

ActiveBPEL[®] Enterprise Administration Console Help

Version 4.1

Revised September 2007

ActiveBPEL Administration Console Help

Copyright © 2004-2007 Active Endpoints, Inc.

Printed in the United States of America

ActiveBPEL is a registered trademark of Active Endpoints, Inc.

Contents

ActiveBPEL[®] Enterprise Administration Console Help 1

Chapter 1: ActiveBPEL[®] Enterprise Administration Console Overview

- Prerequisites for Running the Administration Console 7
- Options for Different Application Servers 7
- Understanding the Properties of an Invoked Service 7
- Engine Settings and Process Overrides 8

Chapter 2: Home

Chapter 3: Engine

- Engine Properties 11
- URN Mappings 16
- Function Contexts 18
- Server Configuration (Tomcat) 21
- Alerts 21
- Monitoring Properties 22
- Cluster Configuration (WebSphere/WebLogic) 26
- License 26
- Monitoring 27
- Storage 28
- Version Detail 32

Chapter 4: Extended Services

- Email Service 33
- Identity Service 34
- Creating a Trusted Keystore File 38
- Task Manager (ActiveBPEL for People Inbox) 39

Chapter 5: Deployment

- Deploy BPR 41

Deployment Logs	42
Deployed Processes	43
Deployed Process Detail	44
Deployed Process Version Detail	45
Deployed Services	46
Indexed Properties	47
Partner Definitions	48
Resource Catalog	48

Chapter 6: Process Status

Active Processes	51
Using Selection Filters for Active Processes	51
Alarm Queue	54
Receive Queue	54

Chapter 7: Process ID and Process Details

Using the Process Details Page	58
Using the Process Details Graphic View	59
Using the Process Details Outline View	61
Inspecting Where and Why an Activity Faulted	64
Working with Variable Attachments	65

Chapter 8: Process Versions

Process Version Life Cycles	69
Process Version Persistence Type	71
Exception Management Type	72
Process Instance Retention Days	72

Chapter 9: ActiveBPEL Enterprise Clusters (Application Server Specific)

Chapter 10: Process Exception Management

Appendix A: BPEL Standard Faults

Appendix B: ActiveBPEL Custom Faults

1

ActiveBPEL® Enterprise Administration Console Overview

What's in this chapter

- [Prerequisites for Running the Administration Console](#)
 - [Options for Different Application Servers](#)
 - [Understanding the Properties of an Invoked Service](#)
-

The ActiveBPEL Enterprise Administration Console allows you to manage and configure the ActiveBPEL engine and the artifacts that are deployed into it.

The Administration Console provides ways to deploy, select, inspect, and correct processes and related endpoint references.

You can do the following from the Administration Console pages:

- [Home](#)
Start and stop the engine. View number of deployed processes.
- [Engine](#)
Configure engine properties, set up URN/URL maps and custom functions, add an alert system, and manage database storage and licenses.

[Engine Properties](#)

[URN Mappings](#)

[Function Contexts](#)

[Server Configuration \(Tomcat\)](#)

[Alerts](#)

[Monitoring Properties](#)

[Cluster Configuration \(WebSphere/WebLogic\)](#)

[License](#)

[Monitoring](#)

[Storage](#)

[Version Detail](#)

[ActiveBPEL Enterprise Clusters \(Application Server Specific\)](#)

- **[Extended Services](#)**

[Email Service](#)

[Identity Service](#)

[Task Manager \(ActiveBPEL for People Inbox\)](#)

- **[Deployment](#)**

Deploy processes, view deployment and process details, manage process versions.

[Deploy BPR](#)

[Deployment Logs](#)

[Deployed Processes](#)

[Process Versions](#)

[Deployed Process Detail](#)

[Deployed Process Version Detail](#)

[Deployed Services](#)

[Indexed Properties](#)

[Partner Definitions](#)

[Resource Catalog](#)

- **[Process Status](#)**

View active processes and their status, waiting alarms, and waiting receives. Select an active process using extensive filters and expressions. Add/remove variable attachments for a running process. Perform process exception management.

[Active Processes](#) and [Using Selection Filters for Active Processes](#)

[Alarm Queue](#)

[Receive Queue](#)

- **[Process ID and Process Details](#)**

[Process Exception Management](#)

Prerequisites for Running the Administration Console

Before running the Administration Console in your browser, be sure to complete configuration and database setup by following the instructions in *ActiveBPEL Enterprise Installation, Configuration, and Deployment Guide* located in the *ActiveBPEL product installation\doc* folder. Open *index.html* to view this guide.

If your license is not valid, you can open the Administration Console, and add a new license from the License page.

To start the Administration Console, start up your application server, and then type the following into your browser:

```
http://localhost:8080/BpelAdmin
```

Change the port number if 8080 is already in use on the computer.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#).

Options for Different Application Servers

ActiveBPEL Enterprise works with a variety of application servers, and some of the settings in the Administration Console do not affect all application servers.

Note that special features for your application server are called out separately in online help.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#).

Understanding the Properties of an Invoked Service

ActiveBPEL server can invoke the services within a BPEL process using a variety of addressing options. In most cases, these options are defined for each partner role in the Process Deployment Descriptor (PDD) file that is deployed with a BPEL process. Alternately, addressing may occur dynamically within the process. In addition, some addressing options, such as service retry and security policies may be specified in a deployed BPEL file.

You can view the address properties of an invoked service on the Deployed Process Version Detail page. Under the Linkage column for a partner role, select the Static link to see the details. See [Deployed Process Version Detail](#).

For a detailed discussion of addressing options, refer to the *ActiveBPEL Enterprise Server User's Guide*.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#).

Engine Settings and Process Overrides

There are several default engine settings that can be overridden by an individual process version, but some cannot be overridden. There is one process setting that does not have a default engine setting

Engine Setting	Process override?
Validate input/output messages	no
Logging enabled	no
Replace existing resources on deployment	yes
Suspend process on uncaught fault	yes
Process instance retention days	yes

Engine Default?	Process setting
no	Persistence Type

2 Home

The Home page of the ActiveBPEL Administration Console provides an overview of the engine that executes BPEL processes. It contains the following items.

Item	Description
Date Started	Engine start date
Deployed Processes	Number of business processes (.bpel files) currently stored in the database
Description	Engine configuration. This is the application server platform supported for this engine.
Status (or Cluster Status)	Statuses for the ActiveBPEL Enterprise engine are Running and Stopped. Additional database messages are included. Select <i>Storage</i> to see more detailed information regarding the database.
Monitoring Level (or Cluster Monitoring Level)	Level indicates a monitoring severity; that is, whether a warning or error is detected. If there is no monitoring set up, or if the engine is running normally, the level is Normal. Levels include Normal, Warning, and Error.
Version	Engine version number

Stopping and Starting the Engine

- **ActiveBPEL Enterprise for Apache Tomcat.** The engine starts when you start Tomcat and stops when you shut down Tomcat. Select **Stop Engine** when you need to stop all running processes. Select **Start Engine** to change the engine status to *Running*.
- **ActiveBPEL Enterprise.** The engine starts when you start your application server. The engine stops when you shut down the application server. Select **Stop Engine** when you need to stop all running processes. Select **Start Engine** to change the engine status to *Running*.

License Warning

The License Warning field displays messages to help you stay in compliance with your ActiveBPEL license agreements. ActiveBPEL can verify that the number of installed licenses matches the number of CPUs on your server.

To check for license/CPU compliance in ActiveBPEL Enterprise, see [ActiveBPEL Enterprise Clusters \(Application Server Specific\)](#).

To check license/CPU compliance in ActiveBPEL Enterprise for Apache Tomcat:

- 1** Select **Configuration** in the Administration Console navigation bar.
- 2** In the Server Configuration section, type in the number of CPUs on the server where ActiveBPEL is installed, and select **Update**.
- 3** Select **Home** in the Administration Console navigation bar. If the number of CPUs on the server exceeds the number of licenses installed, a warning is displayed.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

3 Engine

What's in this chapter

- [Engine Properties](#)
 - [URN Mappings](#)
 - [Function Contexts](#)
 - [Server Configuration \(Tomcat\)](#)
 - [Alerts](#)
 - [Monitoring Properties](#)
 - [Cluster Configuration \(WebSphere/WebLogic\)](#)
 - [License](#)
 - [Monitoring](#)
 - [Storage](#)
 - [Version Detail](#)
 - [ActiveBPEL Enterprise Clusters \(Application Server Specific\)](#)
-

Engine Properties

On the **Engine Properties** tab of the Engine **Configuration** page, you can make configuration changes without stopping and restarting the engine. When you make a change and select **Update**, the changes are in effect immediately as well as persisted to the engine configuration file. The changes are also persisted to the database and propagated to other engines in the cluster.

View and update engine configuration settings as shown in the following table. Note that some of these properties are the same as the ActiveBPEL Designer Simulation preferences.

Property Name

Description

Auto create target path for Copy/To

Applies only to processes that are validated against the BPEL4WS 1.1 specification. For WS-BPEL 2.0 processes, this property can be added as an extension on a per process basis. Refer to Extensions help topic in *ActiveBPEL Designer Online Help*.

Determines if ActiveBPEL server is allowed to create a location path for a non-existent node in a complex variable in a process instance document. When an assignment refers to a non-existent node (or to more than one node), the standard BPEL fault, bpws:selectionFailure, must be thrown, according to the BPEL specification.

Enabling this option allows selections to be created on-the-fly. This means an assign copy TO operation can refer to a non-existent node and assign a value to it. This option is disabled by default.

Disable bpws:selectionFailure fault

Applies only to processes that are validated against the BPEL4WS 1.1 specification. For WS-BPEL 2.0 processes, this property can be added as an extension on a per process basis. Refer to Extensions help topic in *ActiveBPEL Designer Online Help*.

Enabling this option allows a null value to be returned from a function or assignment that contains an XPath query string. You can enable this to override XPath behavior, for cases that handle data samples with optional elements.

By default, this option is not enabled, and if the query string returns an empty selection from an assign copy FROM, the process throws a bpws:selectionFailure fault, which is the standard response described in the BPEL4WS specification.

Property Name

Logging Level

Description

By default the ActiveBPEL engine does not generate an execution log for running processes. Logging is turned off to enhance engine performance. You can enable this setting, and then view or download an execution log for a running or completed process. An execution log provides start/end times for activity execution and helps you troubleshoot faulted processes. The logging levels are:

- **None.** (default)
- **Full.** All execution statements are logged, including the *Will Not Execute* statements for deadpath activities. For example, all fault handling statements that are not executed are logged.
- **Execution.** All execution statements are logged, except for *Will Not Execute* statements. Using this setting can greatly decrease the size of the log file.

Replace existing resources on deployment

Overwrites the current WSDL definition and other resources such as schema files.

By default, resources are not replaced. However, if you enable this setting, ActiveBPEL allows you to replace a WSDL definition or schema file currently in cache without restarting the server. You can deploy a new version of a BPR file containing updated resources.

BPEL developers who are testing and modifying processes and WSDL definitions may find this option useful.

Note that you can add this setting to an individual BPR file in the ActiveBPEL Designer Export Wizard. The Deployment Log shows whether or not the resources are being replaced.

Property Name

Suspend process on uncaught fault

Description

According to the WS-BPEL 2.0 specification, a process with an uncaught fault terminates.

Enable this option to suspend all processes on an uncaught fault to put them in a suspended-faulting state. You can then perform process exception management on the faulting process followed by retrying or completing the faulting activity or scope.

An individual process can override this setting with an entry in the PDD file. See [Exception Management Type](#).

See [Process Exception Management](#).

Validate input/output messages against schema

Validates the data loaded into process variables against the WSDL schema.

Enable this option to validate data before execution starts. Disable this option for faster execution. This option is enabled by default.

Deployment Plan Cache

A deployment plan corresponds to each deployed version of a process, including associated disposition of running processes. Process versions that are active can be cached for better engine performance. The default number of plans that are cached is 100. For details regarding versions, see [Process Versions](#).

Process Count

Specifies the maximum number of processes in memory. The default number is 50. Specifying 0 indicates no limit, but is not recommended.

Process Idle Timeout

Specifies the number of seconds to wait until process state information is written to the database during idle processing times, such as waiting for a reply from an invoked service. You can increase the timeout value to enhance engine performance. You can decrease the value to ensure the full process state is always in the database. Doing so avoids potential process recovery time in the event of a server failure. The default is 10 seconds.

Resource Cache

The number of WSDL files and other resources in stored cache. The default is 100. Modifying the cache size may improve engine performance. A value of -1 means unlimited caching, but is not recommended.

Property Name

Unmatched Correlated Receive Timeout

Description

Sets the amount of time to wait (in seconds) for a correlated message to be matched to a receive, onMessage, or onEvent activity, in the case that the message arrives before the activity becomes active. If this value is exceeded, a message is discarded so that the process can complete normally. The default is 300 seconds. Specifying 0 indicates that unmatched correlated messages are immediately discarded.

Web Service Timeout

For performance reasons, a reply activity matching a receive, as well as synchronous invokes, are timed out if they do not execute within 10 minutes. If you are receiving timeout errors, you can specify a greater amount of time to wait before a process is timed out due to a reply or synchronous invoke activity not executing within 10 minutes. The default is 600 seconds.

Work Manager Thread Pool Min

Set the minimum number of execution threads the engine allocates for its Work Manager. The default is 10. A simple rule of thumb is to have enough work threads to run the number of processes plus the number of simultaneous invokes that processes may execute.

Note: This property does not appear in the Administration Console if ActiveBPEL server is configured to use an application server Work Manager.

Work Manager Thread Pool Max

Set the maximum number of execution threads the engine can spawn simultaneously. The default is 50. A value of -1 means that there is no maximum number of threads.

Note: This property does not appear in the Administration Console if ActiveBPEL server is configured to use an application server Work Manager.

Property Name

Work Manager Threads For Alarms Max

Description

Set the maximum number of threads the engine will use from the work manager to dispatch work scheduled by an alarm in a process. If there are 100's of alarms firing concurrently, it is possible that all of the threads in the work manager could be used just to dispatch the alarms. If you experience performance issues or deadlocks due to all of the threads being used by the alarm manager, you can increase this value. The default is 5.

Work Manager Threads Per Process Max

Set the maximum number of execution threads the engine can spawn simultaneously for an individual process. The default is 10.

Go back to [Engine](#).

URN Mappings

On the **URN Mappings** tab of the Engine **Configuration** page, you can assign a physical address to a universal resource name (URN).

URN mappings provide a flexible and dynamic way to define target endpoint references. Use URN mappings to specify the physical address of a partner link endpoint reference instead of using the address specified in a process deployment descriptor (.pdd) file or WSDL file. By mapping an URN to an URL, you do not have to rely on invoking a statically defined endpoint address. URN mappings give you flexibility, for example, to deploy the same BPR files for testing and production environments.

Instead of using the default invocation, you can specify a logical or physical address for a static endpoint reference in the .pdd file. If you specify a logical address, or URN, you can then map the URN to the physical address in the URN Mappings page of the Administration Console. If you specify a URL, you can replace the URL by mapping it to a different URL.

The following example illustrates one type of URN to URL mapping:

```
urn:localhost = http://localhost:8080/active-bpel/services/  
${urn.3}
```

This mapping might be used when a process is deployed with the following partner link address information:

```
<partnerLink name="assessor">
```



```

<partnerRole endpointReference="static"
  invokeHandler="default:Address">
  <wsa:EndpointReference xmlns:assessor="http://
    tempuri.org/services/loanassessor">
    <wsa:Address>urn:localhost:AssessRisk</wsa:Address>
    <wsa:ServiceName PortName=
      "SOAPPort">assessor:LoanAssessor</wsa:ServiceName>
  </partnerRole>
</partnerLink>

```

The ActiveBPEL invocation framework resolves the URN as follows:

urn:localhost:AssessRisk = http://localhost:8080/active-bpel/services/AssessRisk

Here are some ways you can map URNs to URLs. Note that each segment of the URN is separated by a colon. This means you can use a variable, such as `${urn.4}` shown in the third example below, to indicate a replaceable token in the fourth segment.

URN	URL
urnSegment1:urnSegment2	http://localhost:8080/active-bpel/services/MyService
http://ServerA:8080/active-bpel/services/MyService	http://ServerB:8081/active-bpel/services/MyService
urn:localhost:service	http://localhost:\${AE-NODE1-PORT}/activebpel/services/\${urn.4}

The last example in the table above shows how you can use variable substitution in an URL.

The URL values can optionally contain variables. The variables can be environment variables accessible through `java.lang.System.getProperties()` or a segment from the URN itself. The Apache Ant style variable declaration of `${property}` is used to identify a property within the URL. Segments from the input URN value can be referenced by using a special property naming convention of `${urn.offset}` where *offset* is a one-based offset identifying the segment from the input URN value to use for substitution.

The URL in the mapping above contains two variables. The `${AE-NODE1-PORT}` variable pulls the port number from an environment variable. This variable would need to be set as a `-D` parameter on the Java runtime environment (e.g., `java -D${AE-NODE1-PORT}=8080 . . .`) or populated externally to the ActiveBPEL server.

The `${urn.4}` variable in the above mapping references the fourth segment from the input URN value. Notice that the URN contains only three segments. The URN in the `.pdd` file should contain at least one other segment. A sample URN might be:

```
urn:localhost:service:StoreService.
```

The value of the fourth segment of this URN is `StoreService`. The resulting URL is:

```
http://localhost:8080/activebpel/services/StoreService/.
```

Updating or Deleting a URN Mapping

To update a URN mapping, select the URN. The URN and URL values appear in the text boxes where you can edit them and select **Update**. Editing the URN results in a new URN mapping. It does not update the existing one. Only the URL can be updated.

To delete a mapping, select the check box next to the mapping and select **Delete**.

Go back to [Engine](#).

Function Contexts

On the **Function Contexts** tab of the Engine **Configuration** page, you can add custom function information.

BPEL processes may contain custom functions that are used within XPath or other expression languages. ActiveBPEL provides a `FunctionContext` interface for implementation of custom functions. By using the `FunctionContext` interface, new or different functions may be installed and made available to the ActiveBPEL XPath (or another) expression writer.

If you already have custom functions implemented with a different interface, such as the `jaxen FunctionContext` interface, you can use them in your BPEL process.

If you have not yet written custom functions, you can download a BPEL process example from the Samples page of <http://www.active-endpoints.com>. The Custom

Function sample includes all the necessary source files for using custom function in ActiveBPEL Designer and on the server.

Compiling Your Custom Function with Function Context Classpath References

To implement the **Java-based** FunctionContext interface, do the following:

- 1 In your ActiveBPEL Designer installation, locate the following folder:
ActiveBPEL Designer product folder\Server\ActiveBPEL_Tomcat\shared\lib
- 2 In the \lib folder, locate *ae_rtbpel.jar*.
- 3 The class file in *ae_rtbpel.jar* you need in order to implement the FunctionContext interface is:
org.activebpel.rt.bpel.function.IAeFunctionContext

To implement the **.NET** FunctionContext interface, do the following:

- 1 In your ActiveBPEL for .NET installation, locate the following folder:
ActiveBPEL for .NET\bin
- 2 In the \bin folder, locate *AeEngineClient.dll*.
- 3 The package you need in order to implement the FunctionContext interface is *ActiveBPEL.Function*.

Adding Custom Functions to ActiveBPEL Server

You add custom function details to make the functions known to the engine.

Select one of the following procedures:

- Custom functions for Java-based ActiveBPEL servers
- Custom functions for ActiveBPEL Enterprise for .NET

Custom Functions for Java-based ActiveBPEL Servers

As noted in the procedure below, you can specify an absolute classpath location for the function or use a system property to indicate a location that any server in a cluster can point to. You define the system property as appropriate for your application server.

- 1 From the Engine Configuration page, select **Function Contexts**.
- 2 In the **Add Function Context Details** section, do the following:

- Type in a **Name** for the custom function. The name appears in the Custom Function list.
- Type in a **Namespace**. Use the namespace identifying the function. The namespace is declared in BPEL processes using the function.
- Type in the fully qualified **Class** name of the container file that implements the custom function.
- Type in a **Classpath** location for the custom function folder, zip or jar file. The classpath can be an absolute path, such as `C:\jakarta-tomcat-5.0.28\shared\lib\MyCustomFunction.jar`, or can be a system property. For example, a classpath using a JBoss system property value might look like: `${jboss.home}\MyCustomFunction.jar`.

3 Select **Add Context**.

Custom Functions for ActiveBPEL Enterprise for .NET

1 Add your custom function assembly to the following location:

[ActiveBPEL for .NET installation folder\bin]

2 From the Engine Configuration page, select **Function Contexts**.

3 In the **Add Function Context Details** section, do the following:

- Type in a **Name** for the custom function. The name appears in the Custom Function list.
- Type in a **Namespace**. Use the namespace identifying the function. The namespace is declared in BPEL processes using the function.
- Type in the fully qualified **Class** name of the container file that implements the custom function.
- Type in an **Assembly Location** for the custom function assembly. The file is expected in the bin folder, so you can just type, for example:
`MyCustomFunction.dll`.

4 Select **Add Context**.

ActiveBPEL server validates the function details and ensures that a class loader can load the class files.

If an error is reported, ensure that you have a valid class name and classpath/assembly location.

For each successfully added context, the name, namespace, and class of the function is displayed in a list. You can delete any function that you no longer need, if you delete the associated processes.

Go back to [Engine](#).

Server Configuration (Tomcat)

The **Server Config** tab of the Administration Console's **Configuration** page is for Apache Tomcat application server.

To help you stay in compliance with your ActiveBPEL license agreements, ActiveBPEL can verify that the number of installed licenses matches the number of CPUs on your server. For details, see *License Warning* in the [Home](#) topic.

Go back to [Engine](#).

Alerts

On the **Alerts** tab of the Engine **Configuration** page, you can add the name of the service you want to run when processes are faulting.

You can add the service name of a BPEL process that is designed to send out an alert when a certain process state is encountered, currently *suspended* or *faulting*. When the state occurs, the ActiveBPEL server instantiates the alert service, which can then invoke some action, such as notifying an administrator that a process is faulting.

To add a service, type in the **Service** name, and select **Update**. The service name is the My Role partner link service, identified in the PDD file deployed with the BPEL process to be used as the alert service. You can find this name by looking on the Deployed Services page.

After you add the service, select the process name to view process version details.

For details on how to create an alert service, see the topic *BPEL Processes as Engine Services* in the *ActiveBPEL Enterprise Server User's Guide*.

To delete an Alert service, select the Service Name, and press the **Update** key.

Go back to [Engine](#).

Monitoring Properties

On the **Monitoring** tab of the Engine **Configuration** page, you can select engine properties to monitor. For each property, you can provide a statistic and threshold that, when reached, alerts you to a warning or error condition.

You can decide what the frequency and interval of monitoring periods should be, as described in the following table.

Threshold Interval	Period for collecting and aggregating statistics. The default is five minutes.
Evaluation Frequency	Number of times during the threshold interval the statistics are evaluated. For example, if the threshold interval is five minutes, and the evaluation frequency is five times, then statistics are collected and aggregated once a minute during every five-minute period. The default is five times.
Maximum Trouble Items	Number of error/warning items per engine to display on the Monitoring page of the Administration Console

Monitor Alert Service

Add the name of the service you want to run when errors and/or warnings occur for one or more monitored properties.

When errors occur, the ActiveBPEL server instantiates the alert service, which can then invoke some action, such as notifying an administrator that a monitored property has an error condition. The service can also monitor engine status (running or stopped).

To add a service, type in the **Service** name, and select **Update**.

The service name is the My Role partner link service, identified in the PDD file deployed with the BPEL process to be used as the alert service. You can find this name by looking on the Deployed Services page.

After you add the service, select **View Details** to view the BPEL process.

For details on how to create an alert service, see the topic *BPEL Processes as Engine Services* in the *ActiveBPEL Enterprise Server User's Guide*

The properties you can monitor are as follows:

Property	Description	Evaluating This Property
Faulted/faulting processes (count)	Number of processes that end in a faulted state or are suspended due to an uncaught fault.	You may have an expectation that processes running on this engine should not fault, and you want to be notified if they do. Also see Alerts for a description of the Alert Service that enables you to configure a service for notification of faulted processes.

Property

Process cache efficiency (percent)

Description

The engine configuration contains a count of the maximum number of processes which can be kept in memory before they are forced into storage. A process is cached in memory if it is currently executing some logic or if it is quiescent but is being cached in anticipation of receiving another message, alarm, or response to an invoke. This property reports the percentage of processes that are read from memory versus process instances read from storage. For example, the value of 100% indicates that all process reads are coming from the memory cache.

Evaluating This Property

On the Engine Properties tab of the Configuration page, you can set values for Process Count and Process Idle Timeout. The Process Count setting controls the size of the cache. The Process Idle Timeout setting controls how long to keep an idle process in the cache. If the process cache efficiency percentage is low and your processes contain bi-directional invokes or can process multiple inbound messages, then you may benefit from increasing the Process Count and Process Idle Timeout. This will help to keep processes in memory. However, if your processes are long-running in nature and receive messages only periodically, then a low process cache efficiency percentage is not necessarily a problem.

Time to obtain process (ms)

The time it takes to obtain a process is useful to determine if this operation is trending significantly higher under load situations. The monitoring includes maximum and average values. This property includes the time it takes to acquire a lock on the process as well as restore its state from storage if necessary.

This property works in conjunction with process cache efficiency.

Work manager work start delay (ms)

The time it takes between scheduling of a work item request and the actual start of work can help in tuning of the work manager pool. If the time delay is trending upwards, this can indicate that there are not enough threads available to handle the amount of work. The monitoring includes maximum and average values.

If you are using the default Work Manager then the size of the work pool can be configured on the Engine Properties tab of the Configuration page. If you are using a Work Manager implementation provided by an application server, then the size of the pool and the priorities of its threads should be configurable in your application server administration console.

Property	Description	Evaluating This Property
Database connection acquisition time (ms)	Tracks the amount of wait time to get a connection from the datasource. An excessively long wait may indicate signs of trouble with the size of the connection pool. The monitoring includes maximum and average values.	The ActiveBPEL storage layer does not do any connection pooling. It relies on the storage implementation (usually a <code>javax.sql.DataSource</code>) to pool connections. Consult your application server or database documentation to address any issues with poor performance related to the connection pool size or connection acquisition time.
Discarded Unmatched Correlated Receives (count)	When a message with correlation properties fails to route to a running process instance and is not able to create a new process instance, the engine will keep trying to dispatch the message for the configured amount of time. There is a limit to the number of such unmatched messages that the engine will retry. This property tracks the number of messages that were discarded due to the buffer of unmatched messages being full.	The unmatched correlated receive timeout on the Engine Properties tab of the Configuration page controls the amount of time that the engine will keep the unmatched message queuing until it is routed to a process instance. The pool of unmatched receives may fill up if this timeout is too high. However, such a problem may be an issue with process design as opposed to the timeout or buffer size.

Selecting Engine Properties to Monitor

You can indicate the engine properties you want to monitor as follows:

- 1 On the Administration Console Configuration page, select the Monitoring tab.
- 2 Select **Add Row**.
- 3 In the **Property to Monitor** column, select an engine property.
- 4 In the **Level** column, select a severity: Error or Warning.
- 5 In the **Statistic** column, select a statistic. Sometimes there is only one choice that is appropriate for the property.
- 6 In the **Op** column, select a relational operator for the threshold.
- 7 In the **Threshold**, add a non-negative integer, appropriate for the evaluation.
- 8 Add as many rows as you want, and select **Update**.

After you set up the properties, go to the [Monitoring](#) page to view the results.

Go back to [Engine](#).

Cluster Configuration (WebSphere/WebLogic)

On the **Cluster** tab of the Engine **Configuration** page, you can set properties applicable for WebSphere and WebLogic engine clusters.

WebSphere Properties

You can add the name of the WebSphere application server **JAAS login** for cluster communications.

If your WebSphere application server is running with global security enabled, you can provide or update the JAAS username. ActiveBPEL uses this name when performing cluster communications. You can choose from the following:

- **ActiveBPELProvidedUser.** JAAS user created with Monitor rights
- **ActiveBPELIdentityAssertion.** JAAS user created with Monitor rights and uses the user name and password of the requester to reassert the credentials
- **JAAS Custom Login.** JAAS user that is available on the WebSphere application server that has at least Monitor rights

In WebSphere, you can also set a **Communications Time Out** value for ActiveBPEL to send and receive communications to and from engines in the cluster. The default value is 180 seconds.

WebSphere and WebLogic Property

You can set a **Membership Update Interval** to indicate how often you want to poll engines in the cluster to ensure they are active. The default value is 30 seconds.

Go back to [Engine](#).

License

The License page displays the installed licenses for your ActiveBPEL server product. You can re-install, add, and remove licenses as needed.

Note that you must have at least one license installed in order to run the engine.

Do the following on this page:

- To add or re-install a license, select **Add**. In the window that opens, paste in the license key that was emailed from Active Endpoints. Select the **Add License** button in the window.
- To remove a license, select **Remove** for one of the installed licenses
- To view details for one license, select a serial number under Installed Licenses. The **License Detail** page shows specifics for the organization, application server, license type, and number of licensed CPUs.

Go back to [Engine](#).

Monitoring

Use the Monitoring page to view statistics on engine properties. ActiveBPEL collects engine statistics and then aggregates them by intervals. If desired, you can configure a threshold interval and an error/warning level for the engine properties listed. If you configure these settings, you will see additional signals displayed on this page, such as an eyeglass icon indicating a property is being watched.

During a collection interval, ActiveBPEL maintains statistics for each configured property value, including the maximum, average, and total count values. In addition, historical statistics are collected. Historical statistics include minimum/maximum values for the intervals which have been recorded. Statistics are collected in-memory.

In a clustered environment, the monitoring report's top level is an aggregate of values from all engines in the cluster. Below the aggregate information is a report for each individual engine in the cluster. In a clustered environment, statistical information is relayed from one node to the next to allow for aggregation.

Select **Reset** to clear statistics. For example, if you correct an error condition, you can reset the statistics collection starting with zero values to see if performance improves after tuning engine configuration. Once all errors and warnings have been cleared, the engine monitoring level is reset to Normal.

Monitoring indicators include the following:

- **Eyeglass icon.** Monitoring details for the property have been configured on the Monitoring tab of the Configuration page, and if no red or yellow highlights appear in the row, the collected statistics are within normal range.

- **Yellow highlight.** A warning-level condition has occurred for the highlighted statistic during the threshold interval.
- **Red highlight.** An error-level condition has occurred for the highlighted statistic during the threshold interval.

Monitoring Report

During each threshold interval, if an error or warning occurs, it is reported at the end of an evaluation period. For example, if your threshold interval is set to five minutes, and the evaluation frequency is set to five times per interval, the first trouble item may appear after one minute has elapsed, but will not be reported more than once per threshold interval. If competing threshold levels occur during same evaluation cycle, the highest severity will be reported.

Each time you refresh your browser or open the Monitoring page, monitoring statistics are updated. The report, however, is updated only at the end of an evaluation period.

The list of trouble items shows the error/warning level, time, property name, and configuration details for each item reported. The following is an example of a report.

Monitoring Level:	Error
⊗	[8/21/07 11:36 AM] Faulted/Faulting Processes (count): Count=3 (> 2)
⚠	[8/21/07 11:34 AM] Process cache efficiency (percent): Avg=94 (< 96)
⊗	[8/21/07 11:34 AM] Faulted/Faulting Processes (count): Count=2 (> 0)

For details on setting up monitored engine properties, see [Monitoring Properties](#).

Go back to [Engine](#).

Storage

The ActiveBPEL server engine includes persistent storage based on the database settings you configured during installation. The engine supports both relational databases and XML databases. You must configure one database before running the engine. For configuration instructions, see the *ActiveBPEL Enterprise Installation, Configuration, and Deployment Guide*. To view this guide, open the *index.html* file in the \doc folder of your installation folder.

The Storage page displays database configuration properties and allows you to maintain the database.

For relational databases, the displayed properties are:

JNDI Location (Java-based servers)	The Java Naming and Directory Interface (JNDI) context that specifies where to look for the database. For example, <code>jdbc/ActiveBPEL</code>
Database Type	The type such as <code>mysql</code>
User Name (Java-based servers)	Username, if required for ActiveBPEL server's access to the database
Connection String (.NET)	Parameters for access to the database. For example, <code>Server=localhost;Port=3306;Database=ActiveBPEL;</code>

Note: If you need to change the user name and password for database access, we recommend that you re-run the ActiveBPEL Configure and Deploy utility. This utility, located in your ActiveBPEL Enterprise installation folder, allows you to modify the database user name and password without modifying other settings.

For XML databases, the displayed properties are:

DatabaseType	The type such as <code>tamino</code>
Tamino URL	The path to the tamino database <code>http://host/tamino</code> where <code>host</code> is the name of the computer on which the Tamino Server resides. You can specify <code>localhost</code> for your local computer, or a path such as <code>yourpc.ourcompany.com</code> for a remote computer.
Database Name	Name that was entered in the ActiveBPEL Installation, Configuration, and Deployment wizard
Collection	For tamino databases, this is the name that matches the collection name specified in the database schema file (.tsd file). If the collection name was modified in the Installation, Configuration, and Deployment wizard, it must also be modified in the schema file that ActiveBPEL server provides for database set up.
Connection Pool Size	Number of simultaneous connections allowed to the database. The default size is 30.
Domain	Network domain for host computer
Username	Username if required for ActiveBPEL server's access to the database
Password	Password associated with the Username

Manual and Scheduled Database Maintenance

The ActiveBPEL database stores completed processes and deployment logs. You can delete old processes and logs, as desired, either manually or on an automated schedule.

Note: The Process Deployment Descriptor (PDD) file contains an option for disabling storage for a completed process. For details see the *ActiveBPEL Server User's Guide*.

Manual Maintenance

Use the **Manual** tab on the Storage page to delete files from the database as follows.

Deleting Completed and Faulted Processes

- 1 Enter a date in the **Completed/Faulted before** field.
- 2 Select **Delete**. The number of matching processes is displayed.
- 3 Select OK.

Deleting Inactive Plans

A *plan* consists of a process version plus the associated disposition of running processes. An *inactive plan* refers to a process version and associated processes that either have reached their expiration date or have been manually expired. If you have deleted completed processes from the database, you can delete the plan associated with those processes.

This means that the process version associated with the plan will no longer be displayed on the Deployed Processes page. The associated WSDL and other resources are not deleted, nor is any Partner Definition, since these files may be associated with other plans.

If a plan is associated with a subprocess, you cannot delete it until the main process is deleted. A *subprocess* is a BPEL process that is invoked by another process.

Deleting Deployment Logs

- 1 Select the type of log in **Log Contents**.
- 2 If desired, enter dates in the **Deployed between** fields.
- 3 Select **Delete**. The number of matching logs is displayed.
- 4 Select OK.

Scheduled Maintenance

Use the **Scheduled** tab on the Storage page to specify when and how often to delete items from the database.

Maintenance Schedule

Frequency

- **None** resets the engine for no scheduled maintenance
- **Daily, Weekly, or Monthly** sets the frequency schedule

Every

- For **Weekly**, select one or more days of the week. For example, select Monday to delete database items every Monday. Select Monday and Thursday to delete every Monday and Thursday.
- For **Monthly**, select the first, second, third, fourth, or last day of the month. For example, select Last Monday to delete items on the last Monday of the month.

Start Time

Type in a time of day, or select the Date Chooser to enter a time. In the Date Chooser, select Now to enter the current time.

Plan and Process Delete Options

Delete Inactive Plans

Select this setting to delete old process versions. (For an explanation of process versions, see the topic above, *Deleting Inactive Plans*.) If you do not select this setting, only completed/faulted processes can be deleted.

Delete Completed and Faulted Processes

Select this setting to delete old processes. If you do not select this setting, only inactive plans can be deleted.

Default Process Retention Days

Enter a number of days to retain completed and faulted processes in the database before deleting them. The default is one day, indicating all processes one day old can be deleted on the next scheduled maintenance.

Processes are deleted on the next scheduled time after the retention days duration. For example, if process retention days is set to 30, and scheduled deletions occur every Friday, the processes that have been in the database at least 30 days are deleted each Friday.

This setting can be overridden for individual processes. See [Process Instance Retention Days](#) for details.

Delete Deployment Logs

Delete all deployment logs.

After you set up a maintenance schedule, select **Update** to save your settings.

Select **Run Now** to run the current schedule immediately, in addition to the regular schedule, or to select a different schedule and run it. Any changes you make to the schedule do not take effect unless you select Update.

Go back to [Engine](#).

Version Detail

The Version Detail page shows the version number and build date of the ActiveBPEL Server engine libraries. This information may be useful for troubleshooting purposes.

Go back to [Engine](#).

4

Extended Services

What's in this chapter

- [Email Service](#)
 - [Identity Service](#)
 - [Task Manager \(ActiveBPEL for People Inbox\)](#)
-

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

Email Service

An email service provides a way for a BPEL process to send email messages. The messages are routed from the mail server and username you configure on the Email Service page.

Any BPEL process that implements email-based activities imports the WSDL provided with the *ActiveBPEL Designer* product. The name of the WSDL is *email.wsdl*. Refer to the *ActiveBPEL Designer* documentation for details about this WSDL. *ActiveBPEL for People* developers can add email reminders to the People activity tasks they create without specifically importing this WSDL.

As an example of using email, a BPEL process can be triggered to send an email to an administrator when other running processes are suspended on an uncaught fault.

Recipients of an email may reply to the username you add on this page.

View and update email service settings as shown in the following table.

Enable	Add a checkmark to use the email service. Initially the service is disabled since it is not configured and ready for use. Configure the remaining settings, and then enable the service.
Host	Enter the email server's DNS name such as mail.my-domain-name.com or IP address such as 192.168.1.1.
Port	Enter the port to use for communications between the ActiveBPEL server and the email server. The default value is 25.

From Address	Add the email address that appears in the FROM field of an email. For example, <code>no-reply@example.com</code> .
Username	Enter the name used to login to your email server, usually the account name. This name is the sender of all email sent in BPEL processes. For example, <code>ActiveBPEL.notification@my-domain-name.com</code> .
Password	Enter the password for the above email address, and confirm it.

Testing the Current Configuration

Select **Test Settings**, enter an email address and select **Send**.

Verify that the email was sent.

When your settings are correct, select **Update**.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

Identity Service

An identity service provides a way for a BPEL process to look up users and groups in an enterprise directory service. The ActiveBPEL Enterprise identity service is based on either Lightweight Directory Access Protocol (LDAP) or file-based support.

By providing the communication details for look-up access to your directory service, you can do the following:

- Run BPEL processes that implement identity-based activities. When a process runs, the person or group specified in the process is looked up in your directory service.
- For the *ActiveBPEL for People* product, enable administrators and users to assign and forward tasks to people and groups

Note: If you are setting up an identity service for use in the *ActiveBPEL for People Inbox* application, you must add a role (group) name to your group list: *workflowuser*.

Any BPEL process that implements identity-based activities imports the WSDL provided with the *ActiveBPEL Designer* product. The name of the WSDL is *identity.wsdl*. Refer to the *ActiveBPEL Designer* documentation for details about this WSDL. ActiveBPEL for People developers do not need to import this WSDL when using the People activity.

View and update identity service settings as shown in the following table.

Provider Configuration

Enable	Add a checkmark to use the identity service. Initially the service is disabled since it is not configured and ready for use. Configure the remaining settings, and then enable the service.
Provider Type	Select from the drop-down list: <ul style="list-style-type: none"> • LDAP. An LDAP-based service, such as Microsoft Active Directory • LDIF. (LDAP Data Interchange Format). A file-based service representing (LDAP) directory content. • XML File. A file-based service, such as the entries found in <i>tomcat-users.xml</i> file. <p>Be sure to see the Notes below table.</p>
File	This field is displayed only if the Provider Type is LDIF or XML File. <p>Do one of the following:</p> <ul style="list-style-type: none"> • For an LDIF provider, provide a path to your .LDIF file. (See Notes below table.) • For a provider based on XML user entries, provide a path to your file. (See Notes below table.)
Host	Enter the LDAP server's DNS name such as ldap1.my-domain-name.com or IP address such as 192.168.1.1.
Port	Enter the port to use for communications between the ActiveBPEL server and the LDAP server. The default value is 389.
Use SSL	Enable this checkbox to provide encrypted transport communication between ActiveBPEL and the LDAP service. If you enable this, you must enter a trusted keystore file location in the next field.
Trusted keystore file location on the server	Enter the full path to the aeTrustedCA.ks file for the Trusted Keystore Path. This file must be accessible by all instances of the server when deployed in a clustered environment. This path is required if SSL is enabled. For details, see Creating a Trusted Keystore File .

User DN Enter the user distinguished name. Most directory servers do not allow anonymous access, therefore the username and password is required. The username should be the distinguished name of the user.

For Microsoft Active Directory, an example of the user distinguished name is:
 CN=Administrator, CN=Users, DC=domainname, DC=com (or local)

For an open LDAP service, an example of the DN is:
 uid=admin, ou=system

Password Enter the administrator password for access to the directory service, and confirm it.

User Search Filter Configuration

User search base DN Enter the root distinguished name to indicate the base search criteria for authenticated users and groups.

For Microsoft Active Directory, an example is:
 CN=Users, DC=domainname, DC=com (or local)

For an open LDAP service, an example is:
 ou=employees,dc=example,dc=com

User search filter Enter the parameters needed to query the service for users. These parameters should exclude directory objects such as printers, servers, other non-user computers.

For Microsoft Active Directory, an example is:
 &(objectclass=person)!(objectclass=computer)

For an open LDAP service, an example is:
 (objectClass=inetOrgPerson)

Returned user attributes Enter the attributes that are returned for a user when a query is made. You must enter `cn` (common name) and email. You can use other attributes, such as pager or phone number.

For Microsoft Active Directory, an example is:
 cn,mail

For an open LDAP service, an example is:
 cn,mail

Users search scope To make a directory search efficient, select the appropriate level to search for entries.

One Level. Select if the user entries are all at the same level in the directory structure, for example in a folder called *Users*.

Subtree. Select this if user groups are nested in a directory structure.

Principal name attribute Enter the username attribute in a directory entry.
For Microsoft Active Directory, enter `sAMAccountName`
For an open LDAP service, the default value is `uid`.

Group Search Filter Configuration

Group search base DN Enter the directory tree where you want to start the search.
For Microsoft Active Directory, an example is:
`CN=Users, DC=domainname, DC=com` (or `local`)
For an open LDAP service, an example is:
`ou=groups, dc=example, dc=com`

Group search filter Enter the parameters needed to query the service for groups. These parameters should exclude directory objects such as printers, servers, other non-user computers.
For Microsoft Active Directory, an example is:
`(objectClass=group)`
For an open LDAP service, an example is:
`(objectClass=groupOfUniqueNames)`

Returned group attributes Enter the attributes that are returned for a group when a query is made. You must enter `cn` (common name) so that the group name is returned along with the users in the group. You can use other attributes, if desired.
For Microsoft Active Directory, the entry is:
`cn`
For an open LDAP service, the entry is:
`cn`

Group search scope To make a directory search efficient, select the appropriate level to search for entries.
One Level. Select if the group entries are all at the same level in the directory structure, for example in a folder called *Groups*.
Subtree. Select this if groups are nested in a directory structure.

Group name attribute Enter the group name attribute in a directory entry.
For Microsoft Active Directory, the entry is:
`cn`
For an Open LDAP service, the entry is:
`cn`

Group member attribute Enter the group member attribute in a directory entry.
For Microsoft Active Directory, the entry is:
`member`
For an open LDAP service, the entry is:
`uniquemember`

Notes:

- If you are running in a clustered environment, we recommend that you select LDAP for your Identity Service. If you select a file-based service, ensure that the file is shared or is on the same path in all servers.
- The file you select must be on the ActiveBPEL Enterprise server
- You can add an email entry to your XML file. For example, `<user username="user1" password="user1pw" roles="admin" email="user1@example.com"/>`
- The LDIF or XML file is read each time the server starts. You can update the file at any time. If the server is running, select **Update** to refresh the file contents.

Testing the Current Configuration

Enter a user name and select **Test Settings**.

If your settings are correct, you will see a success message.

If an error occurs, there may be helpful information for connection, security, or possibly an incorrect setting.

When your settings are correct, select **Update**.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

Creating a Trusted Keystore File

To ensure that the ActiveBPEL server can trust the SSL certificate presented by the LDAP server, the LDAP server's certificate, or its CA's certificate such as Veri-Sign, must be installed in a Java key store file which is designated as the store keeping a list of trusted certificates. This trust key store file must be accessible by the ActiveBPEL engine.

Example

Assuming your LDAP server is Apache DS running SSL using a self-signed certificate, you would use the following steps to create a trusted keystore file.

- 1 Export the Apache DS SSL certificate as a DER-formatted file using the Java/Sun keytool. For example:

```
c:> keytool -export -keystore apacheds.ks -alias
apacheDsAlias -file aeldap.cer
```

where *apacheds.ks* is the key store database in which the Apache DS SSL certificate is stored, *apacheDsAlias* is the alias within that key store, and *aeldap.cer* is the name of the file where the certificate is exported to.

- 2 When you run the command in Step 1, you will be prompted for the keystore password. Add the password for the *apacheds.ks* file.
- 3 Create a new keystore file, such as *aeTrustedCA.ks*, on the same machine as the ActiveBPEL engine. This key store will contain trusted certificates. Also in the same command, import the certificate. For example:

```
c:> keytool -import -file aeldap.cer -alias apacheDsAlias -keystore aeTrustedCA.ks -storepass secret
```

where *aeldap.cer* is the file that was exported from the Apache DS server *aeTrustedCA.ks* is the name of the new key store file on machine that is running the ActiveBPEL engine, and *secret* is the password for the *apacheds.ks* file.

- 4 When you run the command in Step 3, you will be asked whether to trust this certificate. Type in *yes* to add the certificate.
- 5 From the Identity Service Page of the Administration Console, enter the full path to the *aeTrustedCA.ks* file for the Trusted Keystore Path, as described in [Identity Service](#).

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

Task Manager (ActiveBPEL for People Inbox)

This page applies to the ActiveBPEL for People Inbox application.

Use the Task Manager page to select the number of day to retain completed, faulted, and exited tasks in users' Inboxes. Users can see these tasks when they select the **All Closed** filter from their Task List filters.

You can also set a cache size and duration for the Task Details display resources.

Set the properties as follows:

- **URL for Task Inbox.** Replace the *localhost:8080* setting with the server and port details where your Inbox application is deployed.
- **Default Retention Days for Closed Tasks.** Set the number of days to keep closed tasks in users' Inboxes. This setting does not affect the BPEL process storage settings.

- **Inbox XSL Cache Size.** To improve performance, the Inbox application can cache the XSL resources that are used to render the Task Details page. You may notice in the Resource Catalog that there are several XSL files. These are the default files for the Inbox. Other XSL files may be in use on other servers. Several of these files are stored in memory when a task is routed to an Inbox. If a task definition contains its own custom set of XSL files, you can increase the cache size. If no caching takes place, the resources are fetched for every transformation. Note that some tasks may not use any XSL resources: A BPEL process may have specified a different presentation technology, such as JSP.
- **Inbox XSL Cache Duration.** Enter the number of minutes before the cached is cleared. The Inbox application periodically empties its cached contents.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

5 Deployment

What's in this chapter

- [Deploy BPR](#)
 - [Deployment Logs](#)
 - [Deployed Processes](#)
 - [Deployed Process Detail](#)
 - [Deployed Process Version Detail](#)
 - [Deployed Services](#)
 - [Indexed Properties](#)
 - [Partner Definitions](#)
 - [Resource Catalog](#)
-

Deploy BPR

The Deploy BPR page allows you to add new business process archives (.bpr) to the server.

You can deploy one .bpr archive at a time, but the .bpr archive can include as many BPEL files, deployment descriptors, partner definition files, WSDL definitions, and other resources as you wish.

For information on .bpr archives, see *What is a Business Process Archive?* in the *ActiveBPEL Server User's Guide*.

To deploy a .bpr archive:

- 1 Browse to a folder containing a business process archive (.bpr file).
- 2 Select a .bpr file.
- 3 Select **Deploy**.

The engine validates the files contained in the .bpr and stores the files in the database. The Deployment Log page appears showing errors, warnings, and information about the deployed process files.

After you deploy a .bpr file, you can view details for deployment descriptors, partner definition files, BPEL files, indexed properties, WSDL definitions, and schema files by making selections in the Administration Console navigation bar.

Go back to [Deployment](#).

Deployment Logs

The Deployment Logs page shows a list of logs generated when new and modified business process archive (.bpr) files are deployed. The number of errors and warnings generated, if any, are shown.

On this page you can:

- Change the display of the logs list by using the **Selection Filter**
- Select a .bpr file to view its deployment log

To select deployment logs:

- 1 Select the **Log Contents** type, if desired.
- 2 Select **Deployed between** dates, if desired.
- 3 Type in the exact **Name** of a .bpr file, if desired.
- 4 Select **Submit**. The Deployment Logs list redisplay based on your selection filters.

Deployment Log

The Deployment Log page shows the name, date, and log for the selected .bpr file. During deployment, the engine validates the deployment descriptor of the BPEL process, ensuring that the associated WSDL file and other resources are available and valid for the current version of the process. If any validation errors or warnings occur, make corrections and redeploy the .bpr file or create a new .bpr file for any invalid processes.

Go back to [Deployment](#).

Deployed Processes

The Deployed Processes page lists all business processes that have been deployed to the server. For each deployed process, basic process version information is displayed, as shown in the table.

Item	Description
Name	Local part of the process qualified name (qname)
Group	Group this process belongs to, as added in the PDD file.
Active Ver.	Version that process instances can attach to or can run to completion. Normally the active version is the current version. However, if the current version has reached its expiration date, active processes can run to completion based on the expired version.
Versions	Number of deployed versions stored in the database
Future Ver.	Yes/no field indicating whether a process version has an effective date set to a future date

You can select the following **Selection Filters** to view a subset of processes:

- Process **Status**, as described below
- **Process Name**. You must type in the exact name and select **Submit**.
- Process **Group**. You must type in the exact group name, and select **Submit**. There is a default System group, which is hidden by default. Remove the checkmark from **Hide System** to display System processes.

Process versions can have one of the following statuses:

- **Current**. By default, when no version information is specified in a deployment descriptor, a deployed process is the current version with an immediate effective date. It is ready to receive requests.
- **Future**. If an effective date is specified in a process' deployment descriptor, a process has a future version.
- **Expired**. A version is expired if reaches the expiration date specified in a process' deployment descriptor, you manually expire the version, or a newer version is deployed.
- **Inactive**. When all process instances of an expired version complete, a process version is inactive.

See [Process Versions](#) for more details.

Select a process to display the Deployed Process Detail page.

Go back to [Deployment](#).

Deployed Process Detail

This page displays a list of all deployed versions of a process.

You can do the following:

- To expire the current version and inactivate future versions of a process, select **Expire All**.

The result of expiring all versions is that all running processes attached to the current version will complete, but no new processes can be started. Future versions will not become active when their effective date arrives. Note that if you want to terminate a running process, you must navigate to the Active Processes page and select a process.

You can reactivate an expired version. For details, see [Deployed Process Version Detail](#).

- To view details for a single version and to inactivate only that version, select the Version number.

The Deployed Process Detail page shows a list of all deployed versions of a process.

For details, see [Process Versions](#).

Property	Description
Version	The version number is automatically incremented when a new process version is deployed, unless a version number is specified in the deployment descriptor. The format of the number is N.nn, where N is major and n is minor.
Plan Id	Id assigned to this version and associated disposition of running processes
Effective Date	If an effective date was not specified in the deployment descriptor, the effective date is the same as the Deployed Date. A process is effective immediately when deployed, unless an effective date is specified.
Expiration Date	A process version does not have an expiration date unless one is specified in the deployment descriptor. By default, a process version automatically expires when a newer version becomes effective.
Deployed Date	Date the process is added to the engine database
Processes	Number of active process instances

Property	Description
Migrated To	Process version that this version migrates to
Status	Current, Future, Expired, Inactive

Go back to [Deployment](#).

Deployed Process Version Detail

The Deployed Process Version Detail page displays all the details from the process deployment descriptor as well as the process definition.

You can do the following on this page:

- View the **Process Group**, if any, that this process belongs to, as specified in the Deployment Descriptor.
- Select **View Process Graph** to see the process in Outline view and Graph view. These views also are available for running processes. For explanations of these views, see [Using the Process Details Page](#).
- Review **Version Detail** information. Refer to [Process Versions](#) for detailed explanations.
- Depending on the version status (Current, Future, Expired, Inactive), you may be able to update the effective date, expiration date, and running process disposition. For example, you can add an **Expiration Date** to the current version, and select **Update**.
- To inactivate this version immediately, select **Expire**. No new process instances can attach to this version.
- If you had previously expired this version, you may be able to restore it. To restore this version to the current or future version, select the available option, **Restore to Current** or **Restore to Future**. Note that the Restore to Current button is not available if any version is Current. You must expire all versions before you can use Restore to Current.
- Select the **BPEL** tab to view the BPEL XML source code

The Deployed Process Version Detail page also shows endpoint references, indexed properties, and resource usage:

- **My Role** partner link endpoint reference details are generated from information in the deployment descriptor. Rest your mouse on the partner link **Type** to view the associated namespace. Select the **Service Name** link to view the WSDL file for the Web Service exposed by the partner link. For details, see

[Deployed Services](#). Select **View** in the **Policy** column to view any attached policy assertions. See the *ActiveBPEL Enterprise Server User's Guide* for details on policy assertions.

- **Partner Role** partner link endpoint reference details are generated from information in the deployment descriptor. Rest your mouse on the partner link **Type** to view the associated namespace. Select a **static** endpoint type from the **Linkage** column to view the endpoint definition.
- **Indexed Properties**, if any are displayed. For details, see [Indexed Properties](#).
- **Resource Usage** shows WSDL, schema, and other files and their target namespace referenced in this process. Select a **Resource** to view the resource definition.

Go back to [Deployment](#).

Deployed Services

A deployed process contains at least one My Role partner link, and this partner link is assigned a service name in the Process Deployment Descriptor (PDD) file. The service name identifies the WSDL that the ActiveBPEL engine generates during deployment, and adds to the Services page. The WSDL includes the messages, operations, service, and binding details for the Web Service exposed by the process' My Role partner link.

The process receives messages at the Web Service address, which is shown in the following example:

```
http://localhost:8080/active-bpel/services/[servicename]?wsdl
```

Note: Some services are deployed as *external*, indicating they are not exposed as Web Services. An example is a Retry Policy service, which is a process deployed to tell the engine how many times to retry a non-communicating service. This type of process is not intended for outside consumption. For more information on External services, see the *ActiveBPEL Server User Guide*.

The following details and links are included on the Deployed Services page.

Name	Service name assigned to a My Role partner link in the PDD file. Select the name to link to the WSDL generated for this partner link. The WSDL is the Web Service that receives inbound messages. The Web Service address is in the form of:
------	--

```
http://localhost:8080/active-bpel/services/[servicename]?wsdl
```

Process Name	Process containing the My Role partner link associated with this service
Binding	Standard SOAP binding styles indicating how to format inbound messages for the service. Can be one of: <ul style="list-style-type: none"> • MSG (Document Literal) • RPC Literal • RPC Encoded • External • Policy
Partner Link	Name assigned to a My Role partner link that is exposing the service. Select the name to link to the Deployed Process Version Detail page.

Using the Selection Filter

To display a service, you can type its name into the **Service Name** field and select **Submit**. You can also use the asterisk (*) wildcard to search for names. For example *par* returns all service names containing the “par” characters. To view all services, leave the Service Name field blank and select **Submit**.

Go back to [Deployment](#).

Indexed Properties

An indexed property is a variable property that serves as a selection filter for active processes. This property holds a piece of data, such as a customer Id, application date, or amount. Using an indexed property in a selection query provides a fast way to filter processes based on important data items.

For example, you can retrieve a list of faulting processes that share the same indexed property, suspend the process(es), fix bad data values, and continue process execution. For details, see [Using Selection Filters for Active Processes](#).

Indexed properties are defined in the process deployment descriptor file. Deployment details are as follows:

Item	Description
Plan Id	The deployed process associated with the indexed property
Name	Indexed property name. This name appears in the Indexed Property list in the selection filters Expression Builder.
Type	Property type, such as string or double
Variable Path	Process variable name and declaration location in the process
Part	Process variable part for message type variables
Query	Process variable part detail (optional)

For details on how to define an indexed property, see the *ActiveBPEL Server User's Guide* or *ActiveBPEL Designer Online Help*.

Go back to [Deployment](#).

Partner Definitions

A partner definition file contains the service information for a partner link that has been deployed designated as a *principal* endpoint reference in the process deployment descriptor (.pdd) file. See the *ActiveBPEL Server User's Guide* for details about partner definitions.

Select a principal, if any exists, to view details.

Partner Definition Detail

The following details are displayed for the selected principal.

Partner link type	The partner link type used in the partner definition for the principal
Role	Role defined for the partner link type

Select a partner link type to view the namespace and endpoint reference details for the partner definition.

Go back to [Deployment](#).

Resource Catalog

The Resource Catalog is the centralized cross reference for all WSDL, schema, XSL, and other resource files referenced in the process deployment descriptor (.pdd) files deployed to the ActiveBPEL server engine.

Any resource in the catalog can be accessed by any deployed BPEL process, and only one copy is maintained. There are no restrictions based on the deployment context.

Item	Description
Total Reads	The number of reads to retrieve resource information during process execution (in cache or not)
Disk Reads (%)	The number of reads made to resource files not in the cache expressed as an absolute number and percentage of Total Reads
Cache Size	The number of resource files in stored cache. The default is 100. You can set cache size on the Configuration page. Modifying the cache size may improve engine performance. See the Engine Properties topic.

The **Deployed Resources** list shows the type, name and namespace for the resource. Rest your mouse on the Type or Resource name to view the physical location where the resource was loaded from.

Select a resource to view details.

Use **Selection Filters** as follows:

- Select a resource **Type** from the list
- Type in a **Resource** name. You can use wildcards, such as tutor*.*
- Type in a **Target Namespace**. You can use wildcards.

Select **Submit** to display the filtered list of resources.

Resource Detail Page

The Resource Detail page shows the same information that is on the Resource Catalog for each resource and also displays the XML source code.

Type	Type of resource, such as WSDL, XSD, or XSL
Location	The actual physical location where the resource is loaded from. This helps to uniquely define the location when the deployment descriptor was created and can be used to have multiple resource files of the same name deployed to the engine. The WSDL location is referenced in the.pdd file
Namespace	Target namespace in the resource
Referenced By	The process versions referencing this WSDL

Updating or Deleting a Resource

You can make minor changes to a resource, such as correcting a referenced URL, and then save your changes by selecting the **Update** button.

You can remove any resource from the database that is not directly referenced by any process. Select the **Delete** button, if available.

Go back to [Deployment](#).

6 Process Status

What's in this chapter

- [Active Processes](#)
 - [Using Selection Filters for Active Processes](#)
 - [Alarm Queue](#)
 - [Receive Queue](#)
-

Active Processes

The Active Processes page shows a list of process instances that have been or are executing in the ActiveBPEL engine. The version that this instance is attached to is also shown. States can be *running*, *suspended*, *completed*, *compensatable* (for a subprocess), or *faulted*. Select a Process ID or Name to view details of the process instance. For more information, see [Process ID and Process Details](#).

Use the comprehensive **Selection Filter** settings for advanced filtering and selecting of processes. For details, see [Using Selection Filters for Active Processes](#).

Note: If an active process is a subprocess, that is, it is invoked by another process, you may see additional state information for it. The additional state for a subprocess is *compensatable*. See the *ActiveBPEL Server User Guide* or *ActiveBPEL Designer Online Help* for more information on creating a BPEL process to be used as a subprocess.

Go back to [Process Status](#).

Using Selection Filters for Active Processes

You can filter the active processes list by using a wide range of properties and functions. Select the filters to apply and then select **Submit**.

The selection filters include:

- **State.** Select one of the following process states:
 - **All.** All states of process instances.
 - **Running.** Normally running processes.
 - **Completed.** Normal completions.
 - **Compensatable.** A sub-process is complete and eligible for compensation.
 - **Faulted.** Processes completed with a fault.
 - **Suspended.** A process suspended for any reason.
 - **Suspended (Faulting).** Suspended on a faulting activity. You can update variables on a faulting process prior to resuming it. For details, see [Process Exception Management](#).
 - **Suspended (Activity).** Suspended on a BPEL suspend activity.
 - **Suspended (Manual).** Manually suspended process.
- **Created between.** Date and time range for process starts.
- **Completed between.** Date and time range for process completions.
- **Name.** Process name.
- **Group.** The group this process belongs to, if any. The Group name is specified in the PDD and is displayed on the Deployed Process Version Detail page. There is a default System group, which is hidden by default. Remove the checkmark from **Hide System** to display System processes.
- **Additional query.** Use the Expression Builder to create a query based on an extensive set of criteria. Select the Dialog Button at the end of the row to open the Expression Builder.

Using the Expression Builder

You can create and submit a query for retrieving processes for display. In the Expression Builder, double-click the properties and functions to build the query, and select OK. The expression appears in the **Additional query** text box. You can edit the expression and can use it in conjunction with the other criteria in the Selection Filters. Select **Submit** to retrieve processes that meet the criteria selected.

The following table describes the functions, variables, and properties you can use for filtering the active processes list.

Criterion	Example Expression
Process Properties	
End Date	<pre>getProcessProperty("EndDate") >= "2007/02/17 10:03 AM"</pre> <p>Use the Date selector to enter a correctly formatted date.</p>
Id	<pre>getProcessProperty("Id") = '102'</pre>
Group	<pre>getProcessProperty("Group") = 'NorthWestRegion'</pre>
Name	<pre>getProcessProperty("Name") = 'LoanApproval'</pre>
Namespace	<pre>getProcessProperty("Namespace") = 'http://services.acme.com'</pre>
Start Date	<pre>getProcessProperty("EndDate") <= "2006/02/17 10:03 AM"</pre> <p>Use the Date selector to enter a correctly formatted date.</p>
State	<p>Property Codes for process states:</p> <ul style="list-style-type: none"> 1 - Running 2 - Suspended 3 - Completed 4 - Faulted 5 - Compensatable <p>Example: <pre>getProcessProperty("State") = '1'</pre></p>
State Reason	<p>Property Codes indicating the reason why a process is suspended:</p> <ul style="list-style-type: none"> 2 - Suspended (Activity). Suspended at a BPEL suspend activity 1 - Suspended (Faulting). Suspended as a result of an uncaught fault 0 - Suspended (Manual). Suspended manually. <p>Example: <pre>getProcessProperty("State'Reason') = '1'</pre></p>
Version	<pre>getProcessProperty("Version") = '2'</pre>
Indexed Properties. For details, see the Indexed Properties topic.	
Functions	
<code>getParentId()</code>	<pre>getParentId() = '101'</pre> (returns all subprocesses whose parent process Id is 101)
<code>getProcessProperty("...")</code>	See Process Property examples above
<code>getIndexedPropertyValue("...")</code>	<pre>getProcessIndexedProperty('amount') >= 50000</pre>
<code>hasWaitingAlarm()</code>	<pre>hasWaitingAlarm()</pre>
<code>hasWaitingReceive()</code>	<pre>hasWaitingReceive()</pre> (For processes with a waiting receive or OnMessage)

Criterion	Example Expression
hasWaitingReceive("partner-LinkName"[,op])	hasWaitingReceive('{http://services.acme.com}AR', 'Invoice')
isParentProcess()	isParentProcess() (returns all processes that invoke another process) isParentProcess() OR isSubProcess() (returns all processes involved either as parent or subprocess)
isSubProcess()	isSubProcess() (returns all processes that are subprocesses of parent processes)
nextAlarmTime()	nextAlarmTime() > '2007-12-31 14:30' (returns all processes which have an alarm scheduled for any time after 2:30 pm on December 31, 2007)
not(...)	(getProcessProperty("Name") = 'LoanApproval' not(getProcessProperty("EndDate") >= '2005-02-01 4 am'))

Go back to [Process Status](#).

Alarm Queue

View a list of active On Alarm process activities.

Select one or more options from the **Selection Filter** option list to narrow your view of active alarms.

Deadline Between	Beginning and Ending date and time for alarm
Process Id	Process instance Id. You can find this Id on the Active Processes page
Process Name	Local part of the process qualified name (qname)

Note: Alarms for system processes are not displayed, unless you ask for them by process id or name.

Go back to [Process Status](#).

Receive Queue

View a list of active receive, onMessage, and onEvent activities. These activities are queued for incoming messages.

You can do the following from this page:

- Select a receive and then select a partner link to view details. A window opens where you can see the BPEL process location in which the receive activity

executes. You can also see the correlation property alias and data, if any, associated with this receive activity.

- Select one or more options from the **Selection Filter** option list to view a selection of active receives. You can find this information on the Deployed Process Version Detail page, which shows the BPEL source code.

You do not need to enter the fully qualified name for the operation.

Note: Receives for system processes, especially for ActiveBPEL for People tasks, are not displayed, unless you ask for them by partner link name.

Go back to [Process Status](#).

7

Process ID and Process Details

You can enter a process instance ID in the **Process ID** text box, select **Go**, and then view the Process Details window, a comprehensive snapshot of a running, suspended, faulting, faulted, or completed process instance. The process IDs to select from are on the Active Processes page.

The Process Details page helps you analyze the execution state of the process instance.

You can do the following from Process Details page:

- View process and activity-level properties and values
- View the execution state of each activity
- Inspect the current value of variables and their attachments, activity links, partner links, correlation sets, fault, compensation, and event handlers
- Download variable attachments
- **Refresh** a running process to view an updated snapshot of the execution state
- **Suspend**, **Resume**, or **Terminate** a running process. These buttons appear at the top of the outline view.
- View and download the execution log to your computer from the process log. Select the **View Process Log** button at the top of the outline view. Copy the execution log to your computer from the **Process Log** text box.
- Perform process exception management by correcting, resuming, retrying or completing a faulting activity. For details see [Process Exception Management](#)

Note: If the Process Log box displays “Log file not available,” it means logging was not enabled when this process instance ran. For future process instances, you can enable logging from the Configuration page of the Administration Console.

See Also:

- [Using the Process Details Page](#)
- [Using the Process Details Graphic View](#)
- [Using the Process Details Outline View](#)
- [Inspecting Where and Why an Activity Faulted](#)
- [Working with Variable Attachments](#)

Using the Process Details Page

The Process Details page presents many details about a process instance:

- An **Outline** view shows the structural elements of a BPEL process and the current process execution state of each activity. You can select an element to view its properties and values.
- A **Graphic** view shows the main process flow. If the process has event handlers, fault handlers, and compensation handlers, you can view them by selecting a tab, such as Fault Handlers shown below. You can also select an activity to view its properties.
- A **Properties** view appears for a selected element. For example, when the Process element is selected from the Outline view, you can see process properties and their current values.

The following illustration shows an example of the Process Details page.

The screenshot displays the activeBPEL interface. The title bar reads "Active Process Detail: loanApprovalProcessComplete (ID 3) Refresh | Help | Close". The left sidebar shows a tree view of the process structure, including partnerLinks, partners, variables, faultHandlers, flow, receive, invoke, assign, and reply. The main area shows a flow diagram with activities: receive, invoke, assign, and reply. The "Process" tab is selected, and the "Process" properties table is visible at the bottom.

Property	Value
Name	tutorial
Current State	Completed
BPEL Namespace	http://docs.oasis-open.org/wsbpel/2.0/process/executable
Target Namespace	http://tutorial
Suppress Join Failure	yes
Start Date	2006-11-20 10:34:30
End Date	2006-11-20 10:45:15

Copyright © 2004-2007 Active Endpoints, Inc.

For details, see:

- [Using the Process Details Graphic View](#)
- [Using the Process Details Outline View](#)
- [Working with Variable Attachments](#)

Go back to [Process ID and Process Details](#).

Using the Process Details Graphic View

The Graphic view of Process Details shows the main process flow and the execution path through the process. You can also view the process fault, event, and compensation handlers, if the process definition includes these process-level handlers. The handlers have their own tabs in the view.

In the upper-right panel of the page, you see the main flow of a BPEL process. The process diagram reflects the layout rendering that is part of Active Endpoints BPEL design tool, *ActiveBPEL Designer*.

Each process activity has an icon, a label, and an execution state indicator, as shown in the following illustration.

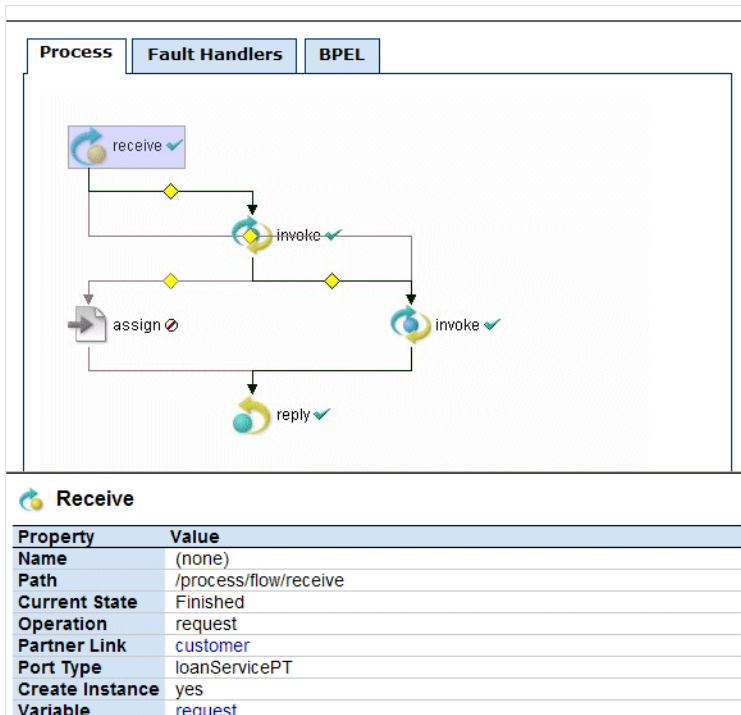


- 1 Activity icon. Activity icons are the same as those supplied with ActiveBPEL Designer.
- 2 Activity label, which can be the activity type, name, type:name, or custom text
- 3 Execution state indicator. For a description of each indicator, see [Using the Process Details Outline View](#).

Activities may appear in different colors, to indicate different execution states, as the following table describes.

Activity Color	Execution State
full color	Executed
muted color	Ready to execute or inactive
gray	Dead path

The Graphic view looks similar to the following example.



To view details, do the following:

- Select an activity from the diagram to view its properties
- Select an activity from the diagram to put the activity in focus in the Outline

To print the diagram, select **Print Picture** from the right-mouse menu. The diagram prints with the same caption that appears in the graph view. The timestamp indicates when the Process Details page was opened or refreshed.

Printing Tips:

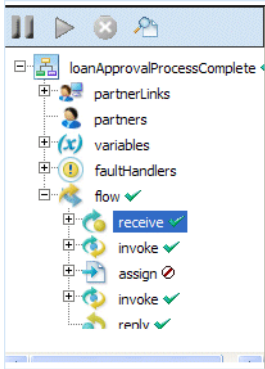
- To print a large diagram, select appropriate scaling options in your Printer options dialog, such as Fit to Page or print as x% of Normal Size
- Save the diagram as an image file to print later

See also [Using the Process Details Outline View](#) and [Working with Variable Attachments](#).

Back to [Process ID and Process Details](#)

Using the Process Details Outline View

The following illustration shows a sample Outline view of a process instance.



Outline View Menu Bar

The **Suspend**, **Resume** and **Terminate** buttons are enabled only if the process is currently running or suspended. If logging is enabled, you can select the **View Process Log** button to display the execution details for the process instance. See [Engine Properties](#) to enable logging.

Structural Elements of a BPEL Process:

- **Process name** is the local part of the process qualified name
- **Partner links** represent the Web services that are invoked
- **Variables** contain the message or other data received, manipulated, and sent from the process. A variable may include attachments.
- **Correlation sets**, if present, contain the message properties that track different conversations carried on by the process
- **Fault handlers**, if present, catch faults thrown by process activities
- **Event handlers**, if present, run concurrently with a process scope and invoke an activity based on an alarm or message received
- **Termination handlers**, if present on a scope, handle process termination
- **Activities** carry out the processing steps


To view details about a process element, select it. The following table describes each element.


Process Element in Outline Details Displayed


Process name	<p>Current state, Start/end time of process instance, and deployment details for the process.</p> <p>Fault details may also be displayed.</p>
Partner links	<p>The type(s) of partner links: partner role and/or my role.</p> <p>The endpoint reference of the partner link service. You can see the address information in the Properties view. Scopes can have their own local partner links.</p>
Variables	<p>The variable type: message, schema type, or schema element.</p> <p>The current value of the variable. For a running process, the value is current as of the time you opened or refreshed the Process Details window.</p> <p>If the variable has attachments, they are also displayed and can be downloaded.</p>
Correlation sets	<p>The message property definition and current value. A correlation set contains a message property to ensure that each process conversation is uniquely identified.</p>
Fault handlers	<p>Name, state, and details of fault handling activity. Scopes can have their own local fault handlers.</p>
Event handlers	<p>Name, state, and details of event handling activity. Scopes can have their own local fault handlers.</p>
Activities	<p>The activities section of the Outline begins with a flow (or other) activity that represents the main container for the whole process. Within the flow, there is a list of all process activities. The activities are in the same order as in the BPEL XML code. If the process was designed in ActiveBPEL Designer, the order matches the Outline view order.</p> <p>Note: The activity list shown is not necessarily in execution order.</p> <p>For each activity, you can view the execution state and activity definition.</p>
Links	<p>If an activity is the source of a link, the link is displayed below the activity node. Link properties are displayed, including link status (whether or not the link executed), the transition condition, if it exists, and the link's target activity.</p>


Activity States


You can determine the execution status of each activity by looking at the icon next to the activity.


 Executing


 Ready to Execute

 Finished

 Faulted. Occurs when a fault is thrown during the execution of an activity.

 Faulting. Occurs when a fault is thrown during the execution of an activity and the fault is uncaught. If desired, you can make corrections or resume faulting processes. See [Process Exception Management](#).

 Terminated. Occurs when the process is manually terminated.

 Dead Path

 Suspended

(none) Inactive (the initial state of an activity)

For a running process, the icon next to an activity may change if you refresh the Process Details window.

Process States

The process can have the following execution states:

Completed Normal completion

Failed	Completed with a fault or termination
Running	Snapshot of the executing process when you open the Process Details window. The process continues to run, but the Process Details window is not updated unless you select Refresh .
Suspended	The process stops running when you select Suspend from the Process Details window.
Faulting	Execution is stopped on a faulting activity. The activity has an uncaught fault and the process is configured for suspension on an uncaught fault.

See also [Process Exception Management](#), [Inspecting Where and Why an Activity Faulted](#) and [Using the Process Details Graphic View](#).

Back to [Process ID and Process Details](#)

Inspecting Where and Why an Activity Faulted

In the Process Details window, the Outline view shows a list of process activities. A red X appears next to an activity that faulted, or a red triangle next to a faulting activity.

You can select the process name to view details about the fault, as the following illustration shows.

Property	Value
Name	unit_simple_throw_with_message
Path	/process
Current State	Suspended (Faulting)
Target Namespace	http://www.active-endpoints.com/process/unit_simple
Suppress Join Failure	yes
Start Date	2006-02-07 14:42:21
Fault Name	selectionFailure
Fault Namespace	http://schemas.xmlsoap.org/ws/2003/05/soap/envelope/

Fault Message Data

```
<messageData name="simpleResponse" namespaceURI="http://www.active-...>
<part name="dataout">FAILURE</part>
</messageData>
```

Fault information includes:

Fault Name	Standard BPEL or engine fault name
Fault Namespace	Standard BPEL or engine fault namespace

Fault Source	Process activity that threw the fault
Fault Message Data	Data in the throw or catch fault variable
Fault Attachments	Content external to the variable may be included and can be downloaded

You can get further information about faults:

- Select the faulted activity to view the Fault Name. For details about BPEL standard faults see [BPEL Standard Faults](#), and for engine faults see [ActiveBPEL Custom Faults](#).
- Select **View Process Log** in the Outline to view the process log. You can see the execution path leading to the faulted activity.
- For a faulting activity, you can correct data, retry or complete the activity. See [Process Exception Management](#).

Note: If the Process Log is not visible, you must enable logging on the Configuration page.

Back to [Process ID and Process Details](#)

Working with Variable Attachments

You can view and download attachments for a variable when a process is running, faulted, or completed.

If a process is running, you can also add and remove variable attachments.

You can do the following to view, download, add, and remove process variable attachments:

- 1 Select a process from the Active Processes list.
- 2 In the Active Process Detail window's Outline View, expand the list of variables.
- 3 Select a variable from the list to view the variable instance data and attachments, as shown in the following illustration for a running process.

(x) Variable

Property	Value
Name	titleMessage
Path	/process/variables/variable[@name='titleMessage']
Current State	Inactive
Message Type	ns1:titleMessage
Type Namespace	http://active-endpoints/QA/testing/dvds

Variable Instance Data

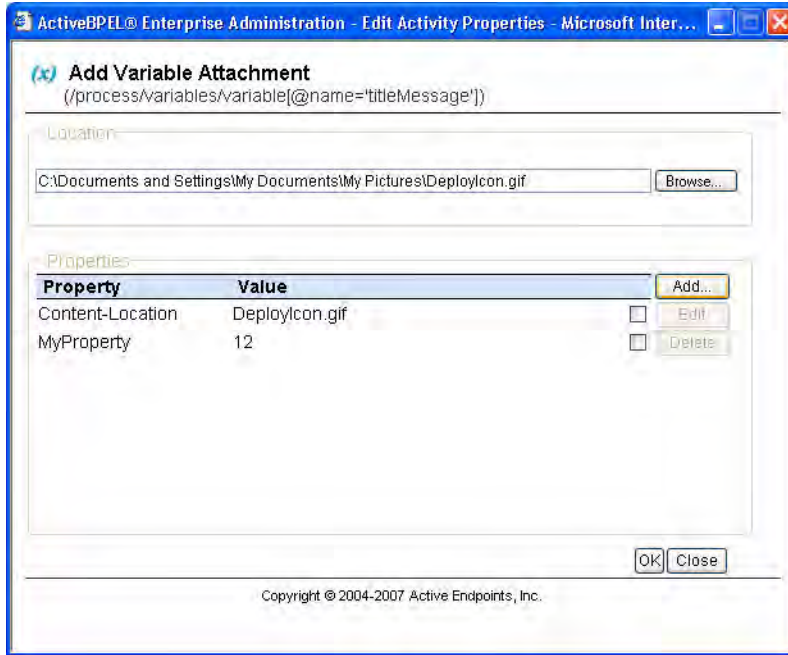
```
<part name="title">t</part>
```

Attachments

Attachment # 1	Download <input type="button" value="Delete"/>	
	Content-Id	bpel-admi
	Content-Type	image/pjp
	Content-Location	70103596
	X-Size	78108
	MyProperty	12
Attachment # 2	Download <input type="button" value="Delete"/>	
	Content-Id	bpel-admi

Adding an Attachment

While a process is running, you can add one or more attachments to a variable. In the Active Process Detail page, expand a variable to display the Attachments header. Select **Add** to open the Add Variable Attachment window, as shown in the example.



You can browse to locate a file to attach. You can also add, edit, or delete attachment properties and values.

Deleting an Attachment

To delete an attachment for a running process, select **Delete**, next to the Download link.

Viewing or Downloading an Attachment

For a running, faulted, or completed process, select the Download link to open a dialog that asks where to open or save the file.

Back to [Process ID and Process Details](#).

8

Process Versions

ActiveBPEL Enterprise manages process versions. Version details include:

- [Process Version Life Cycles](#)
- [Process Version Persistence Type](#)
- [Exception Management Type](#)
- [Process Instance Retention Days](#)

Go back to [Process ID and Process Details](#).

Process Version Life Cycles

Process versioning allows different versions for a given process to exist in ActiveBPEL Enterprise. Two deployments are considered to be different versions of the same process if they have the same target namespace and name in the BPEL file, but one deployment differs from the other in some way.

Process versioning allows you to control when processes become effective and for how long. You can also control what happens to processes created by older versions when a new version becomes effective. While multiple versions of a process can exist concurrently, only the latest effective version is capable of creating new process instances.

The latest effective version is in a *current* state. Other states include *future*, to describe versions that have an effective date in the future, *expired* to describe versions whose expiration date has arrived or has been set, and *inactive* to describe expired versions that no longer have running process instances.

The process deployment descriptor provides selections for describing how a deployment is to be versioned. These selections are all optional and have default values as described below.

The following example shows the syntax for version information in the .pdd file.

```
<version effectiveDate="2005-12-12T00:00:00-05:00"  
expirationDate="2007-12-12T00:00:00-05:00" id="1.5"  
runningProcessDisposition="migrate" />
```

where:

- **effective date** is the date the new version becomes the current version and all new process instances run against it. Depending on the disposition selected for running processes, some may continue to run until complete using the older version. The effective date is an XML schema date/time value. The time expression includes a time zone, indicated as the midnight hour plus or minus the number of hours ahead of or behind Coordinated Universal Time (UTC) for the computer's time zone. In the example above, the computer time zone is eastern standard time, which is five hours behind UTC. If you do not provide an effective date, it defaults to the date and time the process is deployed to the server.
- **expiration date** is the date, beyond the effective date, the current version expires. An expired version is not capable of creating new process instances. Once all of the running processes tied to an expired version complete then the version becomes inactive. All process instances for the current version run to completion. The expiration date is an XML schema datetime value. (Same as effective date). If you do not provide an expiration date, the version does not expire until you manually expire it in the Administration Console or until a newer version is deployed.
- **id** is the process version number in major.minor format. You do not need to provide a version number. ActiveBPEL server auto-increments new versions. The server increments a version number by dropping the minor value and adding 1 to the max number. For example, version 1.5 increments to version 2.0.
- **runningProcessDisposition** is the action the server takes on any other versions of the same process that currently have processes executing once this version's effective date arrives.

Valid values for `runningProcessDisposition` are:

- **Maintain**. Indicates that all process instances for the previous versions should run to completion. This is the default value.
- **Terminate**. Indicates that all process instances running under previous versions should terminate on the effective date of the new version, regardless of whether or not the process instances are complete.
- **Migrate**. All running process instances created by previous versions will have their state information migrated to use the newly deployed process definition once its effective date arrives. If there are incompatible changes between the versions that would not permit them to be migrated, you receive an error mes-

sage during deployment of the new process, and its deployment fails. Changes should be limited to expression language changes in the BPEL XML file.

Criteria for a New Version

A new version of a deployed process must meet the following criteria:

- The fully qualified process name must not change
- A new version can include a change to either the BPEL XML or the process deployment descriptor file
- A version is not new if the autoincrement feature determines that the BPEL XML and the deployment descriptor are the same as the deployed version having the highest version number. If the BPEL XML and the descriptor file do not contain any differences, then the version on the server is up-to-date and no deployment occurs.
- If you need to modify WSDL-related properties, such as partner links or correlation properties, then you should create a new process, not a new version

Go back to [Process Versions](#).

Process Version Persistence Type

Persistence refers to storage of active processes. When a process runs on the server, by default all state information is stored in the server database. However, this setting can be changed in the PDD file.

Persistence setting selections are as follows:

Full	The default setting. For each process instance, all state information is stored for a running, faulted, and completed process.
None	No process information is stored in the server database when a process terminates

Go back to [Process Versions](#).

Exception Management Type

According to the WS-BPEL 2.0 specification, a process with an uncaught fault terminates.

On the Engine Properties tab of the Configuration page, you can enable an option to suspend all processes on an uncaught fault to put them in a suspended-faulting state. You can then perform process exception management on the faulting process followed by retrying or completing the faulting activity or scope.

An individual process can override the engine setting with an entry in the Process Deployment Descriptor (PDD) file. The settings are:

System Default	The current engine setting for all processes. The default engine setting is to disable suspension on uncaught fault; however, the current setting may be different.
False	Do not allow this process to suspend on an uncaught fault. The process will terminate abnormally. This setting overrides the engine setting.
True	Suspend this process on an uncaught fault to put it in a suspended-faulting state. You can then perform process exception management on the faulting process, followed by retrying or completing the faulting activity or scope. This setting overrides the engine setting.

See [Process Exception Management](#).

Go back to [Process Versions](#).

Process Instance Retention Days

You can specify how long to keep completed and faulted processes in the ActiveBPEL database before deleting them on an automated schedule. This setting is available in the following locations:

- Process Deployment Descriptor wizard in ActiveBPEL Designer. Add the setting to a process's deployment descriptor.
- [Deployed Process Version Detail](#) page of the Administration Console. Add or change the setting for a deployed process.
- There is a default retention days setting for all processes on the [Storage](#) page of the Administration Console. The setting for an individual process overrides this setting.

The retention days setting applies to a process version with a status of *current* or *future*, not to *expired* or *inactive* versions. The schedule begins on the Completed or Faulted date of each process, as shown on the Active Processes page. For example, if a process instance completes on December 31, and the Retention Days setting is 30 days, the process instance is deleted from the database after January 30, on the next scheduled deletion.

To set retention days for an individual process:

- 1 In the Administration Console, select **Deployed Processes**.
- 2 Select a deployed process from the list.
- 3 If multiple versions exist, from the Deployed Process Detail page, select a current or future version to open the Deployed Process Version Detail page.
- 4 In the **Process Instance Retention Days** field, specify a number of retention days for the process version.
- 5 Select **Update**. The effect is immediate for all completed and faulted processes.

For details about scheduled database maintenance, see [Storage](#).

Go back to [Process Versions](#).

9

ActiveBPEL Enterprise Clusters (Application Server Specific)

For some application servers, including JBoss, WebSphere, WebLogic, and Dot-Net, ActiveBPEL Enterprise supports clusters of process servers operating with a single persistent process store. This feature is not supported on Apache Tomcat.

Multiple process servers share a set of process instances and their deployments. Two main features of a clustered environment are load balancing and fail over. Load balancing allows processes to be routed to the server in the process cluster with the least load. Fail over allows servers to pick up the load of a failed server in the process cluster.

Cluster

Cluster Id	Id assigned to the cluster node by ActiveBPEL engine
Engine Name	Computer IP address and port where the engine instance is running
Start Time	Date and time the engine started
No. CPUs	Number that is based on licenses. Select the number of CPUs to view the Engine Detail page, where you can update the number. Then view the Administration Home page to verify that the number of licenses matches the number of CPUs. Note: ActiveBPEL reports an estimated number of CPUs based on the information it gathers from the computer. The number may not be accurate if the computer is using CPU hyper-threading technology.
Status	Engine statuses include Running and Stopped

Select an engine to view details.

Engine Detail

On the Engine Detail page you can:

- Update the number of CPUs that are running ActiveBPEL Enterprise
- Stop the engine for a specific CPU

To check license/CPU compliance:

- 1 In the **No. CPUs** field, type in the number of CPUs on the server where ActiveBPEL Enterprise is installed, and select **Update**.

- 2 Select **Home** in the Administration Console navigation bar. If the number of CPUs on the server exceeds the number of licenses installed, a warning is displayed.

Go back to [ActiveBPEL® Enterprise Administration Console Overview](#)

10 Process Exception Management

A process can receive an unplanned for fault during its execution. In such an event, the process may not have the necessary fault handling logic, and the result is process termination. A more desirable result is to suspend the process, retry, step over, or make corrections and then complete the process normally.

You can configure the ActiveBPEL server to suspend processes on uncaught faults. You can then use the following process exception management techniques available in the Administration Console:

- Find faulting processes quickly by using selection filters in the Active Processes page
- Update the value of variables, partner links, and correlation properties
- Retry an activity or scope
- Mark an activity or scope as Complete and continue to the next activity
- Set up alerts for faulting processes

Suspending a process on an uncaught fault

You can configure the engine to suspend processes on uncaught faults so that you can inspect a problem, make corrections, if desired, and continue the process. For details, see [Engine Properties](#). In addition, on a process-by-process basis, you can override the engine-wide setting with a setting in the PDD file. The global setting may not be desirable for all process types. For example, a “straight-through” process, which has someone waiting for a response, may not be a desirable candidate to suspend.

Finding and Correcting Faults on a Faulting Activity

Use the Selection Filters on the Active Processes page to view a list of faulting processes.

Select a process from the list and then from the Active Process Details page, you can do any of the following to correct a problem:

- Update the value of variables
- Update the endpoint reference of partner links, such as the service name or <wsa:Address>
- Update the value of correlation properties

- Retry, complete, or step activities

Selecting a Faulting Process

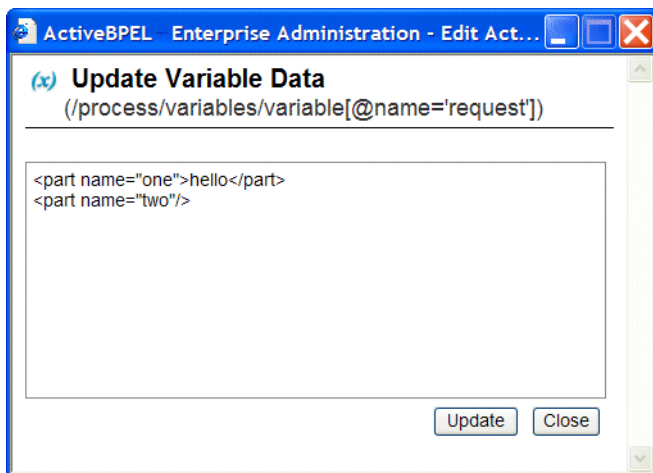
- 1 From the Active Processes page, select a process with a Suspended-Faulting state. Use the Selection Filters to quickly locate the process you are looking for.
- 2 Select the **Plan Id** to display the Active Process Details page.

Updating the Value of a Variable on a Faulting Activity

A variable may cause an uncaught fault if it is invalid with regard to an activity's operation.

- 1 From the Outline view of the Active Process Details page, select a variable. The current value of the variable is shown in the Variable Data Instance box.
- 2 Select **Edit** at the bottom of the Variable Data Instance box.
- 3 In the Update Variable Data dialog, make the modifications necessary. Possible modifications include:
 - data value(s)
 - XML data structure

The following example shows where you can make edits.



- 4 Select **Update**.

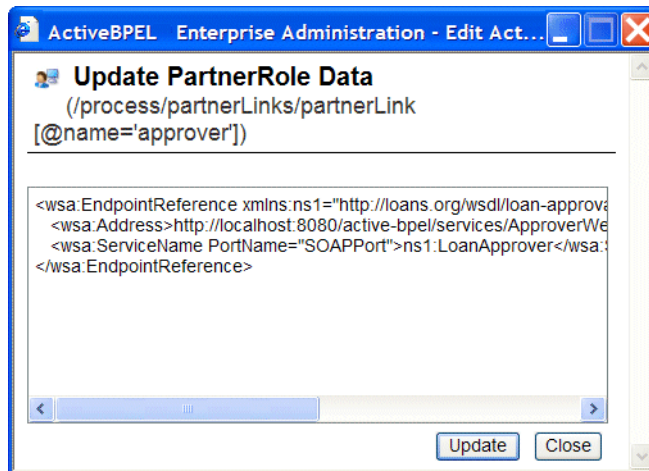
Updating Partner Link Data on a Faulting Activity

A partner link endpoint reference may cause an uncaught fault for a variety of reasons:

- The service is unavailable
- The address contains invalid information
- The address is missing required information, such as a header or credentials

If an endpoint is not available or the address is incorrect or incomplete, you can supply new <wsa:Address> information for a faulting partner link, as follows.

- 1 From the Outline view of the Active Process Details page, select a partner link. The current value of the endpoint reference is shown in the Partner Role Data box.
- 2 Select **Edit** at the bottom of the box.
- 3 In the Update Partner Role Data dialog, make the modifications necessary. The following illustration shows the text box where you can make edits.



- 4 Select **Update**.

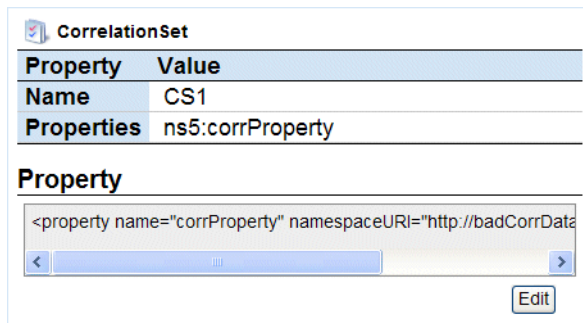
Updating Correlation Data on a Faulting Activity

A *correlation set* is a set of properties shared by messages. The purpose of the correlation set is to act as a conversation identifier: it keeps together all messages intended for the same process instance. If an activity is faulting due to bad message

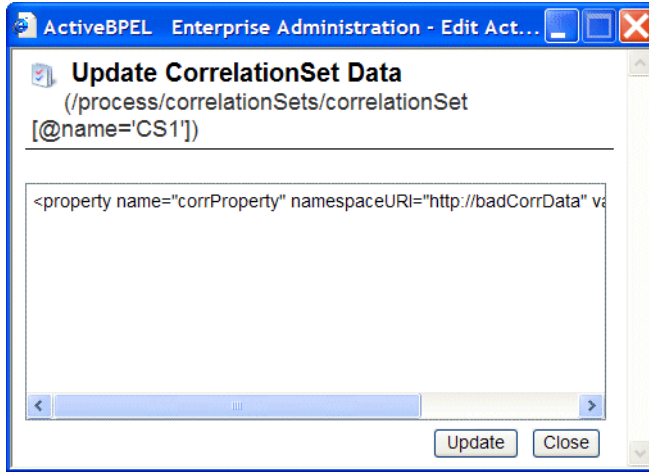
data, you may need to correct the correlation property data as well as the message variable data.

Correcting correlation data requires a comprehensive understanding of the message properties defined in the process's WSDL file and the expected value in the property alias associated with the input/output variables of receive, onMessage, invoke, and reply activities. Also, be aware that a correlation set can be declared at the process level or at a scope level. For details on defining and using correlation sets, see the *ActiveBPEL Designer Online Help*.

- 1 From the Outline view of the Active Process Details page, select a correlation set. The current value of the correlation property is shown in the Correlation Set Property box, as shown.



- 2 Select **Edit** at the bottom of the box.
- 3 In the Update Correlation Set Data dialog, make the modifications necessary. The following illustration shows the text box where you can make edits.

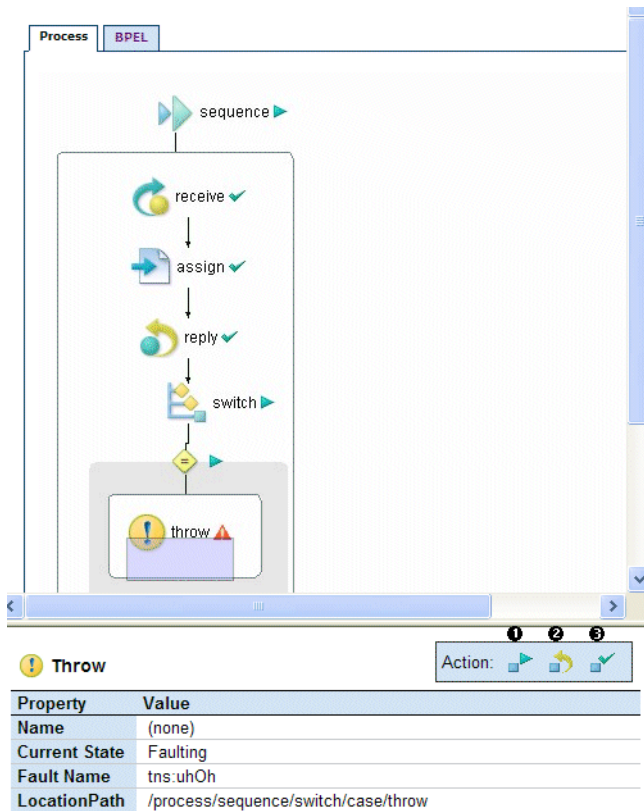


4 Select **Update**.

Using the Step, Retry and Complete Actions

After correcting the variables, correlation properties, or partner links causing an uncaught fault (if desired), you can retry or complete (step over) the faulting activity or an enclosing scope.

On the Active Process Details page, select the faulting activity or scope from the Outline view or Graph view. The activity shows a Faulting state and includes an **Action** bar, as the following illustration shows.



Actions include the following.

- 1 **Step** Steps the current activity. After stepping an activity, the next activity (if any) becomes ready for execution, and you can continue to step through the process or use the process level Resume button to continue. This button is disabled for a faulting activity since you can accomplish the same result simply by resuming the process from the Outline view menu bar.
- 2 **Retry** Retries the faulting activity or scope. If you corrected the cause of an uncaught fault, the next activity will become ready to execute and you can either resume the process or continue to step through the process one activity at a time. If you retry an executing scope, then the scope will first terminate any of its enclosed activities that are still running and also issue a compensate call to any eligible enclosed scopes prior to retrying.

- 3 Complete** Marks the activity or scope as completed normally. The next activity will become ready to execute, and you can either resume the process or continue to step through the process one activity at a time. Keep in mind that you may have to take some additional steps to fix a process if you use the Complete button. Since the activity that you are marking as complete does not execute, there may be variables or correlation sets that are not properly initialized as a result of completing the activity.

Setting Up Alerts for Faulting Processes

When unexpected faults occur, you can manually check for the occurrence of suspended processes through the Active Process Page. Alternately, you may want to have the system notify you if there was a problem with one of your processes. To do so, you can designate a service to be notified of these faults. The service can then take steps to dispatch notifications of the problem (e.g., to an enterprise management system or pager) or in systemic cases automate recovery of the process.

ActiveBPEL server allows you to designate a BPEL process as an alerting service. The service, which you can create in ActiveBPEL Designer, is based on a WSDL file that defines the schema, messages, port types, operations, and partner link types available for building an alerting system. Once the process is deployed, you can go to the Alerts tab of the Configuration page to set up the service to handle alerts.

For more information, see:

- [Alerts](#)
- Process Exception Management topic in *ActiveBPEL Designer Online Help*
- ActiveBPEL Designer installation/Support folder. This folder contains the alerts.wsdl file required for creating an alerting service.

Process Exception Management and Endpoint Policies

ActiveBPEL server uses endpoint policies governed by the WS-Policy specification. A policy can describe when to avoid invoking a service, based on system downtime and how many times to retry a service that does not reply. The endpoint policy information is added to the PDD file of a process.

For more information, see *BPEL Processes as Engine Services* in the *ActiveBPEL Server User's Guide*.

Back to [Process ID and Process Details](#)

A BPEL Standard Faults

The following list specifies the standard faults defined within the WS-BPEL specification. All these faults are named within the WS-BPEL namespace standard prefix *bpel:* corresponding to URI:

urn:oasis:names:tc:wsbpel:2.0:process:executable.

Fault name	Description
ambiguousReceive	Thrown when a business process instance simultaneously enables two or more IMAs for the same partnerLink, portType, operation but different correlation Sets, and the correlations of these activities match an incoming request message
completionConditionFailure	Thrown if upon completion of a directly enclosed <scope> activity within <forEach> activity it can be determined that the completion condition can never be true
conflictingReceive	Thrown when more than one inbound message activity is enabled simultaneously for the same partner link, port type, operation and correlation set(s).
conflictingRequest	Thrown when more than one inbound message activity is open for the same partner link, operation and message exchange
correlationViolation	Thrown when the contents of the messages that are processed in an <invoke>, <receive>, <reply>, <onMessage>, or <onEvent> do not match specified correlation information
invalidBranchCondition	Thrown if the integer value used in the <branches> completion condition of <forEach> is larger than the number of directly enclosed <scope> activities
invalidExpressionValue	Thrown when an expression used within a WS-BPEL construct (except <assign>) returns an invalid value with respect to the expected XML Schema type
invalidVariables	Thrown when an XML Schema validation (implicit or explicit) of a variable value fails

Fault name	Description
joinFailure	Thrown when the join condition of an activity evaluates to false and the value of the suppressJoinFailure attribute is yes
mismatchedAssignmentFailure	Thrown when incompatible types or incompatible XML infoset structure are encountered in an <assign> activity
missingReply	Thrown when an inbound message activity has been executed, and the process instance or scope instance reaches the end of its execution without a corresponding <reply> activity having been executed
missingRequest	Thrown when a <reply> activity cannot be associated with an open inbound message activity by matching the partner link, operation and message exchange tuple
scopeInitializationFailure	Thrown if there is any problem creating any of the objects defined as part of scope initialization. This fault is always caught by the parent scope of the faulted scope
selectionFailure	Thrown when a selection operation performed either in a function such as bpel:getVariableProperty, or in an assignment, encounters an error
subLanguageExecutionFault	Thrown when the execution of an expression results in an unhandled fault in an expression or query language
uninitializedPartnerRole	Thrown when an <invoke> or <assign> activity references a partner link whose partnerRole endpoint reference is not initialized
uninitializedVariable	Thrown when there is an attempt to access the value of an uninitialized variable, or in the case of a message type variable, one of its uninitialized parts
unsupportedReference	Thrown when a WS-BPEL implementation fails to interpret the combination of the reference-scheme attribute and the content element OR just the content element alone
xsltInvalidSource	Thrown when the transformation source provided in a bpel:doXsltTransform function call was not legal
xsltStylesheetNotFound	Thrown when the named style sheet in a bpel:doXsltTransform function call was not found

Back to [ActiveBPEL® Enterprise Administration Console Overview](#).

B ActiveBPEL Custom Faults

The following list specifies the custom faults defined for the ActiveBPEL engine. All these faults are in the namespace *http://www.active-endpoints.com/2004/06/bpel/extensions/*.

Table 1:

Fault name	Description
systemError	Unrecoverable system error
badProcess	Invalid BPEL
validationError	Error in message variable data. Validation errors are reported only if the configuration option “Validate input/output messages against schema” is enabled.
xpathFunctionError	Error in executing XPath function
invalidTransitionCondition	Non-Boolean return from an XPath evaluation of a transition condition
xpathDateParseError	error in parsing an xsd:date or xsd:datetime
xpathDurationFormatError	error in parsing an xsd:duration

Back to [ActiveBPEL® Enterprise Administration Console Overview](#)

