Red Hat JBoss Fuse 6.3

BPEL Development Guide

BPEL

Last Updated: 2018-07-16

Red Hat JBoss Fuse 6.3 BPEL Development Guide

BPEL

JBoss A-MQ Docs Team
Content Services
fuse-docs-support@redhat.com

Legal Notice

Abstract

Use this guide to help you develop integrated applications with SwitchYard.

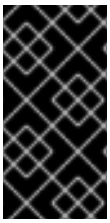
Table of Contents

CHAPTER 1. INTRODUCTION

1.1. UNDERSTANDING BPEL

Business Process Execution Language (BPEL) is an XML-based language that enables task-sharing in a distributed computing environment. It uses easy-to-understand commands that perform complex functions. All the processes in BPEL, export and import information by using web service interfaces only.

The BPEL Engine is optimized for the JBoss Application Server container. It is based on Apache Orchestration Director Engine (ODE) and manages process definitions and instances.



IMPORTANT

BPEL (based on the Riftsaw project) is no longer being actively developed and will be removed from a future release of JBoss Fuse. If you are currently using BPEL, it is recommended that you consider migrating to the Red Hat JBoss BPM Suite (which is supported through the JBoss Fuse Integration Package).

1.2. ABOUT BUSINESS PROCESS ORCHESTRATION

Business process orchestration refers to the act of specifying actions within business processes via web services.

BPEL is mainly used to model web service interactions on a distributed system. It allows for complex orchestrations of multiple service applications through a single controller service.

1.3. ABOUT APACHE ODE

Apache ODE ("Orchestration Director Engine") is a software component that is designed to execute BPEL business processes. It sends and receives messages to and from web services, manipulates data and performs error handling in the method prescribed in your process definition. To learn more about Apache ODE, visit the project website at http://ode.apache.org

1.4. FEATURES OF BPEL ENGINE

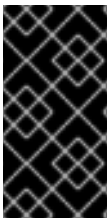
The BPEL Engine is based on Apache ODE and supports the following features:

- Provides BPEL component within the SwitchYard.
- Enterprise quality GWT based BPM console to manage process definitions and instances.
- Compiled approach to BPEL that provides detailed analysis and validation when deployed.
- Short-lived and long-running process executions.
- Process persistence and recovery.
- Process versioning.
- Runs in JBoss Cluster.

CHAPTER 2. BPEL AND SWITCHYARD

2.1. INTRODUCTION

The BPEL Component is a pluggable container in SwitchYard that allows you to expose a WS-BPEL business process as a service through an interface defined using WSDL.



IMPORTANT

BPEL (based on the Riftsaw project) is no longer being actively developed and will be removed from a future release of JBoss Fuse. If you are currently using BPEL, it is recommended that you consider migrating to the Red Hat JBoss BPM Suite (which is supported through the JBoss Fuse Integration Package).

2.2. STRUCTURE OF A SWITCHYARD BPEL APPLICATION

For SwitchYard BPEL applications, the artifacts within the **`src/main/resources`** folder are structured differently. The **`switchyard.xml`** configuration file is located in the META-INF folder. However, the BPEL deployment descriptor (**`deploy.xml`**), and the BPEL process definition are located in the root folder. You can locate the WSDL interface definitions, and any accompanying XSD schemas in the sub-folders. You must ensure that the BPEL process and SwitchYard BPEL component configuration define the correct relative path for the artifacts.

Here is an example that shows the structure of the say_hello SwitchYard BPEL quickstart:

```
say_hello
    src/main/java
    src/main/resources
         META-INF
          switchyard.xml
       deploy.xml
       SayHello.bpel
       SayHelloArtifacts.wsdl
    JRE System Library [JavaSE-1.6]
    src
    pom.xml
```

2.3. PROVIDING A SERVICE WITH THE BPEL COMPONENT

Procedure 2.1.

1. Define your process using WS-BPEL within JBoss Developer Studio (with JBoss Integration and SOA Development tooling installed).
2. Define a WSDL interface for the BPEL service.
3. Define a Deployment Descriptor using the ODE Deployment Descriptor editor bundled with JBoss Tools.
4. Add the component containing the implementation and service interface to the SwitchYard configuration.

2.4. EXAMPLE OF BPEL COMPONENT CONFIGURATION

Here is an example of the component section of the SwitchYard configuration:

```
 <sca:component name="SayHelloService">
        <bpel:implementation.bpel process="sh:SayHello"/>
        <sca:service name="SayHelloService">
          <sca:interface.wsdl
interface="SayHelloArtifacts.wsdl#wsdl.porttype(SayHello)"/>
        </sca:service>
  </sca:component>
```

The BPEL component contains a single ***`implementation.bpel`*** element that identifies the fully qualified name of the BPEL process. This component may also contain one or more service elements defining the WSDL port types through which the BPEL process can be accessed.

In the packaged Switchyard application, ensure that the BPEL process associated with this fully qualified name must be present within the root folder of the distribution, along with the deployment descriptor (**`deploy.xml`**). Here is an example of the deployment descriptor for the BPEL process referenced above:

```
    <deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
     xmlns:examples="http://www.jboss.org/bpel/examples">

    <process name="examples:SayHello">
     <active>true</active>
      <retired>false</retired>
      <process-events generate="all"/>
        <provide partnerLink="client">
          <service name="examples:SayHelloService" port="SayHelloPort"/>
      </provide>
    </process>
    </deploy>
```

2.5. CONSUMING A SERVICE FROM A BPEL PROCESS

To enable a BPEL process to invoke other services, you need to define the WSDL interface representing the service to be consumed, using an invoke element within the deployment descriptor. For example, in the **`deploy.xml`** file:

```
  <process name="ls:loanApprovalProcess">
      <active>true</active>
      <process-events generate="all"/>
      <provide partnerLink="customer">
        <service name="ls:loanService" port="loanService_Port"/>
      </provide>
      <invoke partnerLink="assessor" usePeer2Peer="false">
        <service name="ra:riskAssessor" port="riskAssessor_Port"/>
      </invoke>
  </process>
```

Here, the ***`usePeer2Peer`*** property informs the BPEL engine not to use internal communications for sending messages between BPEL processes that may be executing within the same engine, and instead pass messages through the SwitchYard infrastructure.

For each consumed service, you can then create a reference element within the SwitchYard configuration to locate the WSDL file and identify the port type associated with the required WSDL service or port, as shown in the **`switchyard.xml`** file below:

```
  <sca:component name="loanService">
      <bpel:implementation.bpel process="ls:loanApprovalProcess" />
      <sca:service name="loanService">
        <sca:interface.wsdl
interface="loanServicePT.wsdl#wsdl.porttype(loanServicePT)"/>
      </sca:service>
      <sca:reference name="riskAssessor">
        <sca:interface.wsdl
interface="riskAssessmentPT.wsdl#wsdl.porttype(riskAssessmentPT)"/>
      </sca:reference>
  </sca:component>
```

2.6. PROPERTY INJECTION INTO A BPEL PROCESS

You can inject properties into your BPEL process definition by using the **`SwitchYardPropertyFunction.resolveProperty()`** XPath custom function. The ***`bpel:copy`*** section copies Greeting property value into the **`ReplySayHelloVar`** variable in example shown below:

```
  <bpel:copy>
    <bpel:from
xmlns:property="java:org.switchyard.component.bpel.riftsaw.SwitchYardPrope
rtyFunction"
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath2.0">
       <![CDATA[concat(property:resolveProperty('Greeting'),
$ReceiveSayHelloVar.parameters/tns:input)]]>
    </bpel:from>
    <bpel:to part="parameters" variable="ReplySayHelloVar">
      <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><!
[CDATA[tns:result]]></bpel:query>
    </bpel:to>
  </bpel:copy>
```

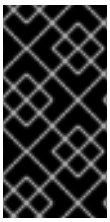
CHAPTER 3. ADMINISTERING YOUR SYSTEM WITH BPEL CONSOLE

3.1. ABOUT THE BPEL CONSOLE

The BPEL Console is a web based interface to manage, administer and debug processes deployed on a BPEL Server. It enables you to run, test and manage BPEL processes.

It allows you to view:

- any process definitions you have deployed to the BPEL engine
- the process instances executing in the BPEL engine
- a process execution history
- the query pertaining to the execution history



IMPORTANT

BPEL (based on the Riftsaw project) is no longer being actively developed and will be removed from a future release of JBoss Fuse. If you are currently using BPEL, it is recommended that you consider migrating to the Red Hat JBoss BPM Suite (which is supported through the JBoss Fuse Integration Package).

3.2. PROCESS DEFINITION

A BPEL Process Definition is an XML file that acts as a template for a process. When deployed as a part of the SwitchYard application, you can access the BPEL process through various bindings supported by SwitchYard. For instance, as a web service via the SOAP binding or via a JMS queue.

3.3. DEPLOYING PROCESS DEFINITION

The BPEL process definitions are deployed as a part of the packaged SwitchYard application. When deploying the SwitchYard application, ensure that it contains one or more BPEL process.

All the BPEL processes are displayed under the list of deployed process definitions in the console.

3.4. MANUALLY UNDEPLOY A PROCESS

Procedure 3.1. task

Execute the following to safely undeploy a process:

1. Launch a web browser and navigate to http://localhost:8080/bpel-console.
2. Enter your user name and password.
3. Click on the **`Runtime`** tab.
4. Select the **`Deployments`** option. You will now be able to see the version information and current status (active or retired) of each process definition.

5. Check if there is any active process consisting of any unfinished instances, retire the process.
6. Undeploy the process only when there is no active process instance.

WARNING

Undeploy the BPEL process only when the containing SwitchYard application is undeployed.

3.5. ACTIVE PROCESS INSTANCE

An active process instance is one execution of a process definition.

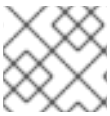
When you select a process definition, BPEL console displays the list of active process instances for that particular process version and definition.

3.6. VIEW THE PROCESS VERSION

Procedure 3.2. task

To view the process version, execute the following:

1. Launch a web browser and navigate to http://localhost:8080/bpel-console.
2. Enter your user name and password.
3. Click on the **Managed Instances** tab to see which BPEL processes are currently deployed. The process version information for each of these processes is also visible.
4. Only one version of a process can be active at a time. When you open a process definition, the active version is automatically selected.



NOTE

BPEL console enables you to select a different version of a process.

3.7. ABOUT BUSINESS PROCESS ANALYTICS FORMAT

The Business Process Analytics Format (BPAF) is designed to provide you with information about the efficiency and effectiveness of your organizational processes. It is an XML-based interchange format for all the process audit events.

3.8. VIEW THE BPAF DATA

Procedure 3.3. task

You can view the Business Process Analytics Format (BPAF) data with the BPEL console. Execute the following:

1. Launch a web browser and navigate to http://localhost:8080/bpel-console.
2. Enter your user name and password.
3. Click on the **`Managed Instances`** tab to see which BPEL processes are currently deployed. The process version information for each of these processes is also visible.
4. Select a process definition to open it.
5. Use the **`Execution History`** to produce a chart. Here you can specify a particular period of time to review and choose whether or not to include failed and terminated instances in the chart.

3.9. NAVIGATING THE EXECUTION HISTORY CHART

Table 3.1. List of Shortcut Keys to Use When Navigating the Execution History Chart

Keyboard and Mouse Command	Result
Up Arrow	Zoom In
Down Arrow	Zoom Out
Left Arrow	Half-Page Left
Right Arrow	Half-Page Right
Page-Up	Page Left
Page-Down	Page Right
TAB	Next Focus
Shift-TAB	Previous Focus
HOME	Max Zoom Out
ENTER	Max Zoom In to Focus
Mouse Drag	Scroll Chart
Shift Mouse Drag	Drag Select/Zoom
Mouse Wheel Up/Z	Zoom In
Mouse Wheel Down/X	Zoom Out
Backspace/Back Button	Back
Right Mouse-Click	Context Menu
Left-Click	Set Focus

Keyboard and Mouse Command	Result
Double-Click	Maximise Zoom-In-to-Focus

3.10. CONFIGURING LOGGING FUNCTIONALITY

Procedure 3.4. task

Execute the following to activate the BPEL Web Console's Logging Functionality:

1. Open the **`deploy.xml`** file in a text editor.

NOTE

`deploy.xml` file exists with every BPEL deployment unit.

2. Configure the **`deploy.xml`** file as follows:

```
<deploy xmlns="schemas"
    xmlns:bpl="examples"
      xmlns:intf="examples/wsdl">
        <process name="bpl:HelloGoodbye">
           <active>true</active>
           <process-events generate="all"/>
           <provide partnerLink="helloGoodbyePartnerLink">
           <service name="intf:HelloGoodbyeService"
port="HelloGoodbyePort"/>
           </provide>
        </process>
</deploy>
```

NOTE

Add the ***`process-events`*** element to generate all the events. For more information, navigate to the Apache ODE events http://ode.apache.org/ode-execution-events link.

3. Save the file and exit.

3.11. VIEW THE INSTANCE DATA

Procedure 3.5. task

Execute the following to view instance data with the BPEL Console:

1. Launch a web browser and navigate to http://localhost:8080/bpel-console.

2. Enter your user name and password.
3. Under the **Managed Instances** tab, select the **Open Button** list to view the deployed process definitions. When you select any one of the process, it displays the list of process definitions and version of that particular process.
4. Select a process instance to open it. You can see its details in the **Execution Details** panel.
5. Click the **Instance Data** button. It displays all the details about the process.
6. Click the **Execution Path** button. It opens the related instance execution graph.

NOTE

The instance execution graph is a visual representation of a running instance of a process. It tells the user about the process instance's performance over time.

7. The **View** tab shows the instance execution graph, while the **Source** tab below it shows all of the "activity" events.

3.12. VIEW A HISTORY INSTANCE QUERY

Procedure 3.6. Task

1. Log into the BPEL Web Console.
2. Choose a process definition and a process status from the list box.

   You can also optionally choose to input the correlation key, the start time and the end time as search criteria.
3. Go to the History Instances List and double-click on a row. A window will pop up showing you all of the execution events that happened when that process ran.

3.13. ACTIVE PROCESS DEFINITION

When you deploy the first version of a BPEL process definition, it automatically becomes the active definition. If this definition is subsequently changed and redeployed, then that version is "retired", and a new version becomes active automatically.

3.14. RETIRED PROCESS DEFINITION

If the active process definition is changed and redeployed, the old version is "retired". The new version automatically becomes active. The only difference between an active version and a retired one is that a retired one can no longer create new process instances. However, if there are active process instances associated with the retired process version, then these will continue to run.

3.15. MANUALLY RETIRE AN ACTIVE PROCESS DEFINITION

Procedure 3.7. Task

1. Launch a web browser and go to http://localhost:8080/bpel-console.

2. Input your user name and password.

3. Click on the **`Runtime`** tab.

4. Select the **`Deployments`** option.

   You will now be able to see the version information and current status (active or retired) of each process definition.

5. Select the particular version of the process definition you want to retire and then press the **`Retire`** button.

NOTE

If you undeploy a process, its end-points will only deactivate if no previous versions of that process have ever existed.

3.16. ENABLE EXECUTION EVENTS

The BPEL Engine generates events to let you track all the activities happening within the engine and produces detailed information about all the process executions. These events exists in the database and can be examined.

Procedure 3.8. task

When you run a process, Apache ODE generates a set of events. To achieve the satisfactory performance, you can deactivate few events that are not in use. All the events may cause a non-negligeable overhead.

To enable these events, execute the following:

1. Open the **`deploy.xml`** file in a text editor.

2. Add **`<process-events generate="all"/>`** parameter in the **`deploy.xml`** file.

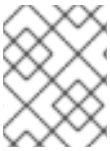
Result

Events are successfully enabled.

CHAPTER 4. CONFIGURING THE BPEL ENGINE TO RUN IN A CLUSTERED ENVIRONMENT

4.1. ABOUT BPEL ENGINE IN A CLUSTERED ENVIRONMENT

To enable the BPEL Engine to run in a clustered environment, you need to configure all the nodes within a cluster. Ensure that all the nodes have access to the shared information concerning the persistent state of process instances.



NOTE

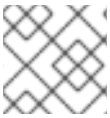
For setting up a cluster, you need to employ a load balancer to distribute the incoming SOAP requests appropriately across all of the nodes in the server.

4.2. INSTALL THE BPEL ENGINE IN A CLUSTERED ENVIRONMENT

Information Needed.

4.3. DEPLOYING A BPEL PROCESS IN A CLUSTERED ENVIRONMENT

In Fuse 6, the BPEL process is deployed as a part of a SwitchYard application. Therefore, the support for clustering is provided by SwitchYard.



NOTE

The **`switchyard.xml`** configuration file is located under the META-INF directory.

4.4. BPEL PROCESS SERVICE INVOCATION

When you invoke the BPEL service deployed in a clustered environment, make sure to specify the load balancer URL instead of SOAP address in the WSDL file. The load balancer then decides about the server, to invoke in the cluster.

CHAPTER 5. BPEL AND REST

5.1. BPEL CONSOLE RESTFUL SERVICES

5.1.1. BPEL Console RESTful Services

This is a list of Restful services that are used by the BPEL Console.

Table 5.1. BPEL Console RESTful Services

Method	Path	Description	Consumes	Produces
Server Info (General REST server information)	GET	/gwt-console-server/rs/server/status	*/*	application/json
-	GET	/gwt-console-server/rs/server/resources/{project}	*/*	text/html
Process Management(Process related data.)	GET	/gwt-console-server/rs/process/definitions	*/*	application/json
-	GET	/gwt-console-server/rs/process/definition/{id}/instances	*/*	application/json
-	GET	/gwt-console-server/rs/process/instance/{id}/dataset	*/*	text/xml
-	POST	/gwt-console-server/rs/process/instance/{id}/end/{result}	*/*	application/json
-	GET	/gwt-console-server/rs/process/definition/{id}/image	*/*	image/*
-	GET	/gwt-console-server/rs/process/definition/{id}/image/{instance}	*/*	image/*
Process Engine(Process runtime state)	GET	/gwt-console-server/rs/engine/deployments	*/*	application/json

Method	Path	Description	Consumes	Produces
Process History(Process History Service)	GET	/gwt-console-server/rs//history/definition/{id}/instances	*/*	applications/json
-	GET	/gwt-console-server/rs//history/definitions	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances	*/*	application/json
-	GET	/gwt-console-server/rs//history/instance/{id}/activities	*/*	application/json
-	GET	/gwt-console-server/rs//history/instance/{id}/events	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances/completed	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances/failed	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances/terminated	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances/chart/completed	*/*	application/json
-	GET	/gwt-console-server/rs//history/definition/{id}/instances/chart/failed	*/*	application/json

CHAPTER 6. BPEL DATABASE

6.1. BPEL DATABASE SCHEMA DIAGRAM

Following is the BPEL database schema entity relationship diagram.

Figure 6.1. BPEL Database Schema Entity Relationship Diagram