Avaya Solution & Interoperability Test Lab

Application Notes for Java Message Service (JMS) Integration between the Avaya Event Processor and JBoss Messaging - Issue 1.0

Abstract

These Application Notes describe a sample configuration for Java Message Service (JMS) integration between the Avaya Event Processor (EP) and JBoss Messaging. JBoss Messaging is an enterprise messaging platform that supports JMS and is installed on the JBoss Application Server (AS). JBoss Messaging implements a JMS provider, which makes available JMS connection factories and destinations (queues and topics) for JMS-capable applications to use in exchanging JMS messages. In the context of the JMS architecture, the Avaya EP is a JMS-capable application that accesses JMS destinations on JBoss Messaging to exchange data with other JMS-capable applications.

1. Introduction

These Application Notes describe a sample configuration for Java Message Service (JMS) integration between the Avaya Event Processor (EP) and JBoss Messaging. The JMS API is an enterprise messaging tool that allows Java 2 Enterprise Edition (J2EE) applications to exchange data reliably and asynchronously. JMS-capable applications can be created to integrate heterogeneous enterprise applications and systems in a loosely-coupled manner, allowing the applications and systems to exchange data independent of the respective native data formats.

JBoss Messaging is an enterprise messaging platform that supports JMS and is installed on the JBoss Application Server (AS). JBoss Messaging implements a JMS provider, which makes available JMS connection factories and destinations (queues and topics) for JMS-capable applications to use in exchanging JMS messages. In the context of the JMS architecture, the Avaya EP is a JMS-capable application that accesses JMS destinations on JBoss Messaging. Upon reception of a JMS message from a JMS destination on JBoss Messaging, the Avaya EP can extract relevant data from the message and insert the data into one or more Avaya EP "Streams". Similarly, the Avaya EP can extract data from one or more Avaya EP Streams and send the data in a JMS message to another JMS destination on JBoss Messaging.

2. Configuration

The sample configuration used in these Application Notes is shown in **Figure 1**.

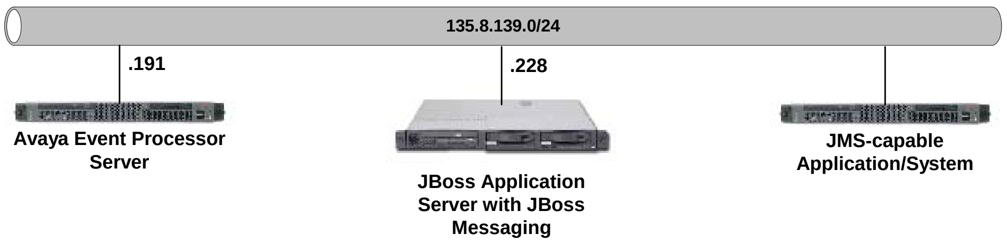
135.8.139.0/24
.191
.228
Avaya Event Processor Server
JBoss Application Server with JBoss Messaging
JMS-capable Application/System

Figure 1: Configuration

In the sample configuration, the JBoss AS and JBoss Messaging are installed on a Microsoft Windows 2003 Server, and the Avaya EP is installed on a Red Hat Enterprise Linux ES 4.0 server. JMS-capable applications and the Avaya EP exchange data by sending and receiving JMS messages to and from one or more specified JMS destinations configured on JBoss Messaging. In these Application Notes, JMS destination queues are configured on JBoss Messaging; consult the JBoss Messaging documentation [3] for guidance on configuring JMS destination topics.

3. Equipment and Software Validated

The following equipment and software were used for the sample configuration:

Equipment & Software	Version
Avaya Event Processor running on Red Hat Enterprise Linux ES Release 4.0 (Nahant Update 5) Server	2.0.110_GA_26
JBoss Application Server	4.2.2 GA
JBoss Messaging	1.4.0 SP1

Table 1: Equipment/Software List

4. Avaya Event Processor

4.1. Event Processing Language

This section describes the Event Processing Language (EPL) statements for configuring the Avaya EP to receive and send JMS messages from and to JMS destinations on JBoss Messaging. The EPL statements may be executed dynamically on a running Avaya EP server instance, or saved in EPL files. EPL files are saved in the [Avaya EP installation directory]/eplserver/[Avaya EP server instance]/engine/epl directory, and are automatically executed when the Avaya EP server instance is started. Consult the Avaya EP documentation [1][2] for further details on EPL structure and syntax, and EPL files.

Step	Description

1.	Create a JMS Pool in the Avaya EP that accesses a JMS connection factory on JBoss Messaging, using the following parameters:
	• **initialContextFactory –** 'org.jnp.interfaces.NamingContextFactory'
	• **jndiURL –** 'jnp://<IP address, hostname, or FQDN of the JBoss AS>:1099'
	• **JMSFactory –** name of JMS connection factory configured on JBoss Messaging in Section 5 Step 1.
	
	The sample EPL statement below creates a Resource of type "JMS_POOL" named "MyJMSPool" that accesses a JMS connection factory named "jms/ConnFactory1" on JBoss Messaging.
	
	`CREATE RESOURCE MyJMSPool JMS_POOL`
	`{`
	`  JMSFactory 'jms/ConnFactory1',`
	`  acknowledgeMode 'AUTO_ACKNOWLEDGE',`
	`  initialContextFactory 'org.jnp.interfaces.NamingContextFactory',`
	`  jndiURL 'jnp://135.8.139.228:1099',`
	`  password '',`
	`  poolSize 10,`
	`  resourceName 'MyJMSPool',`
	`  transacted false,`
	`  username ''`
	`}`

Step	Description

2.	Create a Stream in the Avaya EP into which data extracted from received JMS messages can be inserted. The sample EPL statement below creates a Stream named “MyJMSReceiveStream”, where each event (data record) in the Stream will consist of two fields, “ITEM” and “PRICE”. Note that each field name must match a name-value pair in the received JMS message.
	
	`CREATE STREAM system:MyJMSReceiveStream`
	`TYPE {`
	`  ITEM String,`
	`  PRICE Double`
	`}`

3.	Create a Stream Provider in the Avaya EP that receives JMS messages from a JMS destination on JBoss Messaging, and inserts the data in the JMS messages into the Stream configured in Step 2. Configure the Stream Provider with the built-in type “JMS_PROVIDER” and the following parameters:
	• **jmsDestination –** name of JMS destination queue or topic configured on JBoss Messaging in Section 5 Step 2 from which the Avaya EP will receive JMS messages.
	• **jmsDomain –** ‘QUEUE’ or ‘TOPIC’.
	• **jmsMessageType –** ‘MapMessage’.
	• **poolName –** name of JMS Pool configured in Step 1.
	• **STREAM –** name of Stream configured in Step 2.
	
	The sample EPL statement below creates a Stream Provider of type “JMS_PROVIDER” named “MyJMSReceiveStreamProvider” that receives JMS messages of type MapMessage from the JMS queue “jms/Queue1” on JBoss Messaging, and inserts the data in the JMS messages into the “MyJMSReceiveStream” Stream.
	
	`CREATE STREAMPROVIDER system:MyJMSReceiveStreamProvider JMS_PROVIDER ON`
	`STREAM system:MyJMSReceiveStream`
	`{`
	`  importAllHeaders true,`
	`  jmsDestination 'jms/Queue1',`
	`  jmsDomain 'QUEUE',`
	`  jmsMessageType 'MapMessage',`
	`  poolName 'MyJMSPool',`
	`  selector '',`
	`  useHeaderTimestamp true`
	`}`

4.	Create a Stream in the Avaya EP from which data can be extracted and sent in JMS messages. The sample EPL statement below creates a Stream named “MyJMSSendStream”, where each event (data record) in the Stream will consist of two fields, “ITEM” and “PRICE”.
	
	`CREATE STREAM system:MyJMSSendStream`
	`TYPE {`
	`  ITEM String,`
	`  PRICE Double`
	`}`

Step	Description

5.	Create a Stream Listener in the Avaya EP that listens on the Stream configured in Step 4 and sends the data in the Stream to another JMS destination on JBoss Messaging. Configure the Stream Listener with the built-in type "JMS_LISTENER" and the following parameters:
	• **jmsDestination –** name of JMS destination queue or topic configured on JBoss Messaging in Section 5.Step 3 to which the Avaya EP will send JMS messages.
	• **jmsDomain –** 'QUEUE' or 'TOPIC'.
	• **jmsMessageType –** 'MapMessage'.
	• **poolName –** name of JMS Pool configured in Step 1.
	• **STREAM –** name of Stream configured in Step 4.
	
	The sample EPL statement below creates a Stream Listener of type "JMS_LISTENER" named "MyJMSSendStreamListener" that extracts data from the "MyJMSSendStream" Stream and sends the data in JMS messages of type MapMessage to the JMS queue "jms/Queue2" on JBoss Messaging.
	
	`CREATE STREAMLISTENER system:MyJMSSendStreamListener JMS_LISTENER ON STREAM`
	`system:MyJMSSendStream`
	`{`
	`  exportAllAsHeaders false,`
	`  jmsDestination 'jms/Queue2',`
	`  jmsDomain 'QUEUE',`
	`  jmsMessageType 'MapMessage',`
	`  poolName 'MyJMSPool',`
	`  useHeaderTimestamp true`
	`}`

4.2. JAR Files and Runtime Classpath

This section lists JAR files that must be included in the Avaya EP server classpath at runtime. The JAR files must precede the "ep-classpath.jar" JAR file in the runtime classpath (see Step 3).

Step	Description
1.	Add the following JAR files to the [Avaya EP installation directory]/lib/third_party directory: • jboss-remoting.jar • log4j-1.2.15.jar (see Notes 2 and 3 below) • jboss-messaging-client.jar • jbossall-client.jar • jboss-aop-jdk50.jar • javassist.jar • trove.jar **Notes**: 1. In these Application Notes, a directory named "JBoss" was created in the [Avaya EP installation directory]/lib/third_party directory and the above JAR files were placed in the [Avaya EP installation directory]/lib/third_party/JBoss directory. 2. An Avaya-modified version of version 1.2.15 of log4j.jar must be used. Contact Avaya to obtain this JAR file. 3. This JAR file must precede jboss-messaging-client.jar in the runtime classpath (see Step 2).

Step	Description

2.	In the [Avaya EP installation directory]/bin directory, modify the “setEnv.sh” file, or copy the “setEnv.sh” file to another name, and modify as follows. The modifications are in bold below:
	
	`JBOSS_CP=$LIB/third_party/JBoss/jboss-remoting.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/log4j-1.2.15.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/jboss-messaging-client.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/jbossall-client.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/jboss-aop-jdk50.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/javassist.jar`
	`JBOSS_CP=$JBOSS_CP:$LIB/third_party/JBoss/trove.jar`
	`export JBOSS_CP`
	
	`BASE_CP=$CONF_CP:$TOOLS_CP:$EP_CP:`**`$JBOSS_CP`**`:$LIB/ep-classpath.jar`
	`export BASE_CP`
	
	Note: For the Avaya EP development version running on a Microsoft Windows server, the equivalent file is “setEnv.bat” and the equivalent modifications are in bold below:
	
	`SET JBOSS_CP=%LIB%/third_party/JBoss/jboss-remoting.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/log4j-1.2.15.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/jboss-messaging-client.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/jbossall-client.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/jboss-aop-jdk50.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/javassist.jar`
	`SET JBOSS_CP=%JBOSS_CP%;%LIB%/third_party/JBoss/trove.jar`
	
	`SET BASE_CP=%CONF_CP%;%TOOLS_CP%;%EP_CP%`**`;%JBOSS_CP%`**`;%LIB%/ep-classpath.jar`

4.3. Startup Script

The [Avaya EP installation directory]/eplserver/[Avaya EP server instance]/bin/eplServer.sh file (or eplServer.bat for the Avaya EP development version running on a Microsoft Windows server) is a script for starting an Avaya EP server instance.

Step	Description
1.	If a copied and renamed “setEnv.sh” (or “setEnv.bat”) file was used in Section 4.2 Step 2, then modify the default environment setting in the “eplServer.sh” (or “eplServer.bat”) file with the name of the renamed file. For example: Linux: . /usr/local/EP/bin/**setEnvJBoss.sh** Windows: call “C:\Program Files\Avaya\Event Processor\bin\setEnvJBoss.bat”
2.	If the Avaya EP is unable to resolve the hostname or FQDN of the JBoss AS, then add an entry with the IP address, hostname, and FQDN of the JBoss AS to the /etc/hosts (Linux) or C:\WINDOWS\system32\drivers\etc\hosts (Windows) file on the Avaya EP server.

5. JBoss Messaging

This section briefly describes the steps for configuring a JMS connection factory and a JMS destination queue on JBoss Messaging. Consult the JBoss Messaging documentation [3] for further details on configuring JMS connection factories and destination queues and topics.

Step	Description
1.	Create a JMS connection factory on JBoss Messaging by adding a new ConnectionFactory MBean configuration to the “connection-factories-service.xml” file on the JBoss AS. The example below creates a JMS connection factory named “ConnFactory1” with a JNDI binding of “jms/ConnFactory1”.

```
<mbean code="org.jboss.jms.server.connectionfactory.ConnectionFactory"
   name="jboss.messaging.connectionfactory:service=ConnFactory1" xmbean-
   dd="xmdesc/ConnectionFactory-xmbean.xml">
   <constructor>
      <arg type="java.lang.String" value="MyClientID"/>
   </constructor>
   <depends optional-attribute
      name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
   <depends optional-attribute-
      name="Connector">jboss.messaging:service=Connector,transport=bisocket</
      depends>
   <depends>jboss.messaging:service=PostOffice</depends>
   <attribute name="PrefetchSize">150</attribute>
   <attribute name="DefaultTempQueueFullSize">200000</attribute>
   <attribute name="DefaultTempQueuePageSize">2000</attribute>
   <attribute name="DefaultTempQueueDownCacheSize">2000</attribute>
   <attribute name="DupsOKBatchSize">5000</attribute>
   <attribute name="SupportsFailover">false</attribute>
   <attribute name="SupportsLoadBalancing">false</attribute>
   <attribute
      name="LoadBalancingFactory">org.jboss.jms.client.plugin.RoundRobinLoadB
      alancingFactory</attribute>
   <attribute name="StrictTck">true</attribute>
   <attribute name="JNDIBindings">
      <bindings>
         <binding>jms/ConnFactory1</binding>
      </bindings>
   </attribute>
</mbean>
```

Step	Description
2.	Create a JMS destination queue on JBoss Messaging by adding a new Queue MBean configuration to the “destinations-service.xml” file on the JBoss AS. The example below creates a JMS destination queue named “Queue1” with a JNDI name of “jms/Queue1”. In these Application Notes, the Avaya EP is a consumer of JMS messages from this JMS queue, i.e., the Avaya EP will receive JMS messages from this JMS queue.

```
<mbean code="org.jboss.jms.server.destination.QueueService"
   name="jboss.messaging.destination:service=Queue,name=Queue1" xmbean-
   dd="xmdesc/Queue-xmbean.xml">
   <depends optional-attribute
   name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
   <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="SecurityConfig">
         <security>
            <role name="guest" read="true" write="true"/>
            <role name="publisher" read="true" write="true" create="false"/>
            <role name="noacc" read="false" write="false" create="false"/>
         </security>
      </attribute>
      <attribute name="JNDIName">/jms/Queue1</attribute>
</mbean>
```

Step	Description
3.	Repeat Step 2 to create another JMS destination queue. The example below creates a JMS destination queue named “Queue2” with a JNDI name of “jms/Queue2”. In these Application Notes, the Avaya EP is a producer of JMS messages to this JMS queue, i.e., the Avaya EP will send JMS messages to this JMS queue.

```
 <mbean code="org.jboss.jms.server.destination.QueueService"
   name="jboss.messaging.destination:service=Queue,name=Queue1" xmbean-
   dd="xmdesc/Queue-xmbean.xml">
   <depends optional-attribute
   name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
   <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="SecurityConfig">
         <security>
            <role name="guest" read="true" write="true"/>
            <role name="publisher" read="true" write="true" create="false"/>
            <role name="noacc" read="false" write="false" create="false"/>
         </security>
      </attribute>
      <attribute name="JNDIName">/jms/Queue1</attribute>
</mbean>
```

6. Verification Steps

The following steps may be used to verify the configuration:

- Configure a JMS-capable application to send JMS messages to the JMS queue (e.g., “jms/Queue1”) on JBoss Messaging from which the Avaya EPL application receives JMS messages. From the JMS-capable application, send a JMS message to that JMS queue and verify that the Avaya EPL application receives the message and inserts the data into the appropriate Stream (e.g.,”MyJMSReceiveStream”).
- Configure the JMS-capable application to receive JMS messages from another JMS queue (e.g., “jms/Queue2”) on JBoss Messaging to which the Avaya EPL application sends JMS messages. Insert data into the appropriate Stream (e.g., “MyJMSSendStream”) and verify that the JMS-capable application receives a JMS message from that JMS queue containing the inserted data.

Notes:

1. The Avaya EP Console web application may be used to view data in Streams and insert data into Streams.
2. If an external JMS-capable application is not available, configure the Avaya EPL application to send and receive JMS messages to and from the same JMS queue (i.e., loopback). For example, set the “jmsDestination” parameters in the Stream Provider and Stream Listener configurations in Section 4.1 to the same JMS destination (e.g., “jms/Queue1”), and create the Event Definitions below in the Avaya EP to verify the configuration.

```
CREATE EVENT DEFINITION system:SendEvent
ON
       CLOCK.SCHEDULE(NOW(), TIME(0,0,30))
THEN
       MyJMSSendStream.INSERT(
       ITEM=‘Shades’,
       PRICE=20.95
)
```

```
CREATE EVENT DEFINITION system:ReceiveEvent
ON
       MyJMSReceiveStream
THEN
       PRINT(‘Got input: ITEM=’
       + MyJMSReceiveStream.ITEM
       + ‘ PRICE=’ + MyJMSReceiveStream.PRICE
)
```

The “SendEvent” Event Definition inserts an event (data record) into the “MyJMSSendStream” Stream every thirty seconds, and the “ReceiveEvent” Event Definition prints the contents of each received event (data record) in the “MyJMSReceiveStream” Stream. If the Stream Listener and Stream Provider in Section 4.1 are both configured to access the same JMS destination, then the following should be printed approximately every thirty seconds:

```
Got input: ITEM=Shades PRICE=20.95
```

7. Conclusion

These Application Notes described a sample configuration for Java Message Service (JMS) integration between the Avaya Event Processor (EP) and JBoss Messaging. JBoss Messaging is an enterprise messaging platform that supports JMS and is installed on the JBoss Application Server (AS). JBoss Messaging implements a JMS provider, which makes available JMS connection factories and destinations (queues and topics) for JMS-capable applications to use in exchanging JMS messages. In the context of the JMS architecture, the Avaya EP is a JMS-capable application that accesses JMS destinations on JBoss Messaging to exchange data with other JMS-capable applications.

8. Additional References

The following document may be found under the [Avaya EP installation directory]/docs directory on the Avaya EP server.

[1] "Getting Started with the Avaya Event Processor 2.0"

The following document may be obtained from http://support.avaya.com.

[2] "Avaya Event Processor 2.0 User's Guide", November 2007.

The following document may be obtained from http://labs.jboss.com.

[3] "JBoss Messaging 1.4 User's Guide", Version 1.4.0.SP1, October 31 2007.