is unsorted, and this option remains available after being resorted. The ARIS Tasks view includes tooltips that reflect the following properties of the selected ARIS model in the ARIS Tasks view.

The following table lists these properties:

Property	Value
ARIS database	Database where the process model is located
ARIS Model ID	ARIS process model identification (ID)
ARIS Model Name	ARIS process model name
ARIS user	Username of the ARIS user associated with the process model
CentraSite URL	URL of the CentraSite database
CentraSite Model ID	Process ID in CentraSite

Note: CentraSite URL and Site Model ID data require a CentraSite database configuration. These fields are blank for models without a CentraSite database. See [“ARIS Process Transformation” on page 395](#).

If you close the ARIS Tasks view and want to reopen it, click **Window > Show View > Other > ARIS Integration >  ARIS Tasks**.

Configuring ARIS Server Preferences

To connect to an ARIS server, you must first enter its connection information on the ARIS Server preferences page at **Window > Preferences > Software AG > Process Development > ARIS Server**.

> To configure ARIS Server preferences

- On the **ARIS Server** preference page, configure the listed in the table below:

Property Name	Description
ARIS Server URL	ARIS server URL
User ID	User ID
User Password	Password

- Click **Test Connection** to test the configured connection.

Connection information is stored in Eclipse Secure Storage.

About Importing ARIS Processes as XPDL Files

To import a model in XPDL format from ARIS, Designer connects to an ARIS server, and you must use the credentials on the ARIS Server preferences page. See [“Configuring ARIS Server Preferences” on page 397](#). To import an ARIS model in a solution file, see [“Importing an ARIS Solution” on page 402](#).

When an ARIS user shares a process with you, an ARIS task is created, and the ARIS server sends you an email message to notify you that the task is available. This ARIS task corresponds to the shared ARIS process model; you can import it and work on it in Designer.

You can edit the model in Software AG Designer as you would any process; you can add steps, flows, mapping, adapter services, and anything else available in Designer. You can also generate and upload, debug, simulate, and run the process.

ARIS XPDL imports include the CentraSite web service location in the following format: `centrasite://uddi:unique hexadecimal value`. This enables Designer to offer a CentraSite option when launching the Web Service Creation Wizard to locate or create web service connectors in webMethods Integration Server.

Important:

To ensure the proper exchange of web services between Designer and ARIS, you must use the Process Navigator view to set the **Reuse** property of each web service connector to **Public**. Be sure to publish each web service connector to CentraSite before sending the model back to ARIS.

If your database configuration does not use CentraSite, web services are not supported. If a Designer user adds a web service, it is stored in the process model and available in Designer, but not available for display or editing in ARIS.

Avoiding CentraSite Predefined Services for Import/Export

CentraSite supports its own set of predefined web services. *Do not use these predefined services in process models that will be imported from ARIS and exported back to ARIS from Designer .*

If these services are present in an imported model, the services may not be recognizable by ARIS when the model is exported back to ARIS. The following is a list of these services:

- UDDI Publish API Services
- UDDI Inquiry Services
- UDDI Custody and Ownership Transfer API
- UDDI Security Service
- LifecycleManagementService
- ImportXPDLService
- SearchService
- ApprovalService
- RegistrationService
- ImportWsdService
- ImportXsdService

Importing an ARIS Process (XPDL File)

This procedure describes how to import an ARIS process model in XPDL format directly into Designer. For information about importing a process in a solution file, see [“Importing an ARIS Solution” on page 402](#).

> To import an ARIS process

1. In the ARIS Tasks view, click **Refresh** to update the list. Each ARIS model corresponds to an ARIS task. Click the column name to sort the list by **Model** or by **Task**, alphanumerically up or down as shown in the header. Click again to return to the initial (default) unsorted list.

Note:

The Action column is not sortable.

2. Click the cell in the Action column in the same row as the task you want to execute to enable the **Run** button for the specified action. Click **Run** to execute the task.
3. Complete the Process Import Wizard. See [“Using the Process Import Wizard for ARIS Imports” on page 399](#).
4. The Progress Information dialog box displays the import progress, and prompts you to overwrite existing resources when duplicate file names exist. When the import is complete, click Save to save the import report to a file (*.txt). This action opens the Save As dialog box, where you can specify a file name and location for the report.
5. Click **Close** to close the Progress Information dialog box. Designer automatically opens all processes (.process files) created from those imported.

Tip:

If the imported ARIS process contains steps inside pools and swimlanes, the resizing of these steps in Designer may cause them to move to the wrong swimlane. To resolve this and restore steps to their proper swimlanes, simply right-click on the process and select **Layout steps in swimlanes**. This will adjust the pool and swimlane heights to ensure that the steps are put in the appropriate lanes.

Using the Process Import Wizard for ARIS Imports

The Process Import Wizard works in a particular way for ARIS imports; there are a few considerations specific to the ARIS integration. If you are importing XPDL in the course of an ARIS import, follow these instructions.

Note:

If you want to import a process in a solution file, see [“Importing an ARIS Solution” on page 402](#).

➤ To use the Process Import Wizard for an ARIS import

1. From the **Select process project** list, select an existing project into which to import the XPDL file. If you want to create a new process project for the XPDL file, click **New Project** to open the New Process Project dialog box.
2. From the **Select an Integration Server** list, select an Integration Server that will be the server for the steps in the imported process, and where the imported XPDL that refers to data will be located. If you need to configure Integration Servers, click the **Configure Integration Servers** link to open the Integration Servers preferences page.
3. If the XPDL file you are importing contains multiple WorkflowProcess constructs, Designer combines them all into a single process by default. The **Create a single process model from multiple XPDL WorkflowProcess constructs** check box is automatically selected. Clear the

check box if you do not want Designer to create a single process from multiple WorkflowProcess constructs (see the notes below for more information).

Tip:

For processes coming from ARIS, you typically want to use the default values, since the WorkflowProcess constructs generally represent pools.

Note:

To create a single process design for multiple XPDL WorkflowProcess constructs, each WorkflowProcess must be mapped to a pool in the XPDL file with a valid Process attribute (Process ID). Additionally, each pool must have specified size and coordinates. If these conditions are not satisfied, Designer creates multiple .process files as the default behavior. When this situation occurs, the event is recorded in the Import Log.

Note:

If you clear the **Create a single process model from multiple XPDL WorkflowProcess constructs** check box, importing XPDL files containing multiple WorkflowProcess constructs results in multiple Designer processes (.process files). Multiple imported processes are automatically named based on identifying information (**Name** or **ID**) of the WorkflowProcess constructs in the XPDL file.

4. Designer checks for duplicate step names during the import process, automatically incrementing each duplicate by 1 (Step, Step1, Step2, and so on). This protects against build errors due to duplicate step names. If you do not want your duplicate step names changed, clear the **Enforce Step Name Uniqueness** check box, which is selected by default. Click **Next**.
5. If the XPDL file being imported contains call activities that refer to child processes, Designer checks for the presence of these child processes in the Designer workspace. If they exist, the log reflects this fact and the import continues. If the referenced child processes do not exist in the workspace, Designer alerts you of this and lists these child processes. You must import them before importing the parent process. Click **Cancel** to stop the import wizard, then **Accept implementation** and **Import new process** for each child process listed.
6. When all referenced child processes of the parent call activity exist in the Designer workspace, **Accept implementation** and **Import** the parent process. Designer sets the Implementation of the call activity.
7. CentraSite is required for Web service exchange. If your database does not use CentraSite, skip to step 13.
8. If your database uses CentraSite and the XPDL file being imported contains references to Web services that implement activities, Designer displays the Web Service Descriptors page of the wizard. For each Web Service Descriptor (WSD) you want to use, click to select the browse button in the **Run WSC wizard** column in the row that contains it.

Important:

Restrictions apply concerning CentraSite predefined services. For more information, see [“Avoiding CentraSite Predefined Services for Import/Export” on page 398](#).

9. On the Web Service Descriptors page, select the Create Default Web Service Descriptors at `<Project>.<Process>` check box to automatically create and assign Web service connectors at the default location in the Integration Server namespace.

Designer searches for pre-existing matching Web Service Descriptors automatically. If a WSDL has already been imported as a part of this or another process, Designer locates it. If Designer finds any WSDs, it displays the Select Web Service Connector page of the wizard.

10. On the Select Web Service Connector page, do one of the following:
 - If you find a Web service connector (WSC) you want to use, click to select it and then click **OK** to return to the Web Service Descriptors page.
 - If you do not find a WSC you want to use, click **Cancel** to create a new Web Service Descriptor (WSD).
11. If you click **Cancel** to create a new Web Service Descriptor (WSD), Designer displays the New Web Service Descriptor page of the wizard. Provide the information listed in the table below to create a new WSD:

Server	Integration Server Name, or Default for the defined default Integration Server
URL	Integration Server URL
Package	Integration Server package name
Namespace	Select the parent namespace from the Integration Server tree
Element name	New Web Service Descriptor (WSD) name

12. Click **Finish** to create the new WSD. If you click **Cancel**, no WSD is created, and no WSC is assigned to the XPDL activity.
13. Click **Finish** to start the import.

Re-Importing an ARIS Process

You can re-import a previously imported ARIS process in the Solutions view.

Note:

This procedure applies only to models imported in XPDL format, and does not apply to models imported as part of a solutions file. To re-import a process in a solution file, you must run the import wizard again, as described in [“Importing an ARIS Solution” on page 402](#).

➤ To re-import an ARIS process in XPDL format

1. In the Solutions view, right-click on the process you want to re-import from ARIS.

2. Click **ARIS > Reimport ARIS Process**.
3. Complete the Process Import Wizard. See [“Using the Process Import Wizard for ARIS Imports” on page 399](#).

About Importing ARIS Solutions

Software AG Designer provides support of model-to-execute (M2E) by enabling you to import an ARIS solution design into a process project within Designer. An ARIS solution design can contain any or all of the following assets:

- **Process:** a BPMN collaboration diagram. The diagram is exported from ARIS as an XPD file and is imported into Designer as a .process file.
- **Data:** a data structure diagram. The diagram is exported from ARIS as an XSD file and is imported into the Integration Server as an IS document.
- **Screen:** a screen design diagram. The diagram is exported from ARIS as an XHTML file and is imported into Designer as an XHTML file in an webMethods OpenUI project.

All asset types are optional. For example, a solution design can contain only data, or screens, or all assets types.

The export of the ARIS content is done in the ARIS. The export action generates a container file in zipped format with the filename extension "m2e". The solution import wizard in the Designer can import this m2e file and generate the required assets (.process, IS Document type, and XHTML files).

The process diagram in ARIS may have references to data in a Data Structure diagram. In this case the import wizard retains these references between the webMethods assets. (for example, the .process file will reference IS documents types).

Accessing the ARIS Solution Import Wizard

➤ To access the ARIS Solution import wizard

1. In Software AG Designer, open the Process Development perspective.
2. In the Solutions view or the Navigator view, right click any solution or project, or any of their assets, then click **Import**.
3. In the Import wizard, expand the **Software AG** node, click **ARIS Solution design**, and then click **Next**.

Importing an ARIS Solution

Prerequisites

- Prior to beginning this procedure, you must export a solution file from ARIS, and locate the exported file in a directory that is accessible from Designer. The file must have a file name extension of “.m2e”. For information about exporting a solution file, see the ARIS documentation.
- If the solution contains screens (XHTML files):
 - The Designer Composite Applications feature must be installed.
 - If the Composite Applications feature is installed, at least one OpenUI project must exist as a target for the imported screen files. For more information about working with OpenUI projects, see the *webMethods CAF and OpenUI Development Help*.
- An active connection to an Integration Server must be available.

➤ To import an ARIS Solution

1. Open the ARIS Solution import wizard as described in [“Accessing the ARIS Solution Import Wizard” on page 402](#).
2. Click **Select** and locate the exported *.m2e file you want to import. Select the file, then click **Open**, and then click **Next**.
3. If the solution file contains data structures, the next page in the wizard enables you to specify the namespace for each data structure in the solution file. Select the Integration Server, package, and folder where you want to import the data structure, then click **Next**. The wizard will provide a separate namespace specification page for each data structure in the solution file. Continue to click **Next** until all data structure namespace locations are specified. If no data structures are present, this step is omitted.
4. If the solution file contains screens (XHTML files), the next page in the wizard enables you to specify the CAF project that the screen files will be imported into. Screens are imported in the WebContent folder of the specified project. Select a CAF project from the list in the **Project** field. Accept the ARIS name for the screen, or type a new name in the **Name** field.

Important:

If a file with the same name already exists in the CAF project, you will be asked if you want to overwrite the existing file when you click **Next**. If you click **Yes**, the existing file will be replaced with the imported file.

The wizard provides a separate CAF project selection page for each screen file in the solution file. Continue to click **Next** until all screen file target projects are specified. If no screens are present, this step is omitted.

5. If the solution file contains a process file (XPDL), the next page in the wizard is the standard XPDL import wizard. Follow the procedure described in [“Using the Process Import Wizard for ARIS Imports” on page 399](#).

In all cases, the **Finish** button of the ARIS Solution import wizard is enabled when all components of the *.m2e file have been imported.

Synchronizing ARIS Processes

You can synchronize an ARIS process to upload your changes and push them to ARIS users, request a model review, delegate the implementation to someone, or finalize the implementation. All ARIS users in the relevant ARIS server role(s) receive a notification that the updated model is available and can view the model in ARIS.

➤ To synchronize an ARIS process

1. In the Solutions view, right-click the ARIS process model you want to synchronize.
2. Click **ARIS > ARIS Synchronization**.
3. In the ARIS Synchronization dialog box, select an **Action**:
 - **Request review for model** to request a model review
 - **Upload for update in ARIS** to push your changes to ARIS in response to an **Initiate update**
 - **Finish implementation** to mark a model as complete
 - **Delegate implementation** to delegate the implementation to someone

Optionally, enter a **Description** and a **Message** to accompany the selected **Action**.

4. Click the **Upload Documentation Report for the process** check box to invoke the Generate HTML Documentation Report dialog box and upload the Process Documentation Report with the model. Refer to [“Generating Documentation Reports” on page 381](#) for more information.
5. Designer runs the Process Export Wizard in the background and sends the XPDL to the stored ARIS server location configured in ARIS Server preferences. See [“Configuring ARIS Server Preferences” on page 397](#).
6. The ARIS server queues a task and sends an email notification to the ARIS user that includes information about the process available to be updated, and links to do so in ARIS. IDs are maintained from Designer back into ARIS.
7. After the ARIS server task is accepted by an ARIS user, it disappears from the ARIS Tasks view in Designer.

23 Importing and Exporting Processes

■ About Importing and Exporting Processes	406
■ BPMN to XPDL Mapping	420

About Importing and Exporting Processes

You can import Designer processes from Designer process files and you can export Designer processes to Designer process files or XPDL files. You can also import XPDL and Modeler 6.x processes into Designer.

XPDL files exported from Designer are compatible with ARIS, and can be imported there. See [“Process Integration with ARIS” on page 393](#).

You can  **Import** and  **Export** files in the Solutions view as well, using the buttons in the view. See [“Importing and Exporting Assets from the Solutions View” on page 25](#).

Important:

There is a `.config` file associated with the `.process` file. It is recommended that you manage the process using the Solutions view, which automatically handles these two files together. See [“About the Solutions View” on page 19](#).

Designer can format imported XPDL processes to match the tools used to create them, and can create process simulations directly from imported XPDL files that contain simulation information. It can also create a single process model from multiple XPDL elements. Designer supports XPDL 2.2, which is backward-compatible with XPDL 2.1, 2.0, and 1.0.

Note:

The Designer Process Simulation feature must be installed to create a process simulation.

Importing Designer Processes

You can import an existing Designer process file into a new or existing process project.

➤ To import a Designer process file

1. In Designer: **File > Import > Software AG >  Process File**.
2. Click **Next**. Designer displays the Process File Import Wizard.
3. Click **Browse** to select a `.process` file to import.
4. Select the **Process Project** from the list OR click **New** to create a new process project for your new process.
 - If you create a new process project, Designer displays the New Process Project window. Type a **Project name** and select the workspace in which to save it. For more about creating process projects, see [“Creating Process Projects” on page 48](#).
5. Click **Finish**.

Importing Designer Processes Containing One or More Referenced Processes

➤ To import a Designer process that contains one or more referenced processes

If a process contains one or more referenced processes, you can import them together in either of two ways:

- Import each process separately, using **File > Import > Software AG >  Process File** as described in [“Importing Designer Processes”](#) on page 406.
- Import the entire process project that contains both the parent process and the referenced child process or processes, using **Import > General > Existing Projects into Workspace**.

Exporting Designer Processes

You can export a Designer process file.

➤ To export a Designer process file

1. In Designer: **File > Export > Software AG >  Process File**.
2. Click **Next**.
3. In the Process File Export window, select the files you want to export:
 - If you want to select multiple processes in a process project, select the check box corresponding to the process project on the left to select the entire process project on the right. Click the .project file to clear the check box; only .process files can be exported.
 - If you want to select one or more process files manually, without selecting the entire process project on the left, select the name of the process project and then select the boxes corresponding to the process files you want on the right. If you want to remove a check mark, select the box again to clear it.
4. When you have selected the files for export, click **Browse** to select the directory in your file system where you want to place the exported files.
5. If you want Designer to **Overwrite existing files without warning**, select the check box in the Options section. The option is not enabled by default.
6. Click **Finish**.

Note:

If you select anything besides process files for export, when you click **Finish**, Designer will prompt you to remove the non-process files before you can complete the export.

Importing Modeler Processes

You can import Modeler 6.x processes into Designer. To access the process or processes you want to import, either connect to the Modeler design server (the Integration Server that contains the WmModeler package) or specify the location of the Modeler 6.x client installation with an offline model.

» To import a Modeler 6.x process into Designer

1. In Designer: **File > Import > Software AG >  Modeler 6.x process models**
2. Select the import **Mode**:
 - **Connect to Design Server** (default)
 - **Offline**
3. Do one of the following:
 - In **Connect to Design Server** mode, enter the **Server**, **User**, and **Password**.
 - In **Offline** mode, enter the **User**, and then **Browse** to select the **Modeler 6.x Installation Directory** to access the repository file where the offline model is stored.
4. Click **Next**.
5. In the Select Process Models to be imported window, select the process or processes to import.
6. Click **Finish** to perform the import.

Note:

The repository file in Modeler exists only if Offline mode has been used at least once.

Designer displays the number of selected processes, upgraded processes, and errors.

A summary of the import process is saved in the ModelUpgradeLog.txt file, which is located in your workspace at

```
<workspace_name>\.metadata\plugins\com.process.webmethods.upgrade.impl\logs
```

7. Click **View Upgrade Log** to open ModelUpgradeLog.txt in Designer, or click **OK** to exit.

Important:

After you import a Modeler 6.x process, check to make sure all referenced documents and services exist on the specified Integration Server for each step.

Note:

When you import Modeler 6.x steps with external subscription documents, Designer creates two steps: a receive step and an activity step; the receive step is configured to use the

subscription document. In Modeler 6.x, you could define a complex join condition based on the arrival of the subscription document, like Receive-OrderDoc OR Transition-from-Step2. Designer adds extra flow steps after the receive steps to keep the complex join condition intact.

Modeler to Designer Conversions

Some conversions are made to make Modeler processes work in Designer. The following is a reference of those conversions.

Process Conversions

The following table lists those conversions.

Modeler 6.x	Designer
Flow step	Service task + Integration Server service type
Flow step + document subscription	Message start event with receive document and subscription (for publishable documents) protocol + (Service task with an Integration Server service type). Includes a transition between the message start event and the service task.
Workflow step	User task
Any step + inputs/outputs	Step + inputs/outputs If the step is a flow step, it is a Service task with an Integration Server service type. If the step is a Workflow step, it is a User task.
Internal group	If the model contains multiple internal groups, steps in an internal group are extracted to the canvas pool, as long as the internal group is not nested in another external group. If the internal group is nested, its steps are extracted to the outermost group.
External group	External pool For nested groups, contents of inner groups are extracted to the outermost group. Designer does not support nested pools.
Note	Note (annotation)
External step	n/a
Web Service step	Service task + Web service type
Subprocess (inline)	All steps moved into main process
Referenced process step	Call activity + Referenced Process type

Modeler 6.x	Designer
Empty step	Abstract task
Terminate step	End terminate event
Any step + transition	Step + transition
Any step + document subscription	Message start event with receive document and subscription (for publishable documents) protocol + (Service task with an Integration Server service type). Includes a transition between the message start event and the service task.
Any step + join	Step + join
Logical server	Integration Server Name
Global data	n/a
Data properties	n/a
Web Service interaction	n/a
To Do list	n/a
Process documentation	Process documentation
Step documentation	Step documentation
Join	Join
Join type: OR	Join type: Unsynchronized OR
Join type: XOR	Join type: OR
Join type: AND	Join type: AND
Join type: Complex	Join type: Complex
Join type: XPATH	n/a
Transition condition	Transition condition
Transition condition: XPATH	n/a
Transition condition: Retries exceeded	Transition condition: Step iterations exceeded
Transition condition: Error	Intermediate boundary error event
Transition condition: Timeout	Join timeouts remain as timeout transitions. For step timeouts, an intermediate interrupting boundary timer event is added to the step, with a transition drawn from it to the next step that handles the timeout.

Modeler 6.x	Designer
Transition condition: Else	Transition condition: Else
Transition condition: If > Then	Transition condition: If condition

Note:

The Modeler Process Key is retained when importing a Modeler 6.x process into Designer. The Process Key is used when you **Upload for Analysis Only**, and allows the imported Modeler process to work in My webMethods Server. See [“About Building and Uploading a Process” on page 434](#) for more information about uploading a process for analysis.

Step Conversions

The following table lists those conversions.

Modeler 6.x	Designer
Flow step properties	Service task + Integration Server service type properties
Description	n/a
Logged fields	Logged fields
Step duration	n/a
Join duration	Join timeout
Maximum iterations	n/a
Correlation service	Correlation service
Service invoke	Service invoke
Documentation	Documentation
Step label	Label
Enable resubmission	n/a
Workflow step properties	User task properties
Description	n/a
Logged fields	Logged fields
Logical server	Integration Server Name
Documentation	Documentation
Project	n/a

Modeler 6.x	Designer
	n/a
Implementation module	n/a
Step duration	n/a
Join duration	Join timeout
Maximum iterations	Maximum iterations
Correlation service	Correlation service
Service invoke	Service invoke
Logical server	Integration Server Name
Step label	Label
Enable resubmission	n/a
Web Service step properties	Service task + Web service type properties
Description	n/a
Step label	Label
Logged fields	Logged fields
Logical server	Integration Server Name
Documentation	Documentation
Step with External Subscription documents defined as inputs/outputs	Receive task
Step with External Publication documents defines as inputs/outputs	Reply task
Web Service	Service task + Web service type
Port type	Port type
Operation	Operation
Input	Inputs/outputs
Output	Inputs/outputs

Modeler 6.x	Designer
Enable resubmission	n/a
Empty step properties	Abstract task properties
Description	n/a
Logical server	Integration Server Name
Step label	Label
Documentation	Documentation
Terminate step properties	Terminate step properties
Description	n/a
Step label	Label
Logical server	Integration Server Name
Documentation	Documentation

Importing XPDL Processes

You can import XPDL processes into Designer. Designer supports XPDL 2.2, which is backward-compatible with XPDL 2.1, 2.0, and 1.0.

Designer can format imported XPDL processes to match that of the tools used to create them, such as ARIS and Fujitsu's Interstage. It can create single or multiple Designer process models from imported XPDL containing multiple WorkflowProcess elements, and can create one or more process simulations directly from imported XPDL files that contain simulation information. Designer checks for duplicate step names during the import process, automatically incrementing each duplicate by 1 (Step, Step1, Step2, and so on).

Important:

When you manually import XPDL from ARIS, you can select **Export to be used by ARIS tools**. This configures the XPDL to use workspace references for call activity child processes, rather than the child processes themselves. This allows simultaneous editing of the child processes and the parent process, and is selected by default when importing from the **ARIS Tasks** view. When using this option, first import the child processes to your workspace, and then import the parent process.

Note: Designer's layout of imported XPDL files is dependent upon the information contained therein. There are a great number of tool-specific layout options; Designer uses the available information in the file to create its best approximation of the original tool's layout. Some tool-specific formats, such as extended attributes, cannot be used for conversion. When information is not present in the import file or cannot be used for conversion, Designer creates the process layout based on its auto-layout feature.

Note:

The Designer Process Simulation feature must be installed to create a process simulation. See *webMethods BPM Process Simulation Help*.

Import status is displayed during the import process, and includes information about the import progress and resulting files and artifacts being generated. The full import details are presented as a final report when the import process is complete, and can be exported to a file. The report includes the following information:

- List of generated artifacts
 - Processes
 - Process simulations
 - Integration Server documents with their fully-qualified names
- Informational messages
 - XPDL data objects to which generated Integration Server documents correspond
 - Process simulation resources created and their corresponding scenarios/steps
 - Process models replaced as a result of the import
 - Web service connectors and the process model steps referencing them
- Import errors

> To import an XPDL process

1. In Designer: **File > Import > Software AG >  XPDL File**.
2. In the Import window, click **Next**.
3. In the XPDL Import Wizard window, do the following:
 - a. Click the **Browse** button to navigate to the XPDL file you want to import. In the Open window, click **Open** to select the file and populate the **Select file** field.
 - b. From the **Select process project** list, select an existing process project into which to import the XPDL file. If you want to create a new process project for the XPDL file, click **New Project** to open the New Process Project window.

For instructions on creating a new process project, see [“Creating Process Projects” on page 48](#).

- c. Optionally, edit the name of the process resulting from the XPDL import in the **Process name** field.

- d. From the **Select an Integration Server** list, select an Integration Server that will serve as the server for the steps in the imported process, and where the imported XPDL that refers to data will be located. If you need to configure Integration Servers, click the **Configure Integration Servers** link to open the Integration Servers preferences page. See "Integration Servers Preferences" in *webMethods Service Development Help*.

Tip:

When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

- e. If the XPDL you are importing contains multiple WorkflowProcess constructs, Designer combines them all into a single process by default. The **Create a single process model from multiple XPDL WorkflowProcess constructs** check box is automatically selected. You can clear the check box if you do not want Designer to create a single process from multiple WorkflowProcess constructs.

Note:

To create a single process design for multiple XPDL WorkflowProcess constructs, each WorkflowProcess must be mapped to a pool in the XPDL with a valid **Process** attribute (Process ID). Additionally, each pool must have specified size and coordinates. If these conditions are not satisfied, Designer creates multiple Designer process files, as this is its default behavior. When this situation occurs, the event is recorded in the Import Log.

Note:

If you clear the **Create a single process model from multiple XPDL WorkflowProcess constructs** check box, importing XPDL files containing multiple WorkflowProcess constructs results in multiple Designer processes (.process files). Importing XPDL files containing multiple WorkflowProcess constructs that contain activities with SimulationInformation results in multiple Designer simulations (.simulation files). Multiple imported processes or simulations are automatically named based on identifying information (Name or ID) of the WorkflowProcess constructs in the XPDL.

Important:

When you manually import XPDL from ARIS, you can select **Export to be used by ARIS tools**. This configures the XPDL to use workspace references for call activity child processes, rather than the child processes themselves. This allows simultaneous editing of the child processes and the parent process, and is selected by default when importing from the **ARIS Tasks** view. When using this option, first import the child processes to your workspace, and then import the parent process.

- f. If **Export to be used by ARIS tools** is selected and the XPDL being imported contains call activities that refer to child processes, Designer checks for the presence of these child processes in the Designer workspace. If they exist, the log reflects this fact and the import continues. If the referenced child processes do not exist in the workspace, Designer alerts you of this and lists these child processes. You must import them before importing the parent process. Click **Cancel** to stop the import wizard, then **Accept Task** and **Import** each child process listed.

- g. Designer checks for duplicate step names during the import process, automatically incrementing each duplicate by 1 (Step, Step1, Step2, and so on). This protects against build errors due to duplicate step names. If you do not want your duplicate step names changed, clear the **Enforce Step Name Uniqueness** check box, which is selected by default. Click **Next**.
- h. If the XPDL you are importing contains inputs/outputs, Designer displays the Inputs and Outputs window. Do the following:
- To create Integration Server documents, select the **Create Integration Server Documents** check box in the Inputs / Outputs section. This enables you to create Integration Server documents for the **DataObjects** selected in the table. If the XPDL does not contain any data to be mapped to Integration Server documents, the Inputs / Outputs section is not enabled.
 - The table displays the information listed in the table below:

DataObject	A DataObject is an XPDL element that contains one or more DataField elements. A DataObject can contain internally defined DataFields , such as Strings and Numbers, or DataFields that refer to Complex XML Data Types defined in schema definitions (XSDs).
DataField	Name of the externally referenced data as used in the XPDL file being imported. A DataField is a child of a DataObject .
External Document URL	The URL of the XSD (schema definition) where the Complex XML Data Type (data definition) resides
Create	Select the check box to create the Integration Server document for the DataObject in the same row. You can also use the Select All and Clear All buttons to select or clear all the check boxes in the table at once.

- Click **Finish**
- If you selected **DataObjects** from which to create Integration Server documents, you must specify their creation location (or locations). The Create a New Document Type window is pre-populated with **Server**, **URL**, and **Element name** values. You can edit the **Element name** in the text box.

The table below describes these values, along with **Package** and **Folder**.

Server	The Integration Server selected on the Inputs / Outputs page of the wizard. If no selection is made, <code>Default</code> is displayed to represent the default Integration Server.
URL	The address of the selected Server

Package	The Integration Server package containing the folder where you create the new Integration Server document
Folder	The folder where you create the new Integration Server document
Element name	The default Element name is based on the corresponding DataObject on the Inputs / Outputs page of the wizard. You can edit this name.

- j. Expand the Integration Server tree to select the target folder for the new Integration Server document and populate the **Package** and **Folder** fields.

Tip:

A valid package containing a valid folder must exist before you can select them. You cannot create a new package or folder in the Create a New Document Type window.

- k. Click **Finish** to create the new Integration Server document.

Tip:

If you click **Next** in the Create a New Document Type window, you can select a source for your Integration Server document in the Select a Source window. These options are pre-selected based on the data type being imported. Internally defined data types select **None**, which is the default option. Externally referenced data types select **XML Schema** and also display the **External Document URL**. If you click **Finish**, you bypass the viewing of these pre-selected options.

- l. If you are creating multiple Integration Server documents, populate the **Package** and **Folder** fields, and optionally edit the **Element name** in the Create a New Document Type window for each one. Then click **Finish**.
- m. If the XPDL being imported has been edited using multiple tools, you must choose the tool whose layout you want to import. In the Select Tool ID section of the XPDL Layout window, click the **Select layout** list to see the available options and select the one you want. These options are dependent upon the information in the XPDL file. Click **Next**.
- n. If the XPDL being imported contains simulation information, and you have the Designer Process Simulation feature installed, you can create one or more process simulations (.simulation files) from the import as well.
- If the import wizard finds simulation information, it displays the Create Simulation window with the option to **Create simulation** already selected.
 - If the import wizard finds information for a single simulation, the **Simulation Name** field is populated with the same name entered for the **Process name** in the XPDL Import Wizard window. You can edit the process simulation name without affecting the process name. If the import wizard finds information for multiple simulations, the resulting .simulation files are automatically named based on identifying information (Name or ID) of the WorkflowProcess constructs in the XPDL.

- c. If you do not wish to create any process simulations, clear the **Create simulation** check box.
- o. If the XPDL being imported contains references to Web Services that implement activities, Designer displays the Web Service Descriptors page of the wizard. For each Web Service Descriptor (WSD) you want to use, click to select the browse button in the **Run WSC wizard** column in the row that contains it.

Important:

Restrictions apply concerning CentraSite predefined services. For more information, see [“Avoiding CentraSite Predefined Services for Import/Export” on page 398](#).

- p. Designer searches for pre-existing matching Web Service Descriptors automatically. If a WSDL has already been imported as a part of this or another process, Designer locates it. If any are found, Designer displays the Select Web Service Connector page of the wizard.
- q. On the Select Web Service Connector page, do one of the following:
 - If you find a WSC you want to use, click to select it and then click **OK** to return to the Web Service Descriptors page of the wizard.
 - If you do not find a WSC you want to use, click **Cancel** to create a new Web Service Descriptor.
- r. If you click **Cancel** to create a new Web Service Descriptor (WSD), Designer displays the New Web Service Descriptor page of the wizard. Provide the information listed in the table below to create a new WSD:

Server	Integration Server Name, or Default for the defined default Integration Server
URL	Integration Server URL
Package	Integration Server package name
Namespace	Select the parent namespace from the Integration Server tree
Element name	New Web Service Descriptor (WSD) name

- s. Click **Finish** to create the new WSD. If you click **Cancel**, no WSD is created, and no WSC is assigned to the XPDL activity.
4. Designer checks for duplicate step names during the import process, automatically incrementing each duplicate by 1 (Step, Step1, Step2, and so on). This protects against build errors due to duplicate step names. If you do not want your step names changed, clear the **Enforce Step Name Uniqueness** check box, which is selected by default.
 5. Click **Finish** to start the import.

6. The Progress Information window displays the import progress, and prompts you to overwrite existing resources when duplicate file names exist. When the import is complete, click **Save** to save the import report to a file (*.txt). This action opens the Save As window, where you can specify a file name and location for the report.
7. Click **Close** to close the Progress Information window. Designer automatically opens all processes (.process files) created from the imported XPDL. Simulations (.simulation files) do not open automatically. You can open them in the Solutions view.

Exporting XPDL Processes

You can export Designer process files as XPDL files. Exported XPDL files are compatible with ARIS and can be imported there. See [“Process Integration with ARIS” on page 393](#).

When a Designer process file is exported as an XPDL file, the webMethods-specific .process properties are preserved as extended properties so that they may be used upon the future return of the process file to Designer.

Tip: Designer process properties not required by other tools but required for Designer re-import are preserved as webMethods Extended Properties. For the list of these extended properties, refer to the schema document located at `<Designer installation folder>\eclipse\v36\plugins\com.softwareag.process.xpdl.model_<build number>\schema\xpdl21WMPProperties.xsd`. This file is also located at *Software AG_directory\Designer\util*.

➤ To export an XPDL process

1. In Designer: **File > Export > Software AG >  XPDL File**.
2. Click **Next**.
3. In the XPDL File Export window, select the files you want to export:
 - If you want to select multiple processes in a process project, select the check box corresponding to the process project on the left to select the entire process project on the right. Select the .project file to clear the check box; only .process files can be exported as XPDL.
 - If you want to select one or more process files manually, without selecting the entire process project on the left, select the name of the process project and then select the boxes corresponding to the process files you want on the right. If you want to remove a check mark, select the box again to clear it.
4. When you have selected the files for export, click **Browse** to select the directory in your file system where you want to place the exported files.

5. If you want Designer to **Create an XPDLP WorkflowProcess construct for each pool in the Designer process model**, select the check box in the XPDLP Mapping section. This option is selected by default. If you clear the check box, Designer places all steps in the resulting XPDLP file into a single WorkflowProcess construct.
6. If you want Designer to **Overwrite existing files without warning**, select the check box in the Options section. The option is not selected by default.
7. Click **Finish**.

BPMN to XPDLP Mapping

Designer maps BPMN constructs to XPDLP constructs when you convert Designer processes to XPDLP files, and maps XPDLP constructs to BPMN constructs when you convert XPDLP files to Designer processes. The following describes the relationships between the two formats.

Tip:

For more information about XPDLP, see the specification at <http://www.wfmc.org>.

Mapping Designer BPMN 2.0 Process Models to XPDLP

The following table describes the relationship between BPMN constructs and XPDLP constructs.

Designer BPMN 2.0 Process Model Part XPDLP Construct

Abstract task	An Activity (does not contain any implementation) <pre><Activity Name="Task1" Id="3a15b9a1-a97e-43b7-85ad-8565c4cff736"> <Description> </ns4:Description> <NodeGraphicsInfos> <NodeGraphicsInfo Width="75.0" Height="47.0" LaneId="a130eab8-b249-4dd1-a609-7c1362069886" PageId="702c212f-e358-42e8-aed8-492d6ec6208ePAGE"> <Coordinates YCoordinate="63.0" XCoordinate="256.0"/> </NodeGraphicsInfo> </NodeGraphicsInfos> </Activity> <Activity Name="Task1" Id="3a15b9a1-a97e-43b7-85ad-8565c4cff736"> <Description> </ns4:Description> <NodeGraphicsInfos> <NodeGraphicsInfo Width="75.0" Height="47.0" LaneId="a130eab8-b249-4dd1-a609-7c1362069886" PageId="702c212f-e358-42e8-aed8-492d6ec6208ePAGE"> <Coordinates YCoordinate="63.0" XCoordinate="256.0"/> </NodeGraphicsInfo> </Activity></pre>
Service task (using CentraSite)	An Activity with a TaskApplication

Designer BPMN 2.0 Process XPDL Construct Model Part

An <Application> construct is created for each Web Service referenced by the Service Task. The "Id" attribute of the <TaskApplication> matches the "Id" of the <Application>

```
<Activity Id="0c1b500c-8c13-11e2-5b38-efea8f839e4d"
Name="ws1"> <Description> </Description> <Implementation>
<Task> <TaskApplication
Id="c9adb277-16bd-11e2-8565-c38280f44478"></TaskApplication>
</Task> </Implementation> </Activity> <Application
Id="c9adb277-16bd-11e2-8565-c38280f44478"
Name="calculateCreditIncreaseProposal"> <Description>
</Description> <ExternalReference
location="centrasite://uddi:c8b102f7-16bd-11e2-8565-f675722b0be7"
xref="calculateCreditIncreaseProposal"></ExternalReference><ExtendedAttributes>
<ExtendedAttribute Name="SST"
Value="c8b102f7-16bd-11e2-8565-f675722b0be7"></ExtendedAttribute>
</ExtendedAttributes> </Application>
```

Service task (NOT using CentraSite)

An Activity with a <TaskService> construct

```
<Activity Id="89d57a0c-8c1f-11e2-5b38-efea8f839e4d"
Name="ws7"> <Description /> <Implementation> <Task>
<TaskService /> </Task> </Implementation> <NodeGraphicsInfos>
<NodeGraphicsInfo BorderColor="0,0,0" FillColor="136,187,255"
Height="61" LaneId="89d57a05-8c1f-11e2-5b38-efea8f839e4d"
PageId="738ef641-8c1f-11e2-5b38-efea8f839e4dPAGE" Width="98">
<Coordinates XCoordinate="121" YCoordinate="271" />
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>
```

Business rule task

An Activity with a TaskBusinessRule implementation

```
<Activity Name="Task1"
Id="4d34c7a1-4d62-4dad-87b1-deee02578ead"> <Description>
</Description> <Implementation> <Task> <TaskBusinessRule/>
</Task> </Implementation> <Acvitivity>
```

User task

An Activity whose implementation is a TaskUser + Performer

```
<Activity Name="Task1"
Id="4d34c7a1-4d62-4dad-87b1-deee02578ead"> <Implementation>
<Task> <TaskUser> <Performers> <Performer>test_</Performer>
<Performer>abc</Performer> </Performers> </TaskUser> </Task>
... </Activity>
```

Manual task

An Activity with a TaskManual implementation

```
<Activity Name="Task1" Id="S24"> <Implementation> <Task>
<TaskManual> <Performers> <Performer>test_</Performer>
```

Designer BPMN 2.0 Process XPD L Construct Model Part

	<pre><Performer>abc</Performer> </Performers> </TaskManual> </Task> ... </Activity></pre>
Send task	<p>An Activity whose implementation is a TaskSend</p> <pre><Activity Name="Task8" Id="4d34c7a1-4d62-4dad-87b1-deee02578ead"> ... <Implementation> <Task> <TaskSend/> </Task> </Implementation> </Activity></pre>
Receive task	<p>An Activity with a TaskReceive implementation</p> <pre><Activity Name="Task9" Id="4d34c7a1-4d62-4dad-87b1-deee02578ead"> ... <Implementation> <Task> <TaskReceive Instantiate="false"/> </Task> </Implementation> ... </Activity></pre>
Call activity calling either a BPMN Callable Process or a webMethods Referenced Process	<p>Results in multiple WorkflowProcess elements in XPD L, one for the parent process, and one for either the BPMN Callable Process or the webMethods Referenced Process. The parent process has an Activity whose implementation is a subflow. The ID attribute of the subflow matches the ID of the call activity.</p> <pre><Activity Name="Call Activity1" Id="4d34c7a1-4d62-4dad-87b1-deee02578ead"> <Description> </Description> <Implementation> <SubFlow Id="xpd l1_c1"/> </Implementation> <NodeGraphicsInfos> <NodeGraphicsInfo Width="75.0" Height="47.0"> <Coordinates YCoordinate="280.0" XCoordinate="105.0"/> </NodeGraphicsInfo> </NodeGraphicsInfos> </Activity></pre> <p>The referenced process is exported as a WorkflowProcess whose ID matches the subflow ID.</p>
Subprocess	<p>There are two parts to subprocess mapping.</p> <ol style="list-style-type: none"> 1. The subprocess itself is represented as BlockActivity element: <pre><BlockActivity View="EXPANDED" ActivitySetId="a6b99dac-2746-4d4a-82e4-02e68bad55adACTIVITYSET"/></pre> 2. The contents of the subprocess (steps, etc. within the subprocess) are represented as an ActivitySet: <pre><ActivitySet Name="Subprocess1" Id="a6b99dac-2746-4d4a-82e4-02e68bad55adACTIVITYSET"> <Activities> <Activity Name="Message Event1" Id="7b9176b8-b8c0-489f-bc54-80ceb36d6f8b"> <Description> </Description> <Event> <StartEvent Trigger="Message"/></pre>

Designer BPMN 2.0 Process XPD L Construct Model Part

```
</Event> <NodeGraphicsInfos> <NodeGraphicsInfo Width="35.0"
Height="35.0"
PageId="3c1f437e-1216-4045-8062-825a7702e55bPAGE">
<Coordinates YCoordinate="39.0" XCoordinate="64.0"/>
</NodeGraphicsInfo> <Activity> <Activity Name="Message Event2"
Id="418db8db-e5e6-49bc-95db-f914c7de4292"> ... </Activity>
</ActivitySet>
```

Subprocess loop marker on a subprocess or a call activity A marker that shows that the subprocess or call activity should be looped. Looping behavior is based on a Boolean condition. The subprocess or call activity will loop as long as the Boolean condition is true. The condition is evaluated for every loop iteration, and may also be evaluated at the beginning or at the end of the iteration. Another option is to specify a numeric cap, the value of which the number of iterations may not exceed.

```
<Activity Name="Subprocess1"
Id="d61205b2-dec8-4fdf-9ce0-c3a56ba97bba"> <Description>
</Description> <BlockActivity View="EXPANDED"
ActivitySetId="d61205b2-dec8-4fdf-9ce0-c3a56ba97bbaACTIVITYSET"/>
<Loop LoopType="Standard"> <LoopStandard TestTime="Before"
LoopMaximum="123"/> </Loop> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="336.0" Height="227.0"
PageId="006f3646-2ad3-49fc-a50c-249fd9b516cbPAGE">
<Coordinates YCoordinate="74.0" XCoordinate="323.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>
```

Standard loop marker on a task Regular (standard) loop on a task

```
<Loop LoopType="Standard"> <LoopStandard TestTime="Before"
LoopMaximum="10"/> </Loop>
```

Note:

Standard looping is not available for any type of BPMN task in Designer 9.0.

Parallel multiple instance (MI) marker Loop with a Parallel attribute
not supported in Designer 9.0

Sequential multiple instance (MI) marker Loop with a Sequential attribute
not supported in Designer 9.0

Compensation marker isCompensating attribute

Event subprocess not supported in Designer 8.2

Error handler

Designer BPMN 2.0 Process XPD L Construct Model Part

Timeout handler	
Cancel handler	
Sequence flow	<p>A Transition. The "From" attribute refers to the ID of the source. The "To" Attribute refers to the ID of the target.</p> <pre><Transition Name="From "Message Event1" to "Call Activity1" " To="S5" From="S3" Id="T9"></pre>
Default flow	<p>A Transition (same as sequence flow above). The "From" attribute refers to the ID of the source. The "To" Attribute refers to the ID of the target.</p> <pre><Transition Name="From "Message Event1" to "Call Activity1" " To="S5" From="S3" Id="T9"></pre>
Conditional flow	A Transition with a Condition Construct
None start event	<p>An Activity with a StartEvent whose trigger is "None"</p> <pre><Activity> <Event> <StartEvent Trigger="None"/> </Event> </Activity></pre>
Throwing none intermediate event	<p>An Activity that throws an Intermediate Event. The event is IntermediateEvent whose trigger attribute is "None" (a None event)</p> <pre><Activity Name="None Event1" Id="b1bf9ad0-74f2-4fe3-bb55-f35acalbad9"> <Description> </Description> <Event> <IntermediateEvent Trigger="None"/> </Event> ... </Activity></pre>
None end event	<p>An Activity with an EndEvent whose Result attribute value is "None"</p> <pre><Activity> <Event> <EndEvent Result="None"/> </Event> </Activity></pre>
Message start event	<p>An Activity with a StartEvent whose trigger is a message</p> <pre><Activity Name="Message Event1" Id="S19"> ... <Event> <StartEvent Trigger="Message"/> </Event> ... </Activity></pre>
Message end event	<p>An Activity with an EndEvent whose result is a message</p> <pre><Activity Name="Message Event3" Id="S32"> ... <Event> <EndEvent Result="Message"/> </Event> </Activity></pre>

Designer BPMN 2.0 Process XPD L Construct Model Part

Throwing message intermediate event

An Activity with an IntermediateEvent triggered by a Message. The CatchThrow attribute has a value of "THROW" indicating that the message is thrown.

```
<Activities> <Activity Name="Message Throw"
Id="4b64afb5-edf1-4b92-b98c-98d6ca0b3601"> <Description>
</Description> <Event> <IntermediateEvent Trigger="Message">
<TriggerResultMessage CatchThrow="THROW"/>
</IntermediateEvent> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="35.0" Height="35.0"
LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="197.0" XCoordinate="235.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> <Extensions>
<webMethodsDesignerCommonExtensions isTestAfter="false"
loopMaximum="0" retries="0" retryCount="1"
retryInterval="60000" stepLock="false"
logicalServer="Default"> <documentation/> <fontData
fontSize="8" italicFont="false" fontFamily="Tahoma"
boldFont="false" red="0" green="0" blue="0"/>
<ext:wmExpressionFilter/>
</webMethodsDesignerCommonExtensions>
<webMethodsDesignerEventExtensions isStart="false"
correlationServiceName="" correlationFieldName=""
useCorrelation="false" receiveType="EMPTY"
receiveProtocol="SUBSCRIPTION"
isStartStepServiceRequired="false"
externalizeConditions="false" allowsSynchronousReply="false"/>
</Extensions> </Activity> </Activities>
```

Catching message intermediate event

An Activity with an IntermediateEvent triggered by a Message. The CatchThrow attribute has a value of "CATCH" indicating that the message is caught by the Activity.

```
<Activities> <Activity Name="Message Catch"
Id="c2c75b01-cb42-41ef-b79a-3a82b5877120"> <Description>
</Description> <Event> <IntermediateEvent Trigger="Message">
<TriggerResultMessage CatchThrow="CATCH"/>
</IntermediateEvent> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="35.0" Height="35.0"
LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="197.0" XCoordinate="105.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> <Extensions>
<webMethodsDesignerCommonExtensions isTestAfter="false"
loopMaximum="0" retries="0" retryCount="1"
```

Designer BPMN 2.0 Process XPD L Construct Model Part

```

retryInterval="60000" stepLock="false"
logicalServer="Default"> <documentation/> <fontData
fontSize="8" italicFont="false" fontFamily="Tahoma"
boldFont="false" red="0" green="0" blue="0"/>
<ext:wmExpressionFilter/>
</webMethodsDesignerCommonExtensions>
<webMethodsDesignerEventExtensions isStart="false"
correlationServiceName="" correlationFieldName=""
useCorrelation="false" receiveType="EMPTY"
receiveProtocol="SUBSCRIPTION"
isStartStepServiceRequired="true"
externalizeConditions="false" allowsSynchronousReply="false"/>
</Extensions> </Activity> </Activities>

```

Interrupting boundary
message intermediate event

An `IntermediateEvent` attached to an Activity, indicated by the "Target" attribute of the `<IntermediateEvent>`. The event interrupts the flow of the process, so the "Interrupting" attribute has a value of "true". The Activity to which the `<IntermediateEvent>` is attached is indicated by its "Target" attribute.

```

<Activity Name="Attached Intermediate Message Interrupting
event" Id="d802b7ae-9b98-4278-82d7-be220c7ae4c0">
<Description> </Description> <Event> <IntermediateEvent
Interrupting="true"
Target="04975f26-d385-4d54-bb1e-604713235d34"
Trigger="Message"> <TriggerResultMessage CatchThrow="CATCH"/>
</IntermediateEvent> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="35.0" Height="35.0"
LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="193.0" XCoordinate="198.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>

```

Non-interrupting boundary
message intermediate event

An `IntermediateEvent` attached to an Activity, indicated by the "Target" attribute of the `<IntermediateEvent>`. The `IntermediateEvent` is triggered by a message whose "Interrupting" attribute has a value of "false".

```

<Activity Name="Attached Intermediate Message Non interrupting
event" Id="d802b7ae-9b98-4278-82d7-be220c7ae4c0">
<Description> </Description> <Event> <IntermediateEvent
Interrupting="false"
Target="04975f26-d385-4d54-bb1e-604713235d34"
Trigger="Message"> <TriggerResultMessage CatchThrow="CATCH"/>
</IntermediateEvent> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="35.0" Height="35.0"

```

Designer BPMN 2.0 Process XPD L Construct Model Part

```

LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="193.0" XCoordinate="198.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>

```

Boundary interrupting timer intermediate event An Activity with an IntermediateEvent. The "Target" attribute of the IntermediateEvent is the ID of the parent Activity associated with the Timer. The "Interrupting" attribute is "true".

```

<Activity Name="Timer1"
Id="500b242f-ffc8-4c18-bacd-e9948b00c3d8"> <Description>
</Description> <Event> <IntermediateEvent Interrupting="true"
Target="e8065ce7-745e-4fba-9294-ca9522f81776"
Trigger="Timer"/> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="23.0" Height="23.0"
LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="205.0" XCoordinate="222.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>

```

Non-interrupting boundary timer intermediate event An Activity with an IntermediateEvent. The "Target" attribute of the IntermediateEvent is the ID of the parent Activity associated with the Timer. The "Interrupting" attribute is "false".

```

<Activity Name="Timer2"
Id="81969c32-62b4-4d39-9a2b-0c80e595d870"> <Description>
</Description> <Event> <IntermediateEvent Interrupting="false"
Target="a2c162bb-0a9c-425e-9993-b8f890b308ec"
Trigger="Timer"/> </Event> <NodeGraphicsInfos>
<NodeGraphicsInfo Width="23.0" Height="23.0"
LaneId="d27f70dc-c404-48ae-942f-c126ac361427"
PageId="2fdb377f-8e60-4dbd-a906-59693ba3c40cPAGE">
<Coordinates YCoordinate="205.0" XCoordinate="475.0"/>
</NodeGraphicsInfo> </NodeGraphicsInfos> </Activity>

```

Escalation intermediate event not supported in Designer 9.0

Escalation end event not supported in Designer 9.0

Conditional start event not supported in Designer 9.0

Conditional intermediate event not supported in Designer 9.0

Conditional end event not supported in Designer 9.0

Designer BPMN 2.0 Process XPDL Construct Model Part

Interrupting boundary error intermediate event	<p>Activity with an IntermediateEvent whose trigger is an error. The value of the "Target" attribute for the IntermediateEvent matches the associated activity.</p> <pre><Activity Name="Error1" Id="S24"> ... <Event> <IntermediateEvent Target="S18" Trigger="Error"/> </Event> ... </Activity></pre>
Error end event	<p>Activity with an EndEvent whose result is an error.</p> <pre><Activity Name="Error Event1" Id="S31"> ... <Event> <EndEvent Result="Error"/> </Event> ... </Activity></pre>
Interrupting boundary cancel intermediate event	not supported in Designer 9.0
Cancel end event	not supported in Designer 9.0
Interrupting boundary compensation intermediate event	not supported in Designer 9.0
Compensation end event	not supported in Designer 8.2
Signal start event	<p>An Activity with a StartEvent whose trigger is a Signal, as indicated by the value of the "Trigger" attribute.</p> <pre><Activity Name="Signal Event1" Id="80b5ff5b-95d7-4fa0-8654-d28fc5bcba97"> <Description> </Description> <Event> <StartEvent Trigger="Signal"/> </Event> <NodeGraphicsInfos> <NodeGraphicsInfo Width="35.0" Height="35.0" PageId="006f3646-2ad3-49fc-a50c-249fd9b516cbPAGE"> <Coordinates YCoordinate="182.0" XCoordinate="243.0"/> </NodeGraphicsInfo> </NodeGraphicsInfos></pre>
Catching signal intermediate event	<p>An Activity, with an Intermediate event, that catches a signal . The "CatchThrow" attribute has a value of "CATCH" and the value of the Trigger attribute is "Signal")</p> <pre><Activity Name="Signal Event2" Id="be77266a-cf58-43b2-9932-227ded08d8d7"> ... <Event> <IntermediateEvent Trigger="Signal"> <TriggerResultSignal CatchThrow="CATCH"/> </IntermediateEvent> </Event> ... </Activity></pre>
Throwing signal intermediate event	<p>An Activity, with an Intermediate event, that throws a signal . The "CatchThrow" attribute has a value of "THROW" and the value of the Trigger attribute is "Signal")</p>

Designer BPMN 2.0 Process XPD L Construct Model Part

	<pre><Activity Name="Signal Event2" Id="be77266a-cf58-43b2-9932-227ded08d8d7"> ... <Event> <IntermediateEvent Trigger="Signal"> <TriggerResultSignal CatchThrow="THROW"/> </IntermediateEvent> </Event> ... </Activity></pre>
Interrupting boundary signal intermediate event	<p>An Activity with an IntermediateEvent whose trigger is a signal. The value of the "Target" attribute for the IntermediateEvent matches the associated activity (the step on which the event is placed).</p> <pre><Activity Name="Signal1" Id="d80ca1b1-4ddd-400c-9ae3-55b7de4cb645"> ... <Event> <IntermediateEvent Interrupting="true" Target="9048f37a-4134-43e1-8f7d-5dceb27456f6" Trigger="Signal"> </IntermediateEvent> </Event> ... </Activity></pre>
Non-interrupting boundary signal intermediate event	<p>An Activity with an IntermediateEvent whose trigger is a signal. The value of the "Target" attribute for the IntermediateEvent matches the associated activity (the step on which the event is placed).</p> <pre><Activity Name="Signal1" Id="d80ca1b1-4ddd-400c-9ae3-55b7de4cb645"> ... <Event> <IntermediateEvent Interrupting="false" Target="9048f37a-4134-43e1-8f7d-5dceb27456f6" Trigger="Signal"> </IntermediateEvent> </Event> ... </Activity></pre>
Signal end event	<p>An Activity with an EndEvent resulting in a Signal (as indicated by the Result attribute).</p> <pre><Activity Name="Signal Event2" Id="be77266a-cf58-43b2-9932-227ded08d8d7"> ... <Event> <EndEvent Result="Signal"/> </Event> ... </Activity></pre>
Terminate end event	<p>An Activity with an EndEvent whose result attribute is "Terminate"</p> <pre><Activity Name="Terminate1" Id="S30"> <Event> <EndEvent Result="Terminate"/> </Event> ... </Activity></pre> <p>Note: Extensions added to reflect Terminate Status.</p>
Exclusive gateway	<p>An Activity with a Route construct whose GatewayType is "Exclusive"</p> <pre><Activity Name="Gateway1" Id="S55"> ... <Route GatewayType="Exclusive"/> ... </Activity></pre>

Designer BPMN 2.0 Process XPD L Construct Model Part

Inclusive gateway	<p>An Activity with a Route construct whose GatewayType is "Inclusive"</p> <pre><Activity Name="Gateway1" Id="S55"> ... <Route GatewayType="Inclusive"/> ... </Activity></pre>
Parallel gateway	<p>An Activity with a Route construct whose GatewayType is "Parallel"</p> <pre><Activity Name="Gateway1" Id="S55"> ... <Route GatewayType="Parallel"/> ... </Activity></pre>
Complex gateway	<p>An Activity with a Route construct whose GatewayType is "Complex"</p> <pre><Activity Name="Gateway1" Id="S55"> ... <Route GatewayType="Complex"/> ... </Activity></pre> <p>Note: The legacy webMethods gateway is imported as a complex gateway.</p>
Event-based exclusive gateway	not supported in Designer 9.0
Parallel event-based gateway	not supported in Designer 9.0
Message line	<p>An Association whose "Target" and "Source" attributes specify IDs of the source and target</p> <pre><Association Name="From "annotationNode" to "Call Activity1" " Target="S5" Source="N13" Id="T14"> ... </Association></pre>
Annotation	<p>An Artifact of type "Annotation". The value of the TextAnnotation attribute is the annotation text specified.</p> <pre><Artifact TextAnnotation="Used to be known as referenced subprocess" ArtifactType="Annotation" Id="N13"> </Artifact></pre> <p>Note: The sticky note style Note in Designer is mapped the same way as an Annotation in Designer.</p>
Group	not supported in Designer 9.0
Pool	A Pool is mapped to a Pool as well a WorkflowProcess (if you select that option in the import wizard)
Swimlane	A Lane construct within a Pool
Data Input	not supported in Designer 9.0

Designer BPMN 2.0 Process XPD L Construct Model Part

Data Output	not supported in Designer 9.0
Data Collection Object	not supported in Designer 9.0
Data Store	not supported in Designer 9.0
Data Message	not supported in Designer 9.0
SimulationInformation	<p>Attribute of an XPD L Activity which maps to a resource assigned to a process step in a Designer process simulation (.simulation file). Properties and values in the XPD L correspond to resource information in the process. In the Designer process simulation, SimulationInformation maps to a simulation resource for a process step with the naming convention <activity name>_Activity Resource. Simulation step scenarios are created for Designer activity steps (tasks and call activities). SimulationInformation is mapped for these step types only. SimulationInformation properties include:</p> <ul style="list-style-type: none"> ■ Cost: assigned to the Fixed Cost of the resource ■ TimeEstimation: the Duration value is assigned to a Designer simulation scenario, which has a Process Time property. Other TimeEstimation values are ignored, as they do not correspond to existing simulation properties.

Exported joins are mapped as follows in the table below:

Exported Designer Process Model Join Type	XPD L Join Type
AND	PARALLEL
COMPLEX	COMPLEX
OR	OR
Unsynchronized OR	XOR

24 Building and Uploading Processes

■ About Building and Uploading a Process	434
■ Generating a Process	437
■ About Process Generation and Stage Status Display	440
■ About Mapping Services	441
■ Working with Process Versions	442
■ Working with Subscription Filters	444
■ Working with Triggers	446
■ Working with Protocols	448
■ About Enabling Processes	453
■ Setting Default Deployment Values for a Process Model	453

About Building and Uploading a Process

When you build and upload a process, Designer creates the elements that execute at run time based on the information in your process, such as steps, subscription filters, subscription and transition triggers, and transition conditions. Designer then places these generated run-time elements on the Integration Server and uploads information about the process to the Process Audit Database.

Important:

You can build a process and upload it to the Process Audit database for execution only when you are working in Process Developer mode. When you are working in Business Analyst mode, you are limited to uploading a process to the Process Audit Database for analysis only; the process is not built, and it cannot be executed. For more information about modes, see [“About Process Development Modes” on page 52](#).

Each step in a process model is associated with a specific logical server (**Integration Server Name**), defined on the **Implementation** page in the Properties view. Designer places the run-time elements associated with a step on the physical server that is mapped to the logical server for the step.

Every process step generates a type of flow service called a *mapping service*. This mapping service is created on the specified Integration Server, using the signature specified in the inputs and outputs. When you build and upload a process, Designer automatically updates the mapping services of all steps to include the current step signatures, and generates mapping services for all steps that do not yet have them defined. See [“Configuring Build and Upload Preferences” on page 40](#) and [“About Inputs and Outputs” on page 90](#).

When you build a process, Designer opens the Build Report view. The Build Report view displays the progress of the build, as well as any errors or warnings that may apply, including stack trace lines if you select that preference.

Important:

Be sure to keep the following items in mind when you build and upload a process:

- If you change a process-wide property, such as renaming a process or a package, you must regenerate (rebuild) the process.
- In a clustered environment, after a process is built and the package is sent to other servers in the cluster, the servers must be restarted. Alternatively, you can reload the WmPRT package on the servers or run the `pub.prt.admin:scanPackage` service.
- Before using Deployer, be sure to do a full regeneration of your process.

About Building and Uploading with the Command-Line Utility

You can build and upload Designer processes with a command-line utility, available at `Software AG_directory \Designer\util\buildUploadProcess.bat` on Windows systems, and at `Software AG_directory /Designer/util/buildUploadProcess.sh` on UNIX-based systems.

Important:

When you generate a process model with the command-line utility, the associated process model image and icon images *are not generated with the model*. As a result, any webMethods Monitor APIs that use the process or icon images will not perform as expected with the generated process.

When you run the utility, you must specify a configuration file based on the following XML-based configuration file template:

Software AG_directory \Designer\util\processConfig.xml for Windows, and

Software AG_directory /Designer/util/processConfig.xml for UNIX-based systems.

The processConfig.xml file contains all the information the command-line utility needs to build and upload one or more Designer processes. You can use it as a template to create your own configuration files.

The content of the processConfig.xml file is organized as follows in the table below:

Section	Description
<ProcessConfig>	Beginning element of the process configuration file.
<Processes>	Beginning element of the processes section.
<Process>	<p>Defines a single process with elements for the process name, process project name, and process file path.</p> <p>If you have more than one process to build and upload, you must create consecutive <Process> sections within the <Processes> section, one for each process.</p> <p>One <Process> section is included in the configuration file by default. Copy and paste if you need more than one.</p> <p>Important: Be sure to remove any unused sections before you run the utility.</p>
< Integration Server >	<p>Defines the Integration Server with elements for name, host, port, user ID, password, and SSL use.</p> <p>If a process requires more than one Integration Server, you must create consecutive <Integration Server> sections within the <Process> section, one for each server.</p> <p>Two <Integration Server> sections are included in the configuration file by default. Copy and paste if you need more.</p> <p>Important: Be sure to remove any unused <Integration Server> sections before you run the utility.</p>
<logDir>	<p>Log directory file path.</p> <p>Generated build reports are placed here. File names are appended with a time stamp in the format yyyyMMdHHmmss (for example, 20090503041804 would be 4:18 AM and 4 seconds on May 3, 2009).</p>

Building and Uploading Processes with the Command-Line Utility

➤ To build and upload processes from the command-line

1. Open the directory *Software AG_directory* \Designer\util\ on Windows systems or *Software AG_directory* /Designer/util/ on UNIX-based systems.
2. Open the processConfig.xml file in a text editor. For more information about the contents of this file, see [“About Building and Uploading with the Command-Line Utility”](#) on page 434.
3. Save the file with a new filename, so that you do not overwrite the example file processConfig.xml with your modifications.
4. If you have more than one process to build and upload, copy the <Process> element (from <Process> to </Process>) and paste it after </Process> and before </Processes>.
5. For each process you want to build and upload, populate the ProcessName, ProcessProjectName, and Process file path information in the <Process> section.
6. Add or remove <Integration Server> elements (from <Integration Server> to </Integration Server>) as required within each <Process> section.
7. In each <Integration Server> element, specify the Name, Host, Port, UserID, Pwd, and UseSSL information.
8. Enter the log directory file path in the <logDir> section.

Important:

Be sure to remove any unused sections before you save the configuration file and run the utility.

9. Save the configuration file
10. Open a command session and run *Software AG_directory* \Designer\util\buildUploadProcess.bat on Windows systems or *Software AG_directory* /Designer/util/buildUploadProcess.sh on UNIX-based systems as follows:

On Windows:

```
buildUploadProcess.bat -configFile configFileName
```

In UNIX-based systems:

```
buildUploadProcess.sh -configFile configFileName
```

Important:

The command-line utility does not deploy webMethods CAF task applications associated with task steps in a process model to My webMethods Server. However, it does create Integration Server (IS) services for task steps. You must deploy the task applications manually, see *webMethods BPM Task Development Help*.

Generating a Process

One trigger is generated for each **Integration Server Name** (logical server) defined in a process. If you want to use a different trigger for a given step, you must create that step on a different logical server (with a different **Integration Server Name**).

Generating Process Version 1

In this table example, a Designer process named ProcessName has four steps.

Step Type	Step Label (Name)
Message Start Event (receive)	CatchDocument
Service Task	InvokeService
User Task	DoSomething
Message End Event (publish)	ThrowDocument

The process uses two logical servers (Integration Server Names), both listed in the following table:

the default logical server	Default
a second logical server	LogicalServer2

The first time the process is built (version 1), the package is generated. Its name is the same as the ProcessProjectName. Then a new process version is created (version 2), and the process is regenerated (built again).

Within the generated package, Designer creates a folder named ProcessName, which contains folders for each process version (ProcessName_1 and ProcessName_2), as well as the default logical server folder (Default) and the subscription trigger (subscriptionTrigger) used for all versions of the process. Designer creates the ProcessName_1 folder the first time the process is built. It contains the Integration Server Name folder (Default), which contains the following:

The table below lists the content of the Integration Server Name folder:

A flow service for the Message Start Event step.	CatchDocument
A flow service for the Service Task step.	InvokeService
A flow service for the User Task step.	DoSomething

A transition trigger for version 1 of the process on this transitionTrigger logical server (Default).

The process also uses a second logical server called LogicalServer2, so Designer creates a folder for it under the ProcessName_1 folder. The Message End Event step (ThrowDocument) runs on LogicalServer2, and Designer generates the information listed in the following table in the LogicalServer2 folder:

A flow service for the Message End Event step. ThrowDocument

A transition trigger for version 1 of the process on transitionTrigger
this logical server (LogicalServer2).

Generating Process Version 2

When Designer generates version 2 of the process, the package (ProcessProjectName) containing the logical server folder (Default) and the subscription trigger (subscriptionTrigger) used for all versions of the process already exists, because these components were used in version 1.

Designer creates the ProcessName_2 folder for the second version of the process using the ProcessName_1 folder as a template. Designer creates a new folder for the first logical server for the new process version (Default) under the ProcessName_2 folder, and then copies all the transition triggers and generated flow services that version 1 of the process used, making adjustments for any changes that were made in version 2.

It follows the same procedure for creating the second logical server folder, LogicalServer2, and copies the transition trigger and generated flow services from version 1 of the process, again making adjustments for any changes that made in version 2.

About Receive Steps

A receive step receives information from outside of a process and starts a new process instance or joins an existing process instance based on that information. A receive step can be any of the BPMN constructs listed in the following table:

Receive Step Type	Step Description
Receive Task	“About Receive Tasks” on page 237
None Start Event	“About None Start Events” on page 268
Message Start Event	“About Message Start Events” on page 269
Signal Start Event	“About Signal Start Events” on page 272
Message Intermediate Event	“About Message Intermediate Events” on page 278
Signal Intermediate Event	“About Signal Intermediate Events” on page 283

Each specific step configuration is governed by the protocol used. Each protocol has its own configuration settings. For more information, see [“About Subscription Triggers” on page 446](#) and [“Working with Protocols” on page 448](#).

About Subscription Triggered Process Generation

A subscription triggered process is triggered by a subscription to a webMethods Broker (deprecated) or Universal Messaging document type.

About Simple Service Triggered Process Generation

Simple Service triggered processes do not actually have triggers. They use a simple service to pass data between two steps, the receive task and the send task. No trigger is generated.

About JMS-Triggered Process Generation

When Designer generates a process that uses a JMS trigger, the current protocol configuration specified in the Properties view is applied. This means that if you have modified the trigger in Integration Server and you subsequently change the JMS Properties on the **Implementation** page in the Properties view (**Connection Alias** and **Destination Name**), your previous trigger edits for these fields are overwritten. This is true for initial generation and subsequent regenerations.

Note:

Only the above mentioned JMS properties are overwritten. Other JMS trigger properties modified in the trigger are retained.

Note:

To be able to build and upload a JMS-triggered process, you must have Software AG Universal Messaging installed, running with an appropriate connection factory, and the associated triggers enabled. For information about configuring Software AG Universal Messaging, see the PDF publication *Administering webMethods Process Engine*.

Note:

When a JMS-triggered process uses the PE_NONTRANSACTIONAL_ALIAS connection alias, that alias must have the **Manage Destinations** option shown as **Yes** in the connection alias Advanced Settings. For more information, see [“Setting the Manage Destination Option for a Connection Alias” on page 451](#).

About EDA (Deprecated) Triggered Process Generation

Generating a process with the EDA (For EDA Event Triggered Processes) protocol is very similar to generating a JMS triggered process. By default, the EDA (deprecated) protocol uses the EventBus Connection Alias.

When Designer generates a process that uses an EDA (deprecated) trigger, the current configuration specified in the Properties view is used. This means that if you have edited the trigger and subsequently edit the JMS Properties on the **Implementation** page in the Properties view

(connection alias and destination name), your previous trigger edits for these fields are overwritten. This is true for initial generation and subsequent regenerations.

Note:

Only the above mentioned protocol properties are overwritten. Other JMS properties edited in the trigger are retained.

About Process Generation and Stage Status Display

Designer enables you to define stages within a process to more closely track progress in specific portions of a running process. For more information, see [“Stages and Milestones” on page 351](#). When you generate a process model that contains stages, Designer writes stage settings in the process model to the Process Audit database.

These stage settings are extracted from the database by webMethods Process Tracker which displays stage status information in webMethods Monitor. Process Tracker retrieves this stage information from the database every five minutes.

Because of this five minute retrieval interval, stage settings for a regenerated process will not be picked up by Process Tracker until the next retrieval. Therefore, if a regenerated process is executed prior to the database retrieval, *the stage instance data/status displayed in Monitor will not match the settings in the uploaded model*.

As a result, you have the following choices to ensure the correct stage information is displayed in Monitor:

- Wait at least five minutes before you execute a newly regenerated process model that contains stages.
- Wait at least five minutes and then click the **Refresh** button on the Process Instance Detail page in Monitor.

You can configure the retrieval interval for Process Tracker. For more information, see [“Configuring the Process Tracker Retrieval Interval” on page 440](#).

For more information about viewing stage data in Monitor, see the PDF publication *webMethods Monitor User’s Guide*.

Configuring the Process Tracker Retrieval Interval

➤ To configure the Process Tracker retrieval interval

1. Open this file in a text editor:

```
Software AG_directory \optimize\analysis\conf\ProcessTracking.xml
```

2. Locate the following section:

```
<entry key="modelRefreshInterval" type="Long">  
  <rendering-properties>
```

```

        <property name="type" value="textfield"/>
        <property name="size" value="25"/>
        <property name="hidden" value="true"/>
    </rendering-properties>
    <value>300000</value>
</entry>

```

- The retrieval interval is defined in the `<value>` element, with a default of 300000 milliseconds (five minutes). Type a new value in milliseconds, using a value of not less than 30000 (30 seconds).

Important:

Specifying a value of less than 30 seconds can cause serious disruption to normal operations of Process Tracker and the Analytic Engine.

- Save the file.
- Stop and restart the Analytic Engine to apply the change.

About Mapping Services

Designer generates a type of flow service called a *mapping service* on its specified logical server (**Integration Server Name**), using the signature specified in the step inputs and outputs. When you build and upload a process, Designer automatically updates the mapping services of all steps to include the current step signatures, and generates mapping services for all steps that do not yet have them defined. See [“Configuring Build and Upload Preferences” on page 40](#) and [“About Inputs and Outputs” on page 90](#).

Important:

You must be connected to the Process Audit Database to update process data mapping services during generation.

Designer names mapping services using the conventions are listed in the following table:

Step Type	Flow Service
Tasks (except Manual Tasks)	<code><step label></code>
Manual Tasks (OLD STYLE)	no wrapper flow service <code><step label>_START</code>
webMethods Subprocess (Deprecated) (NEW STYLE)	<code><step label>_END</code> <code>PRECALL_<step label></code>
BPMN Subprocess (OLD STYLE)	<code>POSTCALL_<step label></code> <code>PRESUB_<step label></code>

Step Type	Flow Service
webMethods Referenced Process (Deprecated)	POSTSUB_<step label>
(NEW STYLE)	PRECALL_<step label>
BPMN Call Activity	POSTCALL_<step label>
Start Events and Catching Events (except None Events)	<step label>
None Events	no wrapper flow service
Boundary Events (Interrupting and Non-Interrupting)	no wrapper flow service
Throwing Events	no wrapper flow service
End Events	no wrapper flow service

Note: Designer supports non-unique and empty step labels. When creating mapping services for these steps, Designer uses the <step ID> in place of the <step label>.

By default, Designer does not display generated flow services in the Package Navigator view. For more information, see [“Displaying Generated Flow Services in the Package Navigator View” on page 31](#).

Important: Designer does not validate the existence of document fields referenced by a process during generation. If a document is edited to remove a field that is used in a subscription filter or a transition condition, generation will still proceed without error. However, this may cause the process to fail to run (in the case of a subscription filter), or to follow an incorrect path (in the case of a transition condition).

Working with Process Versions

Each Designer process has a **Version**, displayed in the Properties view of the process. Upon initial creation, the process version is 1. You can create a new process version.

Important:

If you change a process-wide property, such as renaming a process or a package, Designer prompts you to regenerate the process.

Tip:

The **Version** field must not exceed 50 characters.

When you create a new version of a process, you can then upload it to the Process Audit Database for analysis or build it and upload it to Integration Server for execution.

The following table lists the icons with their meaning.



Upload for Analysis Only



Build and upload for execution

Note:

You cannot change the subscription filter from one process version to another. Doing so will prevent correlation with any existing (already running) instances, resulting in a "No trigger available for incoming document" error. In addition, you cannot modify the IS document type between versions.

Both uploads (for analysis, and for execution) insert information about the process into the Process Audit metadata tables, for viewing in My webMethods Server.

The following table lists the Process Audit table along with the Process Information Uploaded.

Process Audit Table	Process Information Uploaded
WMPROCESSDEFINITION	Process name, ID, description, version QOS settings, type (BAM or BPM), and enablement status
WMSTEPDEFINITION	Step labels, IDs, coordinates
WMPROCESSIMAGE	The image for display in My webMethods Server
WMSTEPTRANSITIONDEFINITION	Name and coordinates of transitions
WMCUSTOMFIELDDEFINITION	Custom logged fields

Also, for BPM-executable processes, Designer uploads information about what was last built (generated) to the WMGENERATIONRECEIPT table. This information is used by webMethods Deployer for use in moving a process from one environment to another.

Important:

You cannot delete a process model on My webMethods Server if an instance of that process has ever been executed. For more information, see *webMethods Monitor User's Guide*.

Creating a New Process Version

You can create a new process version by editing the version information.

➤ To create a new process version

1. Click the process editor canvas to select the process.
2. On the **General** page in the Properties view, edit the **Version** field. Alphanumeric characters and spaces are allowed.

Note:Version is a required field. If you delete the version, Designer will assign the default version of 1.

3. Select one of the listed in the following table:



Upload for Analysis Only



Build and upload for execution

Important:

After you have built and uploaded a process for execution, you cannot upload the same version of that process for analysis only. If you want to upload the process for analysis only, you must create a new process version.

About Releasing New Process Versions

After you build and upload the new version, you can update any or all of the currently running instances of that model so that they start using the newer version (assuming it is enabled). For more information, see these topics:

- [“Working with Process Versions” on page 442.](#)
- [“Updating a Process Instance to a New Model Version”](#) in the PDF publication *webMethods Monitor User’s Guide*.
- [“Enabling and Disabling Process Model Versions”](#) in the PDF publication *webMethods Monitor User’s Guide*.

Refer to the PDF publication *webMethods Monitor User’s Guide* for more information about working with running processes in My webMethods, including modifying Quality of Service settings at run time, and managing multiple process versions.

Working with Subscription Filters

A subscription filter enables you to define the instances of a document that can trigger or join a process, based on values in specific fields in the subscription document. Subscription filtering occurs after a subscription document arrives.

You can define a subscription filter for any receive step, with two exceptions:

1. Receive tasks that use the Simple Service (For Synchronous Receive/Send) protocol do not have subscription filters. This protocol represents two-way messaging with specified sender and receiver, so filtering is not necessary. For more information about receive tasks, see [“About Receive Tasks” on page 237.](#)
2. Message start and message intermediate events that use the EDA (For EDA Event Triggered Processes) protocol do not have subscription filters. The filtering for EDA (deprecated) events occurs before the events reach the step. For more information, see [“About Message Start Events” on page 269](#) and [“About Message Start Events” on page 269.](#)

Adding a Subscription Filter

> To add a subscription filter

1. In the process editor, select a step with a subscription to a document.
2. Click the **Implementation** page in the Properties view.
3. In the **Subscription Filters** section, click  **Add** to create a subscription filter.
4. Configure the conditions as described in [“Configuring a Subscription Filter” on page 445](#).
5. Save the process model.

Configuring a Subscription Filter

> To configure a subscription filter

1. In the process editor, select a step with a subscription filter.
2. Click the **Implementation** page in the Properties view.
3. In the **Subscription Filters** section, use the condition editor to specify the fields you want to use in the filter. To begin, click  **Add** to add information that defines the subscription filter. You can add multiple condition lines by specifying the AND/OR operator and clicking **Add** again. Then, define the values listed in the following table:

Field	Description
Field Name	Click this table cell and select an available document field from the list. The subscription filter is based on the value of this field.
Operator	Click this table cell and select an operator to use to compare the Field Name value and the Comparison Value/Field value. See “About Expression Operators” on page 173 for more information.
Comparison Field/Value	Click this table cell and select an available document from the list and select a comparison field in that document, or type a value that you want to compare with the value selected for Field Name .

Note:

Field	Description
	Trading Networks Partner Profiles are not selectable in subscription filters. Type the Corporation Name, predefined in the Partner Profile.
AND/OR	Use the AND and OR operators to define the logical behavior if you specify multiple parameters for the subscription filter.

4. Save the process model.

Removing a Subscription Filter

> To remove a subscription filter

1. In the process editor, select a step with a subscription filter.
2. Click the **Implementation** page in the Properties view.
3. In the Subscription Filters section, select a subscription filter, and then click  **Remove**.
4. Save the process model.

Working with Triggers

About Subscription Triggers

A subscription trigger starts a new process instance or joins an existing process instance at a particular step. You configure an appropriate receive step to receive a message or a signal from outside the process. This information in this message or signal is transmitted to the receive step using the messaging protocol specified for that information type. The receive step then uses this information to either trigger a new process instance or join an existing process instance.

Subscription triggers are generated using the data that comes into the step and the protocol specified to transmit that information. Several different types of receive steps can use subscription triggers to start a process or join an already started process.

The following table lists the top-level process steps and the available messaging protocols.

Top-Level Process Steps	Available Messaging Protocols
Message start event (receive a message and start a process)	Subscription (For Publishable Documents) JMS (For JMS Triggered Processes)

Top-Level Process Steps	Available Messaging Protocols
Message intermediate event (receive a message and join a process)	EDA (For EDA Event Triggered Processes)
Signal start event (observe a signal and start a process)	Subscription (For Publishable Documents) JMS (For JMS Triggered Processes)
Signal intermediate event (observe a signal and join a process)	
Receive task	Subscription (For Publishable Documents)
<ul style="list-style-type: none"> ■ Receive a message and start a process ■ Receive a message and join a process 	JMS (For JMS Triggered Processes) Simple Service (For Synchronous Receive/Send)
None start event (receive a message and start a BPMN callable process from a parent process)	Subscription (For Publishable Documents) JMS (For JMS Triggered Processes) EDA (For EDA Event Triggered Processes)
	<p>Important: A BPMN callable process <i>requires a none start event to start</i>. The none start event can exist in addition to another start step (such as a message start step) that applies to the process when it is running as other than a callable process.</p>

About Transition Triggers

A transition is a progression from step to step in a process. A transition trigger is the mechanism by which a step receives, manipulates, and sends information as part of the defined process structure.

Invoking Child Processes

You can use a call activity to invoke a:

- webMethods referenced process (deprecated)
- BPMN callable process

Invoking a webMethods referenced process is identical to invoking a top-level process. No additional configuration is required. Invoking a BPMN callable process requires a none start event in the child process, as well as a global process specification (process inputs and outputs) for the child process.

Important:

If a callable process also contains a message or signal start event with a subscription filter, that start event is executed only when the callable process is *not* behaving as a child process.

Note:

When a callable process contains both a none start event and a message start event that is configured to use an EDA (deprecated) event type, the message start step and the none start step of the child process must be on different Integration Servers (that is, the **Integration Server Name** property must be different for each of the two steps). Otherwise a validation error occurs during the build and upload procedure.

Working with Protocols

Process steps use messaging protocols to send and receive information. Messaging protocols are applied when a subscription trigger starts or joins a process, and when information is sent and received within the process, such as when catching and throwing BPMN events.

Note:

Parent and child process models must use the same protocol.

Designer supports the following four protocols listed in the table below, although some protocols are not available for specific activity or event types:

Protocol	Description	Used by
Subscription	For Publishable Documents	Used by send and receive task activities, message events, signal events, and the start none event. See “Configuring a Subscription-Triggered Process” on page 449 and “About Subscription Triggered Process Generation” on page 439.
Simple Service	For Synchronous Receive and Send	Used by send and receive task activities for two-way synchronous information exchange. See “Configuring a Simple Service Triggered Process” on page 450 and “About Simple Service Triggered Process Generation” on page 439.
JMS	For JMS Triggered Processes	Used by send and receive task activities, message events, signal events, and the start none event. See “Configuring a JMS-Triggered Process” on page 450

Protocol	Description	Used by
		and “About JMS-Triggered Process Generation” on page 439.
EDA (deprecated)	For EDA Event Triggered Processes	Used by message events. See “Configuring an EDA-Triggered Process” on page 452 and “About EDA (Deprecated) Triggered Process Generation” on page 439.

Message end events only send information, so they use only the send version of the selected protocol. Message start events only receive information, so they only use the receive version of the selected protocol.

Software AG Designer supports the use of the EDA (deprecated) protocol with BPMN message events as a BPMN extension.

Configuring a Subscription-Triggered Process

Note:

When you build and upload a subscription triggered process, Process Engine uses the default Integration Server connection alias, based on the message provider configured in Integration Server. The default connection alias for Broker (deprecated) is IS_BROKER_CONNECTION. The default connection alias for Universal Messaging is IS_UM_CONNECTION. When you change the default connection alias, you must reload the WmPRT package in Integration Server Administrator.

» To configure a subscription-triggered process

1. On the **Implementation** page in the Properties view of the receive step, select **Subscription (For Publishable Documents)** as the **Receive Protocol**.
2. Select the **Receive Document**.
3. Create a **Subscription Filter** to further limit the instances of the **Receive Document** that will trigger the receive step.
4. Save the process model.

Setting the Encoding Type for the Universal Messaging Document Types

When you use the Universal Messaging protocol as the Receive Protocol for a subscription triggered process, you must set the encoding type for the Universal Messaging documents in the process model. By default, the encoding type for process model documents is set to IData, but Universal Messaging documents require ProtoBuf encoding.

➤ **To set the encoding type for Universal Messaging document types:**

1. In Designer, navigate to the document type for which you want to set the encoding type.
2. Lock the document type for editing.
3. On the Property tab, expand the **webMethods Messaging** section.
4. In **Encoding type**, select **Protocol buffers**.

After you set the encoding type for a Universal Messaging document type, synchronize the Universal Messaging documents with the messaging provider.

Configuring a Simple Service Triggered Process

The Simple Service (For Synchronous Receive/Send) protocol is used by receive and send tasks for two-way synchronous information exchange. The receive task, which can start or join a process, has a corresponding send task, which itself is configured to correlate with the receive task.

Receive tasks that use the Simple Service (For Synchronous Receive/Send) protocol do not have subscription filters. This protocol represents two-way messaging with specified sender and receiver, so filtering is not necessary. For more information about receive tasks, see [“About Receive Tasks” on page 237](#).

Configuring a JMS-Triggered Process

Designer enables you to configure JMS-specific properties in the Properties view, so that you do not have to manually modify the generated JMS trigger after generation, and so that subsequent generations will not be in conflict with manual modification. Regeneration always uses current values in the process.

Note:

You must have an active connection to Integration Server to specify a connection alias and destination name as described in this procedure.

Note:

To be able to build and upload a JMS triggered process, you must have Software AG Universal Messaging installed.

➤ **To configure a JMS-triggered process**

1. On the **Implementation** page in the Properties view of the receive step, select **JMS (For JMS Triggered Processes)** as the **Receive Protocol**.
2. Select the **Receive Document**.

3. In the Protocol Properties section, do the following:
 - Accept the default **Connection Alias**, or select a different one with the **Browse** button. The connection alias must exist and be enabled on the Integration Server used by the receive step.
 - Click the **Browse** button to select a **Destination Name** (required). The **Destination Type** is displayed when you hover your mouse over a specified **Destination Name** field.

Note:

The destination name for the start none event is a pre-configured internal value and cannot be viewed or modified.

4. Create a **Subscription Filter** to further limit the instances of the **Receive Document** that will trigger the receive step.
5. Save the process model.

Setting the Manage Destination Option for a Connection Alias

When a JMS-triggered process uses the PE_NONTRANSACTIONAL_ALIAS connection alias, that alias must have the **Manage Destinations** option shown as **Yes** in the connection alias Advanced Settings. Otherwise, the following errors occur during process model generation:

- Error: Failed while checking database for prior build and upload of process
- [ISS.0134.9069] Unsupported action

➤ To set the Manage Destination option for a connection alias

1. Open the Integration Server Administrator for the targeted Integration Server.
2. Click **Settings > Messaging > JMS Settings**.
3. If the PE_TRANSACTIONAL_ALIAS connection alias is enabled, disable it.
4. Click the PE_TRANSACTIONAL_ALIAS link.
5. Click **Edit JMS Connection Alias** and do either of the following:
 - Select the **Manage Destination** check box to enable the option (displayed as **Yes**).
 - Clear the **Manage Destination** check box to disable the option (displayed as **No**).
6. Click **Save Changes**.
7. Re-enable the PE_TRANSACTIONAL_ALIAS connection alias.

Configuring an EDA-Triggered Process

The EDA (For EDA Event Triggered Processes) protocol is built on the JMS protocol. Configuration is very similar to a JMS triggered process. The Connection Alias options are filtered for an EDA (deprecated) triggered process so that only JNDI-compliant entries are displayed.

Note:

If you are working in an environment where the targeted Integration Server nodes are in cluster or distributed mode, you must:

- Ensure that all of the Integration Server nodes are connected to the same webMethods Broker (deprecated) instance.
- Add the current Broker (deprecated) information to the file `com.softwareag.eda.nerv.properties` found on node 2 in: `\profiles\IS\configuration\com.softwareag.platform.config.propsloader`.

Message start and message intermediate events that use the EDA (For EDA Event Triggered Processes) protocol do not have subscription filters. The filtering for EDA (deprecated) events occurs before the events reach the step. For more information, see [“About Message Start Events” on page 269](#) and [“About Message Start Events” on page 269](#).

➤ To configure an EDA-triggered process

1. On the **Implementation** page in the Properties view of the receive step, select **EDA (For EDA Event Triggered Processes)** as the **Receive Protocol**.
2. Select the **Receive Document**.
3. In the Protocol Properties section, do the following:
 - Accept the default **Connection Alias**, or select a different one with the **Browse** button. The connection alias must exist and be enabled on the Integration Server used by the receive step.
 - Click the **Browse** button to select a **Destination Name** (required). The **Destination Type** is displayed when you hover your mouse over a specified **Destination Name** field.

Note:

The destination name for the start none event is a pre-configured internal value and cannot be viewed or modified.

4. Click the **Browse** button to select a custom **Destination Name** instead of using the default trigger names created at generation. The **Destination Type** is displayed when you hover your mouse over a specified **Destination Name** field.
5. Save the process model.

About Enabling Processes

Processes that you build and upload for execution are not automatically enabled. You can enable them in webMethods Monitor. See the PDF publication *webMethods Monitor User's Guide* for more information about enabling and disabling processes.

In addition, you can set a preference in Software AG Designer that specifies the default process-enablement behavior for all process models when they are built and uploaded. You can choose to always enable your processes, never enable them, or to be prompted to choose every time. For more information, see [“Configuring Build and Upload Preferences” on page 40](#).

Setting Default Deployment Values for a Process Model

Use these values to specify the default settings for a process model when you deploy it with webMethods Deployer. At deployment time, the interface enables you to specify the values for various properties in the target environment.

Important:

Default Deployment Values are not related to building (generating) and uploading processes in Designer. When you move a model from one environment to another, during deployment these values are implemented in the deployed model. *This applies only to repository-based deployment.* For runtime deployment, the My webMethods Server and the Deployer resubmission values for enable/disable are used instead.

See [“Configuring Build and Upload Preferences” on page 40](#) for information about setting default preferences for building and uploading.

The values you specify here appear as default values in the Deployer interface. You can modify these values at deployment time; otherwise, these settings are applied to the process model in the target environment.

➤ To set default deployment values for a process model

1. Open the process model you want to work with in the process editor.
2. Click anywhere on the process canvas to put focus on the process. In the Properties view, click the **Default Deployment Values** tab.
3. Specify the settings from the following table as desired:

Setting	Description
Process will be execution-enabled by default during deployment by Deployer	Select this check box to automatically enable the process for execution when deployed by Deployer.

Setting	Description
Process will be analysis-enabled by default during deployment by Deployer	Select this check box to automatically enable the process for tracking and analysis when deployed by Deployer.
Steps Enabled for Resubmission	Select the check box for each step that you want to enable for resubmission when deployed by Deployer.

4. Save the process.

25 Debugging Processes

■ About Process Debugging	456
■ Creating a Process Debug Configuration	459
■ Debugging a Process	460
■ Modifying Input Data	470
■ Working with Breakpoints	472

About Process Debugging

When you debug a process model, you can inspect the way it behaves with real data, and see how that data behaves as it travels through the process. You can control your navigation through process steps, and see the progress of those steps, including errors. You can choose to debug a process in two ways:

- *step through* the process, using navigation buttons to control it step by step, or
- *run* the process from start to finish.

Based on the process design and the input data, Designer identifies the next *eligible step* or steps to run. When there are multiple eligible steps, you can select the one to execute next. You can also take the *default step*, which is selected by the debugger and outlined in the process editor.

You control the debugging session with the buttons on the Trace view toolbar. Step navigation buttons allow you to select the way in which you want to interact with steps: step in, step over, and step out. For more information, see [“Navigating Steps during Debugging” on page 462](#) and [“About the Trace View” on page 463](#).

The process editor displays debug status symbols that reflect this progress, and the views in the Process Debug perspective provide additional supporting functionality, as described in the following table:

Process Debugging View	Description
 Trace view	The Trace view displays details of each step in a debugging session in a tabular form and contains a toolbar you use to control the session in the active process editor tab. You can select a row that corresponds to a step in the Trace view to inspect the corresponding pipeline data output in the Pipeline Data view. You can also view and capture exception data. See “About the Trace View” on page 463 .
 Pipeline Data view	The Pipeline Data view displays the pipeline data associated with the output of the process step selected in the Trace view. You can expand the information to see more details of the data from the pipeline for the selected step, enabling you to troubleshoot the process at the step level. You can also copy the pipeline of one or more steps. See “About the Pipeline Data View” on page 471 .
 Breakpoints view	The Breakpoints view displays all breakpoints in your workspace. It also allows you to manipulate breakpoints. See “About the Breakpoints View” on page 473 .

The process debugger requires that your process start with a minimum of one receive step with an Integration Server document type input. Multiple receive steps, using correlation to join the process, and multiple input documents are also supported. For more information, refer to [“Debugging an Intermediate Receive Step” on page 465](#).

Note:

When you debug a process whose receive step trigger uses the JMS or EDA (deprecated) protocol, Designer checks the Integration Server used by the receive step to ensure the connection alias exists and is enabled, before attempting to debug the process.

Before you can debug a process, you must build and upload it to the Process Audit Database for execution. For more information, see [“About Building and Uploading a Process” on page 434](#).

The process debugger requires an enabled process. Designer does not automatically enable uploaded processes, but you can set a Build and Upload preference to automatically enable a process. For more information, see [“About Enabling Processes” on page 453](#). You can also set a Process Debug preference to automatically enable a process being debugged, and optionally to automatically disable it again when you have finished debugging. For more information, see [“Configuring Process Debug Preferences” on page 458](#) and [“Configuring Build and Upload Preferences” on page 40](#).

When you start debugging a process, Designer can prompt you to enter data input for the receive step or steps. You can save entered data in XML form for reuse, save different versions of the data, and load the XML from a file in subsequent sessions.

Designer applies symbols to the process in the editor to provide visual indicators of debugging progress. Steps are marked as follows in the table below:

Symbol	Step status
	Completed
	Eligible
	<p>All eligible steps are marked with this symbol. You can select the next step to run when you step through the process. This includes the default step, which is outlined in the editor and also in the list of eligible steps. Alternatively, you can run all the steps without making any choices.</p> <p>Timer boundary events, both interrupting and non-interrupting, support breakpoints and can be eligible steps.</p>
	<p>Note: Other boundary events (message, error, signal) do not support breakpoints, and cannot be eligible steps. If you step over a message, error, or signal event, debugger stops on the step after it.</p>
	For more information, see “Navigating Steps during Debugging” on page 462 .
	Running
	Waiting
	Interrupted

Symbol	Step status
	Failed or canceled

Configuring Process Debug Preferences

> To configure Process Debug preferences

1. In Designer: **Window > Preferences > Software AG > Process Development > Process Debug**
2. You can configure the Process Debug preferences as follows in the table below:

Preference	Description
Automatically expand the pipeline	<p>When you click a row in the Trace view, the Pipeline Data view displays the output data for the selected step. You can set the Pipeline Data view's display to automatically expand or not, or to prompt you to choose each time.</p> <p>Choose Always, Never, or Prompt. Default is Always.</p>
Automatically enable process when debugging	<p>Check this box to automatically enable processes when debugging them. If this box is not checked, you must enable processes manually in My webMethods Server in order to debug them, or use the Designer Build & Upload preference to enable the process when it is built and uploaded. See “Configuring Build and Upload Preferences” on page 40.</p>
Restore previous process state after debugging	<p>This option is only available when Automatically enable process when debugging is enabled. Check this box to disable processes that were automatically enabled for debugging after you have finished debugging them.</p>
Peak Memory Use (percent)	<p>Enter the maximum percentage of Designer memory a Process Debugging session will be allowed to consume. The default value is 100.</p> <p>After the threshold is reached, Designer prompts you to decide whether to end the debugger session. If you choose to end the session, the Process Debugger adds a row to the Trace view. If you choose not to end the session, the Process Debugging session will continue to prompt you as long as Designer's memory consumption remains at the threshold set for the Peak Memory Use (percent) preference.</p> <p>You can change the Peak Memory Use (percent) threshold at any time during a debugging session. You might choose to do this to</p>

Preference	Description
	prevent the prompt from recurring if you decide to continue a session that consumes a higher percentage of Designer memory than currently set in Preferences.

3. To apply default preferences on a page, click **Restore Defaults**.
4. Do one of the following:
 - Click **Apply** to save your changes and keep the Preferences window open.
 - Click **OK** to save your changes and close the Preferences window.
 - Click **Cancel** to close the Preferences window without saving your changes.

Creating a Process Debug Configuration

Before you can debug a process, you must create a Process Debug Configuration for it. After you have defined a configuration, you can start a debug session based on it. You can start a debug session from the Debug Configuration dialog box, or by accessing an existing configuration from the Debug menu.

➤ To create a Process Debug Configuration

1. Click **Run > Debug Configurations** to open the Debug Configurations dialog box.

Tip:

You can press the F11 key on your keyboard to start a debugging session. For more information about keyboard shortcuts, see [“Using Keyboard Shortcuts” on page 85](#).

2. Click to select **Process Debug**  in the tree.
3. Click to select an existing Debug Configuration, or click **New launch configuration**  to create a new one.

Tip:

To quickly access existing Debug Configurations, click the menu (▾) button next to the **Debug** button () on the main toolbar. Designer displays a list of recently used configurations from which you can select.

4. On the Process tab, select the **Project** and **Process** you want to debug.
5. Click the **Process Inputs** tab to provide optional default input data for any associated Integration Server documents. For more information, see [“Loading Input Data” on page 470](#).

6. Select the **Prompt for data at launch** check box to be prompted for input data each time you use this configuration.

Note:

This setting applies only to the current Debug Configuration.

7. Do one of the following:
 - Click **Apply** to save the Debug Configuration.
 - Click **Debug** to save the Debug Configuration and start debugging the specified process with it.

Debugging a Process

You can start a debug session from the Debug Configuration dialog box, or by accessing an existing Debug Configuration from the Debug menu. For more information, see [“Creating a Process Debug Configuration” on page 459](#).

When you start debugging a process, Designer can prompt you to enter data input for the receive step or steps. You can save entered data in XML form for reuse, save different versions of the data, and load the XML from a file in subsequent sessions.

The process debugger adds step status symbols in the process editor during the debug session. It also adds bold transition lines to indicate paths taken during debugging.

> To debug a process

1. Click the menu (·) button next to the **Debug** button (🔍) on the main toolbar to open the Debug Configurations dialog box.
2. Select an existing configuration or create a new one as described in [“Creating a Process Debug Configuration” on page 459](#).

Tip:

You can use the F11 key on your keyboard to start a debugging session. For more key combinations, see [“Using Keyboard Shortcuts” on page 85](#).

3. If you selected **Prompt for data at launch** in the Debug Configuration, the default input data is displayed in the Enter input for *document name* dialog box.
4. If required, update the input data provided to the process you are debugging. You can enter data directly in the dialog box or load it from a file, as described in [“Loading Input Data” on page 470](#). For more information about modifying input data, refer to [“Modifying Input Data” on page 470](#).
5. Click **OK** to submit the data and start debugging.

6. When debugging starts, Designer displays a message on the first line of the table in the Trace view:

The following table lists the message along with the icon.

Status Indicator	Message
■ (Green square)	Process: <i>process name</i> started.

Designer displays step status messages as each step is completed:

Status Indicator	Message
■ (Green square)	Completed

7. Use the Trace view toolbar to control your debugging session. For more information, see [“About the Trace View” on page 463](#).
8. To cancel the currently running debug session, click  **Cancel Debug** in the Trace view toolbar. Debugging stops, but the step status symbols remain in the process editor.

Tip:

During debugging, a timeout might occur due to the long-running nature of a typical debug session. Such a timeout would essentially mark the process as done, rendering the debug effort useless. Avoid setting any join or process timeouts in a process for debugging.

9. When the process debugger has finished or been cancelled, Designer updates the process status Message and End Time information in the Trace view:

Status Indicator	Message
■ (Green square)	Process: <i>process name</i> is done.
▲ (Yellow triangle)	Process: <i>process name</i> is done with warnings.
● (Red circle)	Process: <i>process name</i> is done with errors
	OR
	Process: <i>process name</i> was cancelled.

10. Click  **Clear** on the Trace view toolbar to clear the Trace view and remove all debug status symbols in the editor. This button is not enabled while you are debugging a process.

The process debugger can monitor Designer's memory usage while you are debugging, and prompt you to end the session if memory use exceeds a set limit. See [“Configuring Process Debug Preferences” on page 458](#).

Navigating Steps during Debugging

Use the buttons on the Trace view toolbar to control your navigation through the process. All step navigation buttons and their corresponding menus are active during debugging. The Trace view is the active view when you start debugging, providing immediate access to available keyboard shortcuts. For more information, see [“About the Trace View” on page 463](#).

Running/resuming a process means executing it. For a call activity or a subprocess, this means execute to the next enabled breakpoint, or to the end if no breakpoints are enabled.

Stepping over a step means executing it and stopping at the next step on the same level.

- For a call activity or a subprocess, this means stepping over the entire process invoked in the call activity or all the steps inside the subprocess, and stopping at the next step after the call activity or subprocess.
- For a looped call activity or BPMN Subprocess, this means stepping over all loop iterations and stopping at the next step after the call activity or BPMN Subprocess step. You can set a breakpoint on a step inside of the looped step in order to debug loop iterations.
- For a looped webMethods Subprocess, this means stepping over all loop iterations and stopping at the next step after the webMethods Subprocess (Deprecated) step. You can set a breakpoint on the step itself and step over each iteration of the loop.

Stepping into a step means entering it. You can step into call activities and subprocesses.

Stepping out from a step means exiting it and stopping at the next step one level up. You can step out from steps inside call activities and subprocesses. If no level above exists, stepping out behaves just like Run/Resume.

- From inside a call activity or subprocess, this means exiting the call activity or subprocess step and stopping at the next step after it.
- From a looped call activity or subprocess step, this means executing all iterations of the loop and stopping at the next step.

Note:

Step navigation moves through the levels of the process. The actions of the buttons are represented by arrows that point up (out), down (in), and over (over). Think of the process as a stack of steps. You start at the top level. Each time you enter (step into) a call activity or a subprocess, you go down a level in the stack. The process debugger stops at the next executable step (or at the next enabled breakpoint). Stepping over stays at the same level and stops at the next executable step (or at the next enabled breakpoint). Stepping out goes up a level and stops at the next executable step (or at the next enabled breakpoint). If there are no more executable steps at the level you choose, the process debugger stops at the next available executable step (or at the next enabled breakpoint).

You can run/resume a process from the current step at any time.

You can perform step in, out, or over actions on the *default step*, using the button, or on a selected *eligible step*, using the menu.

Timer boundary events, both interrupting and non-interrupting, support breakpoints and can be eligible steps.

Note:

Other boundary events (message, error, signal) do not support breakpoints, and cannot be eligible steps. If you step over a message, error, or signal event, debugger stops on the step after it.

Tip:

The *default step* is always the next *eligible step* in the active process track.

About the Trace View

The Trace view appears in the Process Debug perspective. Use the Trace view to control your navigation through steps, and to see the progress of those steps, including errors, while debugging a process. The Trace view displays details of each step in a debugging session, and contains a toolbar you use to control the session in the active process editor tab. Click a row in the Trace view to inspect the corresponding pipeline data output in the Pipeline Data view.

The Trace view has a toolbar you can use to perform actions on the process as you debug it. The Trace view is the active view when you start debugging, providing immediate access to keyboard shortcuts. The following buttons are available, and available shortcut keys are noted:

The table below lists and describes the buttons.

Button	Description
 Run / Resume F8	<p>Run the process from the current step to the next enabled breakpoint or to the end of the process if no breakpoints are enabled. Parallel tracks run simultaneously. Non-parallel tracks run sequentially.</p> <p>When you click Run/Resume, all tracks run. The selected eligible step does not affect this.</p>
 Cancel Debug	<p>Cancel all processes associated with the current Process Debug session. This action is reflected in the Trace view.</p> <p>Canceling stops all steps in the parent process and any child or descendant processes. Process debugging step status symbols remain displayed in the process editor.</p>
 Clear	<p>Clear the Trace view and remove all Process Debug status symbols from the process editor.</p>
 Step In F5	<p>Enter a call activity or subprocess.</p> <p>When you click Step In on a step that does not support stepping in, the process debugger steps over instead.</p>

Important:

Button	Description
	The button action applies to the <i>default step</i> , which is outlined in the process editor. The menu beside the button allows you to step into an <i>eligible step</i> other than the default.
 Step Over F6	<p>Run the step. If the step is a subprocess or call activity, run all the steps in it to the next enabled breakpoint or to the end of the subprocess or call activity if no breakpoints are enabled.</p> <p>When you step over a child process (of a call activity step) in the course of debugging its parent process, and the child process contains an enabled breakpoint, the process debugger opens the child process in a new process editor tab and stops at the breakpoint. The Trace view and the process editor reflect the current step status.</p> <p>Important: The button action applies to the <i>default step</i>, which is outlined in the process editor. The menu beside the button allows you to step over an <i>eligible step</i> other than the default.</p>
 Step Out F4	<p>Exit from a step inside a call activity or subprocess.</p> <p>When you click Step Out from a step that does not support stepping out, the process debugger runs the process to the next enabled breakpoint, or to the end if no breakpoints are enabled.</p> <p>Important: The button action applies to the <i>default step</i>, which is outlined in the process editor. The menu beside the button allows you to step out of an <i>eligible step</i> other than the default.</p>

Below the toolbar, the Trace view shows a table that contains the following information about the steps in the process being debugged. You can click column headings to change the sort order of the steps.

The table below lists and describes the steps.

Column	Description
 Status Indicator	<ul style="list-style-type: none"> ■ Green square = step or process completed. ▲ Yellow triangle = step or process completed with warnings. ● Red circle = step or process completed with errors OR process canceled.
Step	Step Label.
Step ID	Step ID.

Column	Description
Step Iteration	Number of step iterations. Represents the loop count for transition (sequence flow) and for webMethods Subprocess (Deprecated) steps.
Loop Iteration	Number of loop iterations. Represents standard loop count for all Call Activity steps and BPMN Subprocess steps.
Start Time	Step start time, and process start time when debugging starts. The time format is governed by your operating system settings.
End Time	Step end time, and process end time when debugging has completed. The time format is governed by your operating system settings.
Duration	Step duration time, and process duration when debugging has completed.
Status Message	Details of step or process status, corresponding to the  Status Indicator .

Tip:

The **End Time** and **Duration** options share a column in the table. You can choose which to show by right-clicking anywhere in the table and selecting **Toggle between End Time and Duration**. Designer retains your setting until you change it again.

Debug exceptions appear in the Trace view. If you want to see more of the error information, hold the cursor over the error row. Designer displays extended error data, including a stack trace, if appropriate. You can copy the text in the window. Press the F2 key for focus (after the window has focus, the prompt to do so disappears), and then CTRL+C to copy. The text is highlighted in blue after it has been copied, so you know when you can paste.

If you stop debugging a process in response to a prompt that you have reached the **Peak Memory Use (percent)** setting as described in [“Configuring Process Debug Preferences” on page 458](#), Designer adds a descriptive row to the Trace view.

The Trace view reflects the process in the active process editor.

Note:

The Trace view must be active for its shortcut keys to be enabled.

If you close the Trace view and want to reopen it, click **Window > Show View >  Trace**.

Debugging an Intermediate Receive Step

The process debugger supports multiple receive steps and multiple input documents. When a receive step is configured to start a process, the process debugger presents the input document at the start of the debug session. A receive step that can start a process might be a Message Start Event, a Signal Start Event, or a Receive Task.

When a receive step is not configured to start a process, it must join the process instead. A receive step that cannot start a process might be an Message Intermediate Event, Signal Intermediate Event, or a Receive Task. The process debugger does not present joining documents at the start of the debug session.

In order for a process to receive a joining document during debugging, you must publish the document manually. If the receive step receiving the document is not inside of a BPMN subprocess, you can publish the document at any time after the debug session starts. If the receive step is inside a BPMN Subprocess, you must publish the document after the subprocess starts in order for it to be received. The receive step must be active when the document arrives.

The process debugger does not stop on an intermediate start step (a receive step, for example) unless you set a breakpoint on it. If a process thread starts from an intermediate start step, the debugger runs through the thread to its completion by default if no breakpoint is set on the start step.

Debugging Parallel Tracks

The process debugger runs parallel tracks simultaneously, and marks eligible steps in each track. All step navigation buttons are enabled, including their corresponding menus.

When a process has multiple parallel tracks that lead to an OR join, at least one transition must be taken. The tracks not taken are called *dead paths*, because they do not execute. The process debugger waits for all transitions, both live and dead paths, to arrive at an OR join step before that step can become an *eligible* step, and before the process can proceed to any step after the join step. A join step sometimes enters the waiting state before it appears that any transitions to it have been taken; this is due to the arrival of an incoming dead path transition.

Important:

For an OR join, at least one incoming transition must be live.

Important:

For an AND join, all incoming transitions must be live.

For more information about dead paths, refer to [“How Incomplete Transitions Affect Join Steps and Gateways” on page 132](#).

Debugging a User Task

When you debug a User Task in your process, Designer displays a browser, prompts you to log into My webMethods Server, and then displays the Task List Management page, where you can mark the webMethods task as **Completed**. When this is done, process debugging resumes.

If a process contains a user task step that has not been implemented with a webMethods task, such that there is no Task ID available at run time, the Process Engine does not attempt to execute the step; instead, the empty task step is marked completed, just as empty steps of other types, and the process continues. This allows for greater iterative testing and debugging, since the task can be added and the process regenerated at any time during development.

Tip:

Design your processes with error handling for step failures for the most flexibility at run time and while debugging.

Note:

The process debugger does not support stepping into user task steps.

Debugging a Call Activity

Call activity steps are always displayed in collapsed form in the process model. When you step into a call activity, the process debugger opens the expanded call activity in a new editor tab and also opens a dedicated Trace view for it.

Call activity steps can be looped. This means all the steps within the call activity can be executed multiple times. Call activity steps, including BPMN Call Activity and webMethods Referenced Process (Deprecated) steps support standard loops. Steps with standard loops display a loop marker  in the process editor.

The process debugger does not support webMethods dynamic referenced process steps. Step Over the step to skip it; there is nothing inside it for the debugger to run, so it is bypassed, and you stop at the next eligible step.

Debugging a Subprocess

Subprocesses can be displayed in either collapsed or expanded form in the process model. When you step into a collapsed subprocess, the process debugger expands it in place in the process editor. When the debug session ends, the subprocess returns to its original state, whether collapsed or expanded.

The Trace view displays data for a subprocess, whether it is collapsed or expanded in the process editor.

Subprocess steps can be looped. This means all the steps within the subprocess can be executed multiple times. BPMN Subprocess steps and webMethods Subprocess (Deprecated) steps support standard looping. Steps with standard loops display a loop marker  in the process editor.

As does webMethods Monitor, the Trace view displays loop data differently for BPMN Subprocess steps and webMethods Subprocess (Deprecated) steps. For more information, refer to [“Debugging a Looped Step” on page 467](#).

Debugging a Looped Step

Call activity and subprocess steps can be configured to loop. Looping means to repeatedly run the steps within the top-level call activity or subprocess step.

Call activities (BPMN Call Activity steps and webMethods Referenced Process (Deprecated) steps) and subprocesses (BPMN Subprocess steps and webMethods Subprocess (Deprecated) steps) support standard loops. Standard loops are configured on the Loops tab in the Properties view of the step.

The Trace view displays Loop Iteration information for looped call activities (BPMN Call Activity steps and webMethods Referenced Process (Deprecated) steps) and BPMN Subprocess steps.

webMethods Subprocess (Deprecated) steps, though configured the same way, do not loop the same way. They are displayed differently in webMethods Monitor and in the Trace view. The

difference is that the loop count, represented in the Loop Iterations field for other looped steps, is represented in the Step Iteration column instead. Another difference is that you can set a breakpoint on a webMethods Subprocess (Deprecated) step and step over (execute) the loop one iteration at a time. For the other looped steps, you can debug internal steps by setting a breakpoint on an internal step, not on the parent step.

Step iterations are also counted in transition looping, called sequence flow looping in BPMN. This kind of looping involves using multiple steps and transitions to achieve a looped structure. Use standard looping for a single looped step configuration. For more information, refer to [“About Transition Looping” on page 129](#).

Finally, a step iteration can also represent a step Retry Count, which is configurable on the Implementation tab in the Properties view. See [“About Retries” on page 155](#).

You can use breakpoints to help debug looped call activities and BPMN subprocesses. For example, you can set a breakpoint inside of a BPMN Subprocess (not on the subprocess step itself). When you reach the breakpoint, the process debugger opens the BPMN Subprocess and stops there. You can then click **Run/Resume** to loop around once and stop at that breakpoint again. Finally, from the inside of a looped BPMN Subprocess step, click **Step Out** to complete all remaining iterations of the loop and stop at the next step.

webMethods Monitor supports tree displays for looped call activities (BPMN Call Activity and webMethods Referenced Process (Deprecated)) and BPMN Subprocesses. Monitor displays a flat list for a looped webMethods Subprocess (Deprecated) step.

As you debug a looped step and execute the steps inside the loop multiple times, the Trace view displays each step and iteration separately. The process debugger opens a looped call activity step in a new process editor tab and displays its own Trace view.

When you debug a process with a looped step, the process debugger marks the transitions with a bold line the first time through the loop, leaving the transition lines bold until the loop has completed all its loop iterations.

Transition lines are not reset when loop repeats. For Call Activity steps and BPMN Subprocess steps, the Running status symbol  is displayed until the last loop is completed and the outbound transition is taken. For webMethods Subprocess (Deprecated) steps, the eligible step status symbol  is added each time the process debugger arrives at the looped step, and is replaced by the completed status symbol  each time the loop completes.

Debugging a Boundary Event

Timer boundary events, both interrupting and non-interrupting, support breakpoints and can be eligible steps.

Important:

Other boundary events (message, signal, and error) do not support breakpoints, and cannot be eligible steps. These boundary events respond to non-boundary events when debugging the same way they do when running. If you step over a message, error, or signal boundary event, debugger stops on the step after it.

Debugging a Process-Wide Timeout

The process debugger supports process-wide timeout steps, with and without input transitions, and with and without breakpoints. In order to be able to stop at a timeout step that has no incoming transition, you must set a breakpoint on the timeout step. There are two ways to stop at a timeout step that does have an incoming transition. One is to set a breakpoint, and the second is step through the process if there is no breakpoint.

If you set a breakpoint on the timeout step, the timeout step should become eligible for execution. You can choose to step through or run. If you choose not to execute the step when it becomes eligible, the process should complete without running the timeout path.

If you do not set a breakpoint on the timeout step, the behavior is different depending on whether there's an incoming transition.

For a timeout step with an incoming transition and no breakpoint: you can choose to run the debug session, which will allow the process to execute without intervention as in the Process Engine (when the timer expires, the process takes the timeout path immediately). You can instead choose to step through the debug session, during which the timeout step with an incoming transition will become eligible when the timer expires.

For a timeout step with no incoming transition and no breakpoint: when the timer expires, the process debugger takes the timeout path immediately, just as in the Process Engine.

Debugging a Join Timeout

In earlier versions of the process debugger, join timeouts were fired manually. You can still choose whether or not to execute a join timeout in the process debugger. The interaction is somewhat different.

When a join times out, the join step becomes eligible for debugging.

Note:

There is no visual distinction between a join step that has timed out and a join step that has completed successfully.

When a join step becomes eligible due to a timeout, you may or may not choose to execute that step right away. If you execute the join step BEFORE the join is complete, debugger takes the timeout transition. If you do NOT execute the join step right away, and the join eventually completes, the process behaves as if the join never timed out. In this way, you can choose whether or not to take the join timeout transition while debugging without having to adjust the timeout expiration settings.

Note:

The process debugger does not stop on an intermediate start step (a receive step, for example) unless you set a breakpoint on it. If a process thread starts from an intermediate start step, the debugger runs through the thread to its completion by default if no breakpoint is set on the start step.

Modifying Input Data

When you debug a process, you can modify the input data in order to test alternative behaviors. You can do this when you configure a Debug Configuration, when you start debugging, and also during debugging. You can save input data to a file, and you can load input data from a file. These files are stored in .xml format.

By default, the Debug Configuration uses the data on the Process Inputs tab if it exists. Default input data is optional. You can enter input data directly in the Enter input for *document name* dialog box or load it from a file. For more information, see [“Creating a Process Debug Configuration” on page 459](#) and [“Loading Input Data” on page 470](#).

You can edit the input pipeline for one or more eligible steps, one at a time. The modified data is saved in the Process Engine database and used when the step is executed. Editing the input pipeline does not affect the default step selection.

➤ To modify input data during debugging

1. In the process editor, right-click an eligible step whose input data you want to modify.
2. Click **Edit input data**.
3. Enter input data directly in the Enter input for *document name* dialog box or click **Load** to load it from a file.
4. Click **OK** to resume debugging with the modified data.

Loading Input Data

When you create a Debug Configuration, you can provide input data for your process. You have another opportunity to supply input data if you are prompted at the start of a debug session. This is configurable, as described in [“Creating a Process Debug Configuration” on page 459](#).

Loading input data works the same way in the Debug Configurations dialog box and the Enter input for *document name* dialog box.

➤ To load input data

Do one of the following:

- Click the **Load** button to populate the input data from a saved file.
- Click the **Load/Replace** button to replace existing input data with the contents of the saved file.

Saving Input Data

When you create a Debug Configuration, you can save input data for your process to a file that you can use for subsequent process debugging. You have another opportunity to save input data if you are prompted for input data at the start of a debug session. This is configurable, as described in [“Creating a Process Debug Configuration” on page 459](#).

Saving input data works the same way in the Debug Configurations dialog box and the Enter input for *document name* dialog box.

› To save input data

1. Enter input data in the form.
2. Click the **Save** button to save this input data to a file.

About the Pipeline Data View

The Pipeline Data view appears in the Process Debug perspective. It displays the output pipeline data of the step selected in the Trace view. You can expand the information displayed to see more details, providing an opportunity to troubleshoot the behavior of the overall process at the step level. You can also view and capture exception data.

You can copy the output pipeline of one or more steps. The Pipeline Data view displays errors if it encounters them, and allows you to capture the contents of those errors without having to locate the information in the log file.

The Pipeline Data view displays the following information, as described in the table below:

Column	Description
Pipeline	Shows the pipeline element field names.
Field Value	Shows the field value of the data at the selected step.

Note:

No pipeline data is shown for the row representing the overall process status.

The table below lists and describes the buttons in Pipeline Data view toolbar.

Button	Description
 Expand/collapse all items	Click to expand a collapsed tree, or to collapse an expanded one. By default, Designer opens the collapsed view, and will automatically expand all items. This can be changed in Process Debug preferences. See “Configuring Process Debug Preferences” on page 458 .
 Expand/collapse all items	

Button	Description
 Copy the pipeline for this step to the clipboard	Click to copy the pipeline data for the selected step to the clipboard.
 Copy the pipelines for all steps to the clipboard	Click to copy the pipeline data for all steps to the clipboard.

The Pipeline Data view reflects the process in the currently active process editor tab.

If you close the Pipeline Data view and want to reopen it, click **Window > Show View > Pipeline Data**.

Working with Breakpoints

Breakpoints are stopping points used for debugging processes. You can set breakpoints on process steps. You cannot set breakpoints on pools, swimlanes, notes, annotations, gateways, or boundary events. These are not steps, and they do not run.

You can set breakpoints on the following step types:

- Activities: task, subprocess, and call activity steps
- Start events: None Start, Message Start, and Signal Start
- Intermediate events: None Intermediate, Message Intermediate, and Signal Intermediate
- End events: None End, Message End, Signal End, Error End, and Terminate End
- Boundary events: Timer boundary events, both interrupting and non-interrupting. Timer boundary events support breakpoints and can be eligible steps.

Note:

Other boundary events (message, error, signal) do not support breakpoints, and cannot be eligible steps. If you step over a message, error, or signal event, debugger stops on the step after it.

Breakpoints are visible in all perspectives. Once you set a breakpoint, you can enable, disable, or remove it using the context (right-click) menu in the process editor. You can also use the Breakpoints view to enable and disable a breakpoint, remove one or more breakpoints, and skip all breakpoints.

You can set, enable, disable, or remove breakpoints at any time before or during debugging. The active process in the editor is automatically updated. The process debugger evaluates breakpoint information immediately prior to executing a step. Breakpoint settings are stored in the workspace, and persist across Designer sessions.

You can use breakpoints to help debug looped call activities and BPMN subprocesses. For example, you can set a breakpoint inside of a BPMN Subprocess (not on the subprocess step itself). When you reach the breakpoint, the process debugger opens the BPMN Subprocess and stops there. You

can then click Run/Resume to loop around once and stop at that breakpoint again. Finally, from the inside of a looped BPMN Subprocess step, click Step Out to complete all remaining iterations of the loop and stop at the next step.

The process debugger does not stop on an intermediate start step (a receive step, for example) unless you set a breakpoint on it. If a process thread starts from an intermediate start step, the debugger runs through the thread to its completion by default if no breakpoint is set on the start step.

While you are stopped at a breakpoint, you can select the step in the Trace view and review the data in the Pipeline Data view. Click any of the enabled buttons in the Trace view to resume your debugging session.

When you set and enable a breakpoint on a subprocess or call activity step, the process debugger stops before entering the step, but it does not stop before exiting the step. You cannot set or enable a breakpoint at the end of a subprocess or call activity.

When you step over a child process (of a call activity step) in the course of debugging its parent process, and the child process contains an enabled breakpoint, the process debugger opens the child process in a new process editor tab and stops at the breakpoint. The Trace view and the process editor reflect the current step status.

About the Breakpoints View

The Breakpoints view is a default view in the Process Debug perspective. The Breakpoints view displays a list of breakpoints that exist in your workspace.

When you double-click a breakpoint in the Breakpoints view, Designer displays the process in the editor and selects the step automatically.

The Breakpoints view allows you to enable and disable selected breakpoints; remove selected or all breakpoints; and skip all breakpoints. You can display breakpoints by groups based on their common properties, such as Process Steps. You can apply changes to groups.

If you close the Breakpoints view and want to reopen it, click **Window > Show View > Breakpoints**.

Note:

You cannot set a breakpoint in the Breakpoints view. Set a breakpoint by right-clicking a step in the process editor. For more information, see [“Setting a Breakpoint” on page 473](#).

Setting a Breakpoint

You can set a breakpoint on any step in any process at any time, even while debugging.

> To set a breakpoint

- In the process editor: Right-click a step and click **Toggle Breakpoint**.

Toggle Breakpoint is a toggle switch. This means there are exactly two opposite states: set or remove. Clicking this button when no breakpoint is set changes the setting to set one instead.

Removing a Breakpoint

You can remove a breakpoint from any step in any process at any time, even while debugging.

> To remove a breakpoint

Do one of the following:

- In the process editor: right-click the step and click **Toggle Breakpoint**.
- In the Breakpoints view: right-click the step and then click **✖ Remove**.

Toggle Breakpoint is a toggle switch. This means there are exactly two opposite states: set or remove. Clicking this button when a breakpoint is already set changes the setting to remove the breakpoint instead.

- In the Breakpoints view, click to select a breakpoint. Then click **✖ Remove** on the Breakpoints view toolbar.

Tip:

You can remove multiple breakpoints in the Breakpoints view. Select the steps, right-click, and click **✖ Remove All**, or click **✖ Remove All** on the Breakpoints view toolbar.

Enabling a Breakpoint

You can enable an existing breakpoint in any process at any time, even while debugging. For more information on breakpoint support, refer to [“Working with Breakpoints” on page 472](#).

> To enable a breakpoint

Do one of the following:

- In the process editor: right-click the step and click **Enable Breakpoint**.
- In the Breakpoints view: right-click the step and click **Enable**.
- In the Breakpoints view: select the check box for the step.

Disabling a Breakpoint

You can disable a breakpoint on any step in any process at any time, even while debugging.

> To disable a breakpoint

Do one of the following:

- In the process editor: right-click the step and click **Disable Breakpoint**.
- In the Breakpoints view: right-click the step and click **Disable**.
- In the Breakpoints view: clear the check box for the step.

Skipping All Breakpoints

You can skip all the breakpoints in your workspace with a single click.

> To skip all breakpoints

- In the Breakpoints view: click **Skip All Breakpoints**.

Skip All Breakpoints is a toggle switch. This means there are exactly two opposite states: skip or do not skip. Clicking this button when breakpoints are already being skipped changes the setting to stop skipping them.

26 Working with CentraSite Metadata for BPM and CAF Assets

■ About Metadata in CentraSite	478
■ Publishing Metadata for Processes and CAF Applications	481
■ Retracting Metadata for Processes and CAF Applications	482
■ About Searching for Asset Metadata	483
■ Performing a Basic Search for webMethods Assets	484
■ Performing an Advanced Search for webMethods Assets	485
■ Configuring Search Preferences	486
■ About the Search View	487
■ About Working with Saved Searches	490
■ Working with Saved Searches	491

About Metadata in CentraSite

CentraSite Metadata enables you to locate and reuse shared webMethods and other process assets in Designer.

Metadata is data that describes the assets in webMethods components. An *asset* is any object you create and work with in a webMethods component. For example, in Designer, assets include processes, tasks, portlets, Integration Server documents, e-forms, Integration Server services, and web services. CentraSite captures metadata about each Designer asset stored in its repository.

CentraSite stores a variety of metadata about an asset. All assets have names and (optionally) descriptions. An asset also may have values for certain properties, such as the Process ID of a process. If an asset references other assets, for example a task step that references a service, it will have metadata about such associations.

The purpose of metadata is to provide information describing each asset, what it contains, and what assets it references or depends on. The CentraSite registry enables reuse of assets among users. You can publish metadata about an asset to CentraSite. Then, other users can see the published metadata and incorporate one or more references to the asset into their own projects. If an asset should not be reused, you can *retract* the assets from CentraSite.

To access metadata, you use the Designer search functionality. You can search metadata for information about various assets.

Designer includes two CentraSite perspectives with multiple views and tools for a customized CentraSite experience. You can access the perspectives from:

- **Window > Open Perspective > CentraSite Asset Catalog**
- **Window > Open Perspective > CentraSite Search and Browse**

For more information about working with the CentraSite perspectives, see the *CentraSite Help*.

In addition, the Designer Process Development perspective includes the Registry Explorer view, which enables you to access your CentraSite assets without switching perspectives.

Note:

Local metadata that is stored on your computer is referred to as workspace metadata. Workspace metadata is stored in a database, on your computer, called the workspace index. For more information about working with workspace metadata, see *webMethods BPM and CAF Workspace Metadata Help*.

Working with CentraSite Metadata

Consider the following information when working with CentraSite metadata:

- CentraSite is a shared repository and registry. Assets are stored in the repository, and information about those assets, or metadata, is stored in the registry. Both the workspace index and the CentraSite registry contain metadata about assets. The difference is that the workspace index is accessible only by you on your own computer, while the CentraSite registry is accessible to other users. The CentraSite registry resides on and is maintained by a server on the network.

- webMethods assets are pre-installed in CentraSite. When you publish metadata to CentraSite from the Package Navigator view or Solutions view, Designer performs a check to ensure that CentraSite has the latest asset definitions. If it does, your publish action is completed. If it does not, you must update your asset definitions in CentraSite before you can publish to it. Contact your CentraSite administrator to have the updated definitions installed.
- Although it is possible to configure and connect to multiple CentraSite registries, Designer can connect to a single registry at a time. For more information about connecting to CentraSite, see the *CentraSite Help*.
- Each CentraSite user can belong to only one *organization*. By default, each member of a CentraSite organization is assigned the roles of Asset Provider and Asset Consumer in the Users group of that organization, allowing the publishing and usage of assets there. Administrators can choose to revoke or change role assignments in the Users group, so the default assignments might be different for some organizations. Administrators can also assign permissions to individual users and groups to view and create (publish) assets in other organizations. Designer does not allow users to select an organization when they publish; it publishes using the organization to which the user belongs. As such, it is important for all collaborating Designer users working on the same project to publish to CentraSite using the same organization. Publishing using multiple organizations could lead to access problems, duplicates, and lost data.
- When you reuse an asset from CentraSite, only a minimally required amount of information about that asset is transferred to the workspace where you use it. As a result, when you query the workspace, you do not find the same metadata that you find when you query CentraSite directly.
- Reference chains in CentraSite are not fully reflected in the workspace. For example, if process A references process B, which references process C in CentraSite, when you reuse process B in your workspace, the relationship between B and C is not preserved.
- CentraSite does not support versioning of webMethods asset types. As such, this functionality is disabled by default for these assets. Do not manually enable versioning for them in CentraSite because this will lead to unavailable assets.
- Before making a change to an asset, you might wish to locate any assets that refer to it or are dependent on it. This helps ensure you are not making changes that adversely affect other assets.
- Although you can configure Designer to automatically generate and store metadata in the workspace index, you cannot automatically publish metadata to CentraSite. You must select an asset and manually publish it.
- Workspace indexing is not required for publishing to CentraSite. Local (workspace) metadata about an asset is created automatically when you enable workspace indexing and is stored locally in the workspace index, which is accessible only on your computer. For more information, see *webMethods BPM and CAF Workspace Metadata Help*.

About The Registry Explorer View

The Registry Explorer view displays the contents of the CentraSite registry to which Designer is currently connected. To connect to a different CentraSite registry, you must configure it in Preferences. You can connect to only one CentraSite registry at a time, but you can configure multiple registries.

The CentraSite registry often contains a great amount of data. Designer provides a view filter for the Registry Explorer view to make it easier to locate webMethods assets.

Note:

Due to limitations in the API, the webMethods Assets filter shows some non-reusable assets, such as generated wrapper services and items that have been referenced but not published themselves. These are not available for use in processes. The Registry Explorer view allows you to drag them out of the view, but Designer does not allow you to drop them into any editors.

The Registry Explorer view is displayed by default in the Process Development perspective. If you close the Registry Explorer view and want to reopen it, or if you want to open it in another perspective, click **Window > Show View >  Registry Explorer**.

Note:

If the CentraSite connection is disabled or not properly configured, the Registry Explorer view displays a message indicating so and a link to CentraSite preferences, where you can enable or configure the connection. For information about configuring a connection, see the *CentraSite Help*.

The following table describes the actions that you can perform in the Registry Explorer view.

To	Take this action in the Registry Explorer view
Enable or disable the webMethods Assets filter	To enable the webMethods Assets filter, click ▾ View Menu > Configurations > webMethods Assets . Designer retains your selection in your current workspace until you change it. To disable it, select a different filter, such as Default .
Expand an asset type	The Registry Explorer view is organized by asset type. Click  next to an asset type to view the assets under it.
Refresh the view	Right-click anywhere in the view and click  Refresh . You may need to refresh if you publish metadata with the Registry Explorer view open. Newly published metadata will not appear in the Registry Explorer view until you refresh it. Refreshing the view collapses any currently expanded asset type.
Use an asset in one of your own projects	Drag the asset from the Registry Explorer view onto the process editor canvas for the project.

To	Take this action in the Registry Explorer view
	<p>When you use an asset from the Registry Explorer, you are creating a reference to that asset. You are not including the asset itself in your project; rather, you are referencing the asset based on its metadata in CentraSite.</p> <p>Note: The Reuse property for an asset determines whether the asset can be reused in other projects or canvases.</p>
Retract metadata about an asset	<p>Right-click the asset and click Retract.</p> <p>This removes the asset's metadata from CentraSite.</p> <p>Note: You cannot publish or retract process simulation metadata.</p>

Publishing Metadata for Processes and CAF Applications

You can publish multiple assets at the same time. CTRL + click the assets, right-click any of them, and then click **Publish**.

➤ To publish process or CAF application metadata to CentraSite

1. In the Solutions view, right-click an asset and click **Publish**.
2. In the **Publish Assets** dialog box, select the assets whose metadata you want to publish.

By default, the assets you selected will be marked for publishing to CentraSite. If you do not want metadata for an asset published, clear the check box next to the asset name.

3. Click **OK**.

Notes:

- If you have the Registry Explorer view open, the assets you published should appear in it when the publish operation completes. You may have to refresh the view to see the newly published assets.
- Published metadata may not be visible immediately in the Registry Explorer view or appear in searches performed immediately after publishing. Depending on the amount of metadata being published, and network and computer resources, there may be a slight lag. Generally, if you do not see published metadata within a few minutes, contact your CentraSite system administrator to check on the operation's progress.
- When a process is published, only the process is marked as *published*. The various services, processes, documents, etc. used by the process are *referenced*, not published. To search for,

retract, or reuse a CentraSite asset, it must first be published. However, note that an unpublished asset can exist in the CentraSite registry as a reference.

For example, If you publish a process (procA) and then use it in a second process (procB) but you do not publish the second process (procB), CentraSite contains one process (procA) with no references to it. You can retract that process (procA), because it is published, and it will be removed from the CentraSite registry completely. CentraSite does not keep track of whether or not an asset has been used outside of CentraSite; CentraSite only knows what has been published.

Retracting Metadata for Processes and CAF Applications

Metadata about an asset can be removed from CentraSite — this is referred to as *retracting* metadata. When you retract metadata about an asset from CentraSite, it is no longer available to other users for use in their own Designer projects.

Keep the following points in mind when retracting metadata for processes or CAF applications:

- Retracting metadata about an asset will not affect other users who have already referenced the metadata in their own projects; however, the retracted metadata will no longer be available for future referencing by other users.
- Uninstalling Integration Server can cause any published Integration Server assets to become unreferenced, or orphaned. Retract any published Integration Server assets before uninstalling the Integration Server that contains those assets.
- Problems with retracting can occur if multiple processes with multiple owners exist in a single process project, even if the users belong to the same organization. For example, if User1 publishes the process procA, the ownership of procA and its parent process project (projA) are both set to (User1). User2 publishes another process (procB) to the same process project (projA). When User1 retracts the procA, User2 cannot retract procB because User2 is not the owner of its parent process project (projA), whose only asset is now procB.
- You can publish/retract metadata for a multiple assets by pressing the CTRL or SHIFT keys while selecting assets in Solutions view.

➤ To retract process or CAF application metadata from CentraSite

1. In the Solutions or Registry Explorer view, right-click an asset and click **Retract**.
2. In the **Retract Assets** dialog box, select the assets you want to retract.

By default, the assets you selected will be marked for retraction from CentraSite. If you do not want metadata retraced for an asset, clear the check box next to the asset name.

3. Click **OK**.

Designer retracts the assets.

If you have the Registry Explorer view open, the assets you retracted should no longer appear in it. You may have to refresh the view to see this.

About Searching for Asset Metadata

To locate assets and/or asset references, Designer searches the metadata associated with the assets. You can choose to search only the workspace index, only CentraSite, or both the workspace index and CentraSite.

Important:

When you reuse an asset from CentraSite, only a minimally required amount of information about that asset is transferred to the workspace where you use it. As a result, when you query the workspace, you do not find the same metadata that you find when you query CentraSite directly. Additionally, reference chains in CentraSite are not fully reflected in the workspace. For example, if process A references process B, which references process C in CentraSite, when you reuse process B in your workspace, the relationship between B and C is not preserved.

Note:

If you have workspace indexing disabled now or had it disabled in the past, you might have assets in your workspace for which there is no or incomplete metadata. In this case, the search will not properly locate these assets. For more information, see *webMethods BPM and CAF Workspace Metadata Help*.

The following table lists the types of searches that Designer provides for searching for assets.

Type of Search	How to Access	Description
Basic keyword search	webMethods tab in the Search window	Use to locate assets or asset references based on keywords contained in the asset's name and/or description. In both Basic and Advanced searches, you can locate all assets of a particular type by selecting the asset and specifying no search criteria.
Advanced search	webMethods Advanced tab in the Search window	Use to locate assets of a single type. For example, you can search for portlets or you can search for processes, but not both for portlets and processes in a single search. The advanced search enables you to specify detailed search criteria to locate specific assets. For example, you can search for processes with names that contain "HR", with a model version that is equal to "2", and that references another process with a name that contains "approve". In both webMethods basic and advanced searches, you can locate all assets of a particular type by selecting the asset and specifying no search criteria.

Type of Search	How to Access	Description
Saved searches	Saved Searches view	<p>Use to locate assets of a single type. For example, you can search for portlets or you can search for processes, but not for both portlets and processes in a single search.</p> <p>Designer provides predefined saved searches of assets used in your solutions. You can search your workspace for all CAF task types, portlets, or processes.</p> <p>You can define and save additional searches. When defining your own saved search, you can specify detailed search criteria to locate specific assets, similar to the criteria you can specify for an advanced search.</p>

Performing a Basic Search for webMethods Assets

Use a basic search to locate assets or asset references based on keywords contained in the asset's name or description. Designer displays the results of your search in the Search view.

To locate assets or asset references, Designer searches the metadata associated with the assets. You can choose to search only the workspace index, or both the workspace index and CentraSite.

Note:

If you have workspace indexing disabled, you might have assets in your workspace for which metadata does not exist or is incomplete. In this case, the search will not properly locate these assets.

> To perform a basic search for webMethods assets

- In Designer, select **Search > Search** .
 - The **webMethods** tab has focus by default.
 - Selections from the previous search are retained.
- In the **Name or description contains** field, specify a keyword that is contained in the name or description of the assets you want to locate. The search is not case sensitive.
- Select the type of assets that you want to locate from the **Asset Type** list. You can select **All** to search for all types of assets and asset references, or you can select one of the listed asset types.
- In the **Search in** field, click:

- **Workspace** to search metadata in the workspace index only.
- **CentraSite** to search metadata in CentraSite only.
- **Workspace & CentraSite** to search metadata in both the workspace index and CentraSite.

5. Click **Search**.

Performing an Advanced Search for webMethods Assets

Use an advanced search to specify detailed search criteria to locate specific assets. When you perform an advanced search, you can only locate assets of a single type. For example, you can search for portlets or you can search for task types, but not for both portlets and task types in a single search. Designer displays the results of your search in the Search view.

To locate assets and/or asset references, Designer searches the metadata associated with assets. You can choose to search only the workspace index, or both the workspace index and CentraSite.

Note:

If you have workspace indexing disabled, you might have assets in your workspace for which metadata does not exist or is incomplete. In this case, the search will not properly locate these assets.

> To perform an advanced search for webMethods assets

1. In Designer, select **Search > Search** .
2. Click the **webMethods Advanced** tab.
 Selections from the previous search are retained.
3. Select the type of asset you want to locate from the **Asset Type** list.
4. For each search criterion you want to use:
 - a. Click **Add**.
 - b. In the Add Query Condition window, click **Asset Field** and select an item. The items that Designer includes in the **Asset Field** list depends on the **Asset Type** you select.
 - c. Complete specifying the condition for the selected **Asset Field**. Note that the search is not case sensitive.
 - d. Click **OK**.
5. In the **Match condition** field, select:
 - **Any** to have Designer locate assets that match any of the criteria you specified.

- **All** to have Designer locate assets only if they match all of the criteria that you specified.

Note:

If you select **References** from **Asset Field** and **All** for **Match condition**, Designer finds assets that contain all the specified references, but those references do not necessarily have to be all in one place. For example, if you have a condition to locate all processes that reference processes with names that contain "HR" and also reference processes with a process version that contains "2", Designer might find a process that contains one step for a referenced process with the name "HR-Tasks" and a process version "3" and another step for a referenced process with the name "ApproveOrder" and a process version "2".

6. In the **Search in** field, select:

- **Workspace** to search metadata in the workspace index only.
- **CentraSite** to search metadata in CentraSite only.
- **Workspace & CentraSite** to search metadata in both the workspace index and CentraSite.

Note:

Documents associated with processes are not searchable in the workspace. However, you can show the references and dependencies of processes to locate these assets.

For more information, see [“Showing Asset References and Dependencies” on page 489](#).

7. Click **Search**.

Configuring Search Preferences

You can configure standard Eclipse preferences for searching. For more information, see the Eclipse documentation.

> To configure Search preferences

1. In Designer, select **Window > Preferences > General > Search**.
2. Set the preferences you want for each of the following options. You can also click **Restore Defaults** to reset all Search preferences to their initial installed values.

For more information about each of these options, see the Eclipse documentation.

3. Click **Apply** to apply your changes without closing the Preferences window.

Click **OK** to apply your changes and close the Preferences window.

About the Search View

The Search view is a standard Eclipse view. Designer displays the results of a search in the Search view.

Search results do not display referenced assets. To see what an asset depends on, click **Show Dependencies**. To see what an asset refers to, click **Show References**.

When you choose to show references or dependencies in the Solutions or Registry Explorer view, the results are shown in the Search view.

The table below lists the actions you can take from the Search view.

Action	Description
View dependencies of an asset	<p>Right-click a row in the Search view.</p> <p>Select Show Dependencies >  In CentraSite or Show Dependencies >  In Workspace.</p> <p>Because Designer uses metadata to determine dependent assets, if you have set your Designer preferences to disable workspace indexing, the Search view might not list all dependencies. For more information, see <i>webMethods BPM and CAF Workspace Metadata Help</i>.</p>
View references of an asset	<p>Right-click a row in the Search view.</p> <p>Select Show References >  In CentraSite or Show References >  In Workspace.</p> <p>Because Designer uses metadata to determine referenced assets, if you have set your Designer preferences to disable workspace indexing, the Search view might not list all references. For more information, see <i>webMethods BPM and CAF Workspace Metadata Help</i>.</p>
Add an item in the search results to the process editor's canvas	<p>Select a row in the search results and drag it on to the process editor's canvas.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note:</p> <ul style="list-style-type: none"> ■ The item in the search result must be appropriate for the type of asset on the canvas for the drag and drop to succeed. ■ When you use an asset from CentraSite, you are creating a reference to that asset. You are not including the asset itself in your project or solution, but rather, you are referencing the asset based on its metadata in CentraSite. CentraSite assets show  CentraSite in the Store column. </div>
 Show Next Match	Selects the next item in the search results.
 Show Previous Match	Selects the previous item in the search results.

Action	Description
 Remove Selected Matches	Removes the currently selected matches from the Search view. Select one, or CTRL + click to select multiple matches.
 Remove All Matches	Removes all matches currently shown in the Search view.
 Run the Current Search Again	Repeats the current search to reflect changes to results.
 Cancel Current Search	Cancels the current search.
 Show Previous Searches	<p>Browse previously conducted searches and repeat a previous search. Shows previous searches and allows you to select one.</p> <p>Select ▾ Show Previous Searches Menu to see a list of the most recent searches. Select one to run again.</p> <p>Select ▾ Show Previous Searches Menu > History to see the history of your previous searches.</p> <p>Select ▾ Show Previous Searches Menu > Clear History to remove all previous search history information, including the one currently displayed.</p>
 Pin the Search View	Pinning the Search view means that subsequent searches display results in another Search view, leaving the pinned view and its results unchanged.

Designer displays the Search view automatically when you execute a search. If you close the Search view and want to reopen it, click **Window > Show View > General >  Search**.

Related Assets

For a selected asset, you can view related assets: references and dependencies. Referenced assets are used by the selected asset. Dependent assets use, or "depend on," the selected asset. For example, a process might use a number of Integration Server (IS) services. The IS services are references of the process.

You can view the related assets of an asset by right-clicking it in the Solutions or Registry Explorer view and then choosing **Show References** or **Show Dependencies**. The related assets are displayed in the Search view.

You can also view the related assets of an asset returned by a search. In the Search view, right-click the asset and choose **Show References** or **Show Dependencies**.

Related asset information is metadata that comes from either the workspace index (see *webMethods BPM and CAF Workspace Metadata Help*) or CentraSite. When you choose to view related assets, you are given a choice of viewing the related asset information from the workspace index or CentraSite. The related asset information for the same asset might be different in the workspace

index than in CentraSite. The latest changes to the asset reflected in the workspace index may not have been published to CentraSite, or the version in the workspace index may not have been updated with the latest changes reflected in CentraSite.

Only directly related assets are shown in the Search view. For example, if A references B and B references C, only B is shown as a reference of A. C is not shown because it is not *directly* referenced by A.

Showing Asset References and Dependencies

When you choose to show the related assets of an asset, they are displayed in the Search view:

-  References are the assets that are used by (referred to by) the selected asset.
-  Dependencies are the assets that use (depend on) the selected asset.

Note:

If you have workspace indexing disabled, you might have assets in your workspace for which metadata does not exist or is incomplete. In this case, Designer might not show all references or dependencies.

> To show the references or dependencies of an asset

1. In the Solutions or Search view, right-click the asset for which you want to view references or dependencies.
2. Select **Show References** or **Show Dependencies** and then:
 -  **In CentraSite** to show assets found in CentraSite.
 -  **In Workspace** to show assets found in the workspace index.

Note:

The related asset information for the same asset might be different in the workspace index than in CentraSite. The latest changes to the asset reflected in the workspace index may not have been published to CentraSite, or the version in the workspace index may not have been updated with the latest changes reflected in CentraSite.

3. Designer displays referenced or dependent assets (based on your selection) in the Search view.

Note:

Only direct references and dependencies are shown. For example, A references B and B references C. If you choose to view the references of A, only B is shown. Likewise, A has dependency B and B has dependency C. If you choose to view the dependencies of A, only B is shown.

About Working with Saved Searches

You perform searches that are already defined, using the Saved Searches view. These searches can include assets in your local workspace, CentraSite, or both. When you execute a saved search, Designer displays the results in the Search view.

Designer provides predefined saved searches of assets used in your solutions. You can search your workspace for all CAF Task Types, Portlets, or Processes; and you can search CentraSite for all Documents, E-forms, IS Services, Rule Services, or Web Services.

webMethods predefined saved searches are designated by the  **webMethods Predefined Saved Search** icon. You cannot edit, delete, rename, or duplicate any of the predefined saved searches.

webMethods Predefined Saved Search	Use this saved search to locate
 All CAF Task Types in Workspace	All CAF task types in your workspace.
 All Documents in CentraSite	All Integration Server (IS) or Trading Networks (TN) document types in CentraSite.
 All E-forms in CentraSite	All e-forms in CentraSite. The Name field displays the template name.
 All IS Services in CentraSite	All IS services in CentraSite (including rule services).
 All Portlets in Workspace	All portlets in your workspace.
 All Processes in Workspace	All processes in your workspace.
 All Rule Services in CentraSite	All rule services in CentraSite.
 All Web Services in CentraSite	All Web services in CentraSite.

In addition to the predefined saved searches that webMethods provides, you can create additional saved searches where you define the search criteria. You can delete, rename, and change the criteria for saved searches that you define.

To add a new saved search, you can:

- Create a new saved search where you specify all the criteria.
- Duplicate an existing saved search and edit the criteria if you want to add a saved search that is similar to an existing one.
- Import saved searches that were previously exported.

The Saved Searches view is shown by default in the Process Development perspective. If you close the Saved Searches view and want to reopen it, or if you want to open it in another perspective, click **Window > Show View >  Saved Searches**.

Working with Saved Searches

Creating Saved Searches

Use a saved search to specify detailed search criteria to locate specific assets. When you create a saved search, you can define criteria to locate only assets of a single type. For example, you can define a saved search for portlets or you can define a saved search for processes, but you cannot define a single saved search that locates both portlets and processes. You can save workspace searches, CentraSite searches, or searches of both the workspace and CentraSite.

> To create a saved search

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View >  Saved Searches**
2. In the Saved Searches view, click  **Add Saved Search**.
3. In the Add Saved Search window, type a **Search Name**.
4. Select the type of asset you want the saved search to locate from the **Asset Type** list.
5. For each search criterion you want to use:
 - a. Click **Add**.
 - b. In the Add Query Condition window, click **Asset Field** and select an item. The items that Designer includes in the **Asset Field** list depends on the **Asset Type** you select.
 - c. Complete specifying the condition for the selected **Asset Field**. Note that the search is not case sensitive.
 - d. Click **OK**.
6. In the **Match condition** field, select one of the following:
 - **Any** to have Designer locate assets that match any of the criteria you specified.
 - **All** to have Designer locate assets only if they match all of the criteria that you specified.

Note:

If you select **References** from **Asset Field** and **All** for **Match condition**, Designer finds assets that contain all the specified references, but those references do not necessarily have to be all in one place. For example, if you have a condition to locate all processes that reference processes with names that contain "HR" and also reference processes with a process version that contains "2", Designer might find a process that contains one step

for a referenced process with the name "HR-Tasks" and a process version "3" and another step for a referenced process with the name "ApproveOrder" and a process version "2".

7. In the **Search in** field, select one of the following:
 - **Workspace** to search metadata in the workspace index only.
 - **Workspace & CentraSite** to search metadata in both the workspace index and CentraSite.
8. Click **Save**.

Duplicating Saved Searches

If you want to create a saved search that is similar to an existing one, you can duplicate an existing user-defined saved search and edit its search criteria.

Note:

You cannot edit, rename, or duplicate webMethods predefined saved searches.

➤ To duplicate an existing saved search

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View >  Saved Searches**.
2. In the Saved Searches view, right-click the saved search you want to duplicate and click **Duplicate Saved Search**.
3. In the Copy Saved Search window, specify a **Saved Search Name**.
4. Click **OK**.

You can edit the duplicated saved search to update the search criteria.

Executing Saved Searches

When you execute a saved search, Designer locates assets and/or asset references that match your search criteria and displays the search results in the Search view.

Note:

If you have workspace indexing disabled, you might have assets in your workspace for which metadata does not exist or is incomplete. In this case, the search will not properly locate these assets.

➤ To execute a saved search

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View > Saved Searches.**
2. In the Saved Searches view, right-click the saved search you want to execute and click **Execute Saved Search.**

Tip:

You can also double-click a saved search to execute it.

Renaming Saved Searches

You can rename user-defined saved searches.

Note:

You cannot edit, rename, or duplicate webMethods predefined saved searches.

> To rename a saved search

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View > Saved Searches.**
2. In the Saved Searches view, right-click the saved search you want to rename and click **Rename Saved Search.**
3. In the Rename Saved Searches window, specify a **New Saved Search Name.**
4. Click **OK.**

Editing Saved Searches

You can edit the search criteria for user-defined saved searches.

Note:

You cannot edit, rename, or duplicate webMethods predefined saved searches.

> To edit a saved search

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View > Saved Searches.**
2. In the Saved Searches view, right-click the saved search you want to edit and click **Edit Saved Search.**
3. If you want to rename the Saved Search, update the name in the **Search Name** field.

4. If you want to change the type of assets the saved search locates, select a new asset type from the **Asset Type** list.

Important:

Search criteria are based on **Asset Type**. If you select a new **Asset Type** while editing a saved search, Designer removes all existing search criteria, and you must rebuild the search.

5. To modify an existing search criterion:
 - a. Select the row for the search criterion and click **Edit**.
 - b. In the Edit Query Condition window, update the search condition.
 - c. Click **OK**.
6. To delete a search criterion, select the row for the search criterion and click **Remove**.
7. To add a new search criterion:
 - a. Click **Add**.
 - b. In the Add Query Condition window, click **Asset Field** and select an item. The items that Designer includes in the **Asset Field** list depends on the **Asset Type** you selected.
 - c. Complete specifying the condition for the selected **Asset Field**. Note that the search is not case sensitive.
 - d. Click **OK**.
8. If you want to change whether Designer matches all of the search criteria or any one search criterion, update the **Match condition** field. Select:
 - **Any** to have Designer locate assets that match any of the criteria you specified.
 - **All** to have Designer locate assets only if they match all of the criteria that you specified.

Note:

If you select **References** from **Asset Field** and **All** for **Match condition**, Designer finds assets that contain all the specified references, but those references do not necessarily have to be all in one place. For example, if you have a condition to locate all processes that reference processes with names that contain "HR" and also reference processes with a process version that contains "2", Designer might find a process that contains one step for a referenced process with the name "HR-Tasks" and a process version "3" and another step for a referenced process with the name "ApproveOrder" and a process version "2".

9. In the **Search in** field, select:

- **Workspace** to search metadata in the workspace index only.
- **Workspace & CentraSite** to search metadata in both the workspace index and CentraSite.

10. Click **Save**.

Exporting Saved Searches

You can export all user-defined saved searches to an XML file. You can keep the file as a backup or use it to import the saved searches into another workspace.

> To export all user-defined saved searches

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View >  Saved Searches**.
2. In the Saved Searches view, click ▾ **Menu** (in the upper right corner) and click  **Export Saved Searches**.
3. In the Save As window, select the location where you want to store your saved searches, specify a name for the exported XML file, and save the XML file to your file system.

Importing Saved Searches

You can import user-defined saved searches into Designer from an XML file that has been exported from Designer.

> To import user-defined saved searches from an XML file

1. In Designer, if the Saved Searches view is not visible, open it: **Window > Show View >  Saved Searches**.
2. In the Saved Searches view, click ▾ **Menu** (in the upper right corner) and click  **Import Saved Searches**.
3. In the Open dialog box, browse to and select the XML file that contains the exported saved searches you want to import.
4. Click **Open**.

27 Process Simulation

■ About Process Simulation	498
----------------------------------	-----

About Process Simulation

The Process Simulation feature for Software AG Designer provides a mechanism to simulate processes and see details about how they run. For complete information about working with Process Simulation, see the online help that is installed with the Process Simulation feature, *webMethods BPM Process Simulation Help*.

By simulating a process, or multiple processes, you can save time, energy, and resources that might otherwise be misspent on deploying solutions that do not fit your business needs. Simulation provides the opportunity for testing and tweaking processes in the design phase, before they ever reach production, or even a test environment.

Process simulation enables you to:

- Discover process bottlenecks
- Predict process behavior in multiple scenarios
- Learn about total process cost
- Understand process resource utilization, including consumables and non-consumables
- Compare the behavior (performance, utilization, cost, etc.) of two or more different processes, or of two or more versions of the same process
- Optimize processes
- Visualize how processes actually work

With a greater depth of process knowledge and behavior, you can make informed iterative changes until you reach the desired simulated result, and ultimately deploy the final version of the process to a production environment.

About Simulation Scenarios

Transforming your business process to a simulation requires defining scenarios of attributes for the process. Setting up your scenario for the process model allows you to create a test case for how a process or set of processes will function with particular constraints.

Scenarios are defined by:

- The frequency and the interval of business process instances
- The length of time each step can be actively processing
- Any resources that your process may use to accomplish its steps

About Simulation Resources

Resources can be consumable or non-consumable. Examples of consumable resources are equipment or supplies such as trucks and gasoline. Examples of non-consumable resources would be a salaried

employee or a hired consultant. Resources are defined on a per simulation basis and are available for all processes within a simulation -- meaning that a single pool of a resource is shared.

This allows a more sophisticated comprehension of resource contention across a set of processes in an enterprise. Resources are defined by the units allocated for the resource and the units acquired at a particular process step. A consumable resource acquired at a step is used and removed from the set of allocated resources. A non-consumable resource, such as an office clerk, when acquired is not available by other steps until the processing of that step is completed. This means that if you have only one office clerk and the step that requires the clerk is started, the next process instance that arrives at that step will be queued for processing until the first instance requiring the clerk is completed.

About Simulation Costs

Resources can also have cost attributes:

- a flat usage cost for any access which is not based on time
OR
- a fixed cost per time period
OR
- a variable cost per time period which is associated with non-consumable resources or a variable cost without a time period which is associated with consumable resources.

Some examples of cost types:

- **Consumable resource with variable cost**
- Fuel is an example of a consumable resource with a variable cost. If fuel costs \$3 a gallon, \$3 is its variable cost. Every time a unit (gallon) of fuel is used, a cost of \$3 is applied.
- **Non-consumable resource with variable cost per period**
- An employee paid hourly is an example of a non-consumable resource with a variable cost per period. The employee's hourly pay can be defined as a variable cost of \$30 per hour.
- **Non-consumable resource with fixed cost**
- A salaried employee is an example of a non-consumable resource with a fixed cost. The employee's annual salary can be defined as fixed cost of \$30,000 yearly.

About Defining Simulation Scenarios

By creating scenarios from criteria within your business, you can closely model the behavior, efficiency and results of your process models. Analyzing output from simulations can allow you to:

- Increase service level
- Reduce total process cycle time

- Increase throughput
- Reduce waiting time
- Reduce process cost
- Reduce inventory costs

In support of process simulation, Designer provides the Process Simulation perspective, which includes several views that allow you to configure your simulations and interpret their behavior: the Resources view, the Run Settings view, and the Advanced Run Settings view. The Resources view contains Resource and Cost information; these are displayed in the Properties view for a selected resource. The Run Settings view contains Run Settings and Statistics information. The Advanced Run Settings view contains information about Variables, Historical Data, and Optimization. The Optimization page contains additional pages to configure Objective, Decision Variable Limits, Constraints, and Run Options.

Additional step configuration is supported in the Properties view for various step types. The Schedule, Scenario, Metrics, and Transition Distribution pages allow you the opportunity to finely tune your simulations.

You can create a process simulation directly from an open process, or you can create a new simulation file and then add one or more processes to it. You can create a simulation from an imported XPDL process, and you can create a simulation while importing if the XPDL contains simulation information. See [“Importing XPDL Processes” on page 413](#).

In Designer's Process Simulation preferences, you can set options for recording animation, automatic switching to the Process Simulation perspective, and the types of gauges displayed during an animated process simulation. The Process Simulation control panel provides additional options for viewing a running simulation, in real time or playback mode.

After you have run a process simulation, you can generate a Process Simulation report that captures detailed information about the scenario simulated, and whose data allows you to create charts, graphs, and other materials based on the results of the simulation.

Note: Designer does not extract metadata for process simulations. They do not have references or dependencies. You cannot publish or retract process simulation metadata.