Guide To Integrate Open-Jms And TibcoJms With Borland Application Server 6.6 (BAS-6.6)

By
Subramanian Easwaran
Borland Software Corporation
June 2006
Version 1.0

1. Introduction

The Borland Application Server 6.6, the latest release of the Borland JEE Application Server is fully compliant with the J2EE Connector Architecture Specification 1.5 (JCA 1.5). The J2EE connector Architecture has become a standard way of integrating JMS Providers with a JEE Application server. The inbound communication contract of the JCA 1.5 contract has enabled easy integration of an application server with a JMS provider using a JCA1.5 Resource Adapter like the Generic JMS Resource Adapter.

This document primarily discusses the steps involved in integrating the Open-JMS and Tibco JMS providers with the Borland Application Server 6.6 using the ***Generic JMS Resource Adapter.***

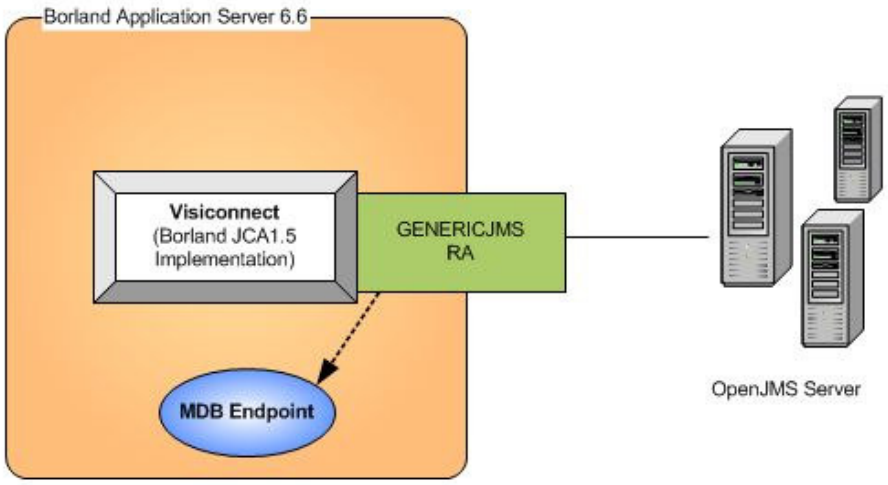

Figure 1: OpenJMS Integration using GenericJMS RA

2. Using the Borland Application Server

The Borland Application server-6.6's, container side implementation of the JCA1.5 specification contract, is called **Visiconnect.** For information on other features of BAS6.6 please refer to the following source [1]:
http://support.borland.com/entry.jspa?entryID=4639

Links to the installation instructions and how to obtain the Borland Application Server 6.6 are also provided in the above site.

3. Step-By-Step Guide to Integrate Open-JMS

Open-JMS is an Open source implementation of the Java Message Service specification. The latest release of Open-JMS (openjms-0.7.7 alpha) is compliant with the JMS 1.1 specification. Since it is still in the alpha release stage we have used the previous release of Open-JMS 0.7.6.1 (which is a JMS1.0.2 Specification implementation) in this integration example.

3.1 Open-Jms Configuration

1. Download and extract the Open-JMS 0.7.6.1 [*openjms-0.7.6.1.zip*] from http://prdownloads.sourceforge.net/openjms/
   We shall call this directory OPENJMS_HOME
2. Create the required Connection Factory and the Destinations. Edit the OPENJMS_HOME/config/openjms.xml and add the following:

```
<Configuration>

 <!-- Optional. This represents the default configuration  -->
 <ServerConfiguration host="localhost" embeddedJNDI="true" />

 <!-- Required when using an RMI connector -->
 <Connectors>
    <Connector scheme="rmi">
       <ConnectionFactories>
          <QueueConnectionFactory name="JmsQueueConnectionFactory" />
          <TopicConnectionFactory name="JmsTopicConnectionFactory" />
          <QueueConnectionFactory name="OpenJMSQCF" />
       </ConnectionFactories>
    </Connector>
 </Connectors>
...
...
<AdministeredDestinations>
...
 <AdministeredQueue name="OpenJMSQueue"/>
</AdministeredDestinations>
```

Figure 2: openjms.xml configuration

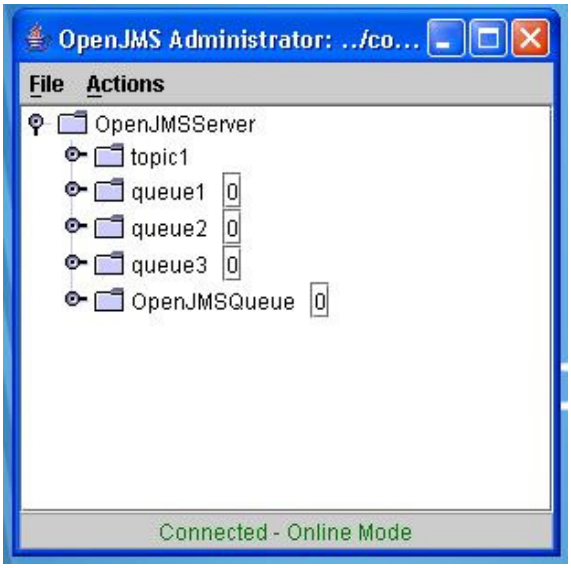

Figure 3: OpenJMS Admin console

3. Start the OpenJMS message broker as follows:
    a. OPENJMS_HOME/bin/admin.bat (or admin.sh for platforms other than windows). Ensure that JAVA_HOME is set to a valid JDK installation. The openjms admin console starts up
    b. In the admin console go to the actions tab and start the OpenJMS server. The openjms message broker starts. Now in the admin console: click Actions → Connections →Online.
    c. You should now see the queue you have added listed in the console.

3.2 GenericJMSRA Resource Adapter Configuration

1. Download and build the Generic JMS Resource Adapter latest source.
2. The GenericJMSRA was used as is except for the following change:
    a. A minor change was required to the signature of the *setSupportsXA* method in **com.sun.genericra.GenericJMSRAProperties.java**. Change the method signature from:

```
public void setSupportsXA(boolean supportsXA)
```

To

```
public void setSupportsXA(Boolean supportsXA) {
 logger.log(Level.FINEST, "setSupportsXA :" + supportsXA);
 this.supportsXA = supportsXA;
 }
```

This change is required because the BAS6.6 Appserver deployment expects only Object types in the deployment descriptor config properties.

3. Alternatively just download the latest genericra.jar binary and replace only the corrected **com.sun.genericra.GenericJMSRAProperties.class file.**
4. Configure the ra.xml
5. Configure Borland AppServer specific deployment descriptor for the connector: ra-borland.xml
6. Build the Resource Adapter Archive

The ra.xml and ra-borland.xml configuration is explained below.

3.2.1 Deployment Descriptor: ra.xml Configuration

In this example we have used the *jndi* ProviderIntegrationMode.

```
<config-property>
        <description>Provider Integration Mode</description>
        <config-property-name>ProviderIntegrationMode</config-property-name>
        <config-property-type>java.lang.String</config-property-type>
        <config-property-value>jndi</config-property-value>
</config-property>
<config-property>
        <description>OpenJMS JNDI Properties</description>
        <config-property-name>JndiProperties</config-property-name>
        <config-property-type>java.lang.String</config-property-type>
        <config-property-value>
                java.naming.factory.url.pkgs=org.exolab.jms.jndi,
                java.naming.factory.initial=org.exolab.jms.jndi.InitialContextFactory,
                java.naming.provider.url=rmi://localhost:1099
        </config-property-value>
</config-property>
```

Figure 4: “jndi” ProviderIntegrationMode Properties: **ra.xml**

NOTE: Configure the *UserName* and *Password*, config properties with the username and password values required by the OpenJMS server. Also configure other properties like SupportsXA etc. as per the requirements. Refer to the GenericJMSRA user-guide [2], for other properties that you can configure on the Resource Adapter.

Outbound Resource Adapter Configuration:
We can define connections within the <outboundresourceadapter> element. A <connection-definition> requires the following information to be configured:

- <managedconnectionfactory-class>
- <connectionfactory-interface>
- <connectionfactory-impl-class>
- <connection-interface>
- <connection-impl-class>
- ConnectionFactoryJndiName config property

For the *ConnectionFactoryJndiName* property, give the JNDI name of connection factory that was configured in the Open JMS config file.

```
<connection-definition>
…
<config-property>
        <config-property-name>ConnectionFactoryJndiName</config-property-name>
        <config-property-type>java.lang.String</config-property-type>
        <config-property-value>OpenJMSQCF</config-property-value>
</config-property>
…
</connection-definition>
```

Figure 5: ConnectionFactoryJndiName property in ra.xml connection-definition in **ra.xml**

The above listed classes and interfaces are configured as follows:

```
<connection-definition>
        <managedconnectionfactory-class>
           com.sun.genericra.outbound.ManagedQueueConnectionFactory
        </managedconnectionfactory-class>
        <connectionfactory-interface>javax.jms.QueueConnectionFactory</connectionfactory-interface>
        <connectionfactory-impl-class>com.sun.genericra.outbound.ConnectionFactory
        </connectionfactory-impl-class>
        <connection-interface>javax.jms.QueueConnection</connection-interface>
        <connection-impl-class>com.sun.genericra.outbound.ConnectionHandle</connection-impl-class>
…
```

Figure 6: Specifying connection and connection factory interfaces and implementation classes**, ra.xml**

Inbound Resource Adapter Configuration:

The inboundresourceadapter configuration is pretty simple and we just specify the listener type and the activation spec class name. E.g.:

```
<inbound-resourceadapter>
        <messageadapter>
          <messagelistener>
            <messagelistener-type>
              javax.jms.MessageListener
            </messagelistener-type>
            <activationspec>
              <activationspec-class>com.sun.genericra.inbound.ActivationSpec</activationspec-class>
            </activationspec>
          </messagelistener>
        </messageadapter>
      </inbound-resourceadapter>
```

Figure 7: Inbound resource adapter configuration **– ra.xml**

3.2.2 Borland Deployment Descriptor: ra-borland.xml Configuration

In the ra-borland.xml we define the JNDI names for the Connection Factories we have defined. We also need to define a *unique Resource Adapter name* and also a *unique factory name* for each connection definition.

```
<instance-name>OpenJMSResourceAdapter</instance-name>
<outbound-resourceadapter>
<connection-definition>
<connectionfactory-interface>javax.jms.QueueConnectionFactory</connectionfactory-interface>
<factory-name>openjmsqueueFactory</factory-name>
<jndi-name>serial://jca/openjms/qcf</jndi-name>
</connection-definition></outbound-resourceadapter>
```

Figure 8: Borland Connector Deployment Descriptor: **ra-borland.xml**

3.3 MDB – End point Configuration

If we want a Message Driven to be configured as an endpoint to receive inbound messages from an Open JMS destination through the Generic JMS Resource Adapter we need to do the following:

- Specify Endpoint activation config in the ejb-jar.xml corresponding to the MDB
- Configure the borland specific ejb-borland.xml with the Resource adapter reference.

3.3.1 ejb-jar.xml – Endpoint activation Config

In the ejb-jar.xml in the message driven bean element, we specify the activation config properties like *ConnectionFactoryJndiName* [with the Connection Factory we configured in OpenJMS] and the *DestinationJndiName* [with the physical destination in OpenJMS]. The MDB will receive messages from this specific destination through the GenericJMS resource adapter.

```
<message-driven>
        <display-name>OpenJMSMessageReceiver</display-name>
        <ejb-name>OpenJMSMessagReceiverBean</ejb-name>
        <ejb-class>com.borland.examples.ejb.mdb.OpenJMSMessageReceiverBean</ejb-class>
        <messaging-type>javax.jms.MessageListener</messaging-type>
        <transaction-type>Container</transaction-type>
        <activation-config>
          <activation-config-property>
          <activation-config-property-name>ConnectionFactoryJndiName</activation-config-property-name>
          <activation-config-property-value>OpenJMSQCF</activation-config-property-value>
          </activation-config-property>

          <activation-config-property>
          <activation-config-property-name>DestinationJndiName</activation-config-property-name>
          <activation-config-property-value>OpenJMSQueue</activation-config-property-value>
          </activation-config-property>

          <activation-config-property>
          <activation-config-property-name>DestinationType</activation-config-property-name>
          <activation-config-property-value>javax.jms.Queue</activation-config-property-value>
          </activation-config-property>

          <activation-config-property>
          <activation-config-property-name>DestinationProperties</activation-config-property-name>
          <activation-config-property-value>imqDestinationName=Queue</activation-config-property-value>
          </activation-config-property

        </activation-config>
    </message-driven>
```

Figure 9: Activation Config**: ejb-jar.xml**

3.3.2 ejb-borland.xml

For the MDB to be able to receive messages through the Resource Adapter we have to specify the resource adapter reference information in the message source of the MDB as follows:

```
<enterprise-beans>
<message-driven>
<ejb-name>OpenJMSMessageReceiverBean</ejb-name>
<message-source>
<resource-adapter-ref>
<instance-name>OpenJMSResourceAdapter</instance-name>
</resource-adapter-ref>
</message-source>
</message-driven>
</enterprise-beans>
```

Figure 10: Resource Adapter reference: **ejb-borland.xml**

Note that we have used the unique Resource adapter instance name [that was specified in the ra-borland.xml] to refer to that particular resource adapter.

3.4 Deploying and Running on Borland Application Server 6.6

In the previous section we have covered how to configure the resource adapter ra.xml and ra-borland.xml. Further we have seen how to configure the Message Driven Bean (message endpoints) to be able to receive messages from the OpenJMS destinations. In this section we will see how to package and deploy the application.

3.4.1 Packaging the Application

We have the following two alternatives for packaging the application:

1. Create an EAR file with the following contents:
   - ejb-jar archive with the ejb(MDB) classes and descriptors (ejb-jar.xml and ejb-borland.xml)
   - Resource adapter archive (.rar file) with the genericra.jar (implementation classes) and the descriptors (ra.xml and ra-borland.xml)
   - If required, a war file or an appclient which uses the Resource adapter outbound connection to send messages etc.
   - Once this packaging is done **Deploy the EAR** file to the *Borland Application server Partition*. Refer to the Borland Application Server developers guide [5] for details regarding deployment etc.

2. Create separate ejb-jars, connector archives (RARs) and genericra.jar (library) and deploy them separately.
   - In this alternative we have to be careful about certain class loader issues. To avoid any such issues we deploy the genericra.jar (with just the genericjms ra

classes) as a separate library to the Borland Application Server partition and then deploy the ejb-jar and the connector (.rar) as separate modules. Note that in this case the connector (.rar) module will just contain the deployment descriptor (META-INF/ra.xml and META-INF/ra-borland.xml) files.

3.4.2 Running the Application

When you download and install the Borland Application Server 6.6 [6], the installation comes packaged with an integrated example for OpenJMS and Tibco integration using JCA 1.5.

You can run and test the example for OpenJMS from the following location:
<BAS6.6_INSTALLATION_HOME>\examples\visiconnect\OpenJMSExample

Note: You have to add the OpenJMS Client libraries to the Borland Application Server's classpath (i.e. to the Partition's class path).
(edit the partition.config in the BAS6.6 bin directory and add the following)
Include Open JMS Client Jars
addpath C:/openjms-0.7.6.1/lib/openjms-client-0.7.6.1.jar
addpath C:/openjms-0.7.6.1/lib/commons-logging-1.0.3.jar

3. Guide to Integrate TIBCO JMS

The integration procedure for TIBCO JMS with BAS 6.6 is quite similar to the OpenJMS procedure. One could refer to the following Borland Whitepaper [4]: http://support.borland.com/servlet/KbServlet/download/4583-102-494/using_jca_to_integrate_jms_with_bas.pdf for a detailed explanation on the Tibco EMS integration procedure with BAS6.6. One could also refer to the Tibco Integration guide with the Sun Application server guide [3] for more information. In this section we briefly describe the main integration points.

- Similar to the OpenJMS case we have used the Jndi ProviderIntegrationMode for Tibco.
- The way we create the Connection factories in Tibco JMS are using the Tibco admin command prompt using the ***create factory*** command.
- We also create the Physical Destinations (queue/topic) in Tibco JMS using ***create queue*** or ***create topic*** commands.
- The rest of the configuration we do for the connector deployment descriptors and mdb (endpoint) descriptors is very similar to what was covered in the OpenJMS section.
- The BAS 6.6 installation comes with a Tibco JMS Integration example:
  See <BAS6.6_INSTALLATION_HOME>/examples/visiconnect/TibcoExample

LIST OF REFERENCES

[1] Borland Application Server 6.6 Information:
http://support.borland.com/entry.jspa?entryID=4639

[2] Generic JMS Resource Adapter User guide
https://genericjmsra.dev.java.net/docs/userguide/userguide.html

[3] Integrating Sun Java System Application Server with Tibco EMS using the Generic Resource Adapter for JMS
https://genericjmsra.dev.java.net/docs/tibco-genericjmsra-integration-sample-doc/Integrating_Sun_Java_System_Application_Server_with_Tibco_JMS_using_the_Generic_Resource_Adapter_for_JMS.html

[4] Borland Whitepaper: Using JCA To Integrate JMS with BAS: (covers the Tibco JMS Integration)
http://support.borland.com/servlet/KbServlet/download/4583-102-494/using_jca_to_integrate_jms_with_bas.pdf

[5] Borland Application Server 6.6 Developer's Guide : Available when you install the application server.

[6] Borland Application Server 6.6 Installation Guide
http://support.borland.com/entry.jspa?externalID=4581&categoryID=392