Joram – Jboss Integration

Last modified 10/11/2010 by ScalAgent D.T. release 5.4.0

Contents

1. Joram in JBoss

This chapter describes how to connect JBoss to a Joram server.

This documentation has been tested with JBoss 4.0.1 and Joram 4.3.14.

1.1. Joram installation

You can either install Joram as a set of Jars or as a single RAR.

1.1.1. Libraries Joram JARs

Copy the following JARS into the directory jboss/server/<server name>/lib:

- joram/ship/lib/JCup.jar
- joram/ship/lib/ow_monolog.jar
- joram/ship/lib/joram-shared.jar
- joram/ship/lib/joram-client.jar
- joram/ship/lib/joram-config.jar

These JARs enables JBoss to be a client of a remote Joram server.

If you need to start an embedded Joram server into JBoss then you have to use the RAR.

1.1.2. Connector Joram RAR

If the Joram server is embedded into Jboss choose the exhaustive connector joram.rar (built with the Ant target ship.adapter).

If the Joram server is remote then choose the remote connector joram.rar (built with the Ant target ship.remoteadapter).

Copy the Joram RAR into the directory jboss/server/<server name>/deploy/joram:

- joram/ship/joram.rar

1.2. MDB activation

There are two ways to activate the MDBs:

- JMSContainerInvoker
- JCA ActivationSpec

1.2.1. JMSContainerInvoker

This container implements the JMS ASF interfaces (ServerSession and ServerSessionPool) in order to activate MDBs. It also provides some more activation features like the reconnection to the JMS server or the maximum delivery number.

First, link the MDB to an invoker called “joram-mdb-invoker” in the deployment descriptor jboss.xml:

```
<message-driven>
  <ejb-name>MyMDB</ejb-name>
  <destination-jndi-name>myDestination</destination-jndi-name>
  <invoker-bindings>
     <invoker>
        <invoker-proxy-binding-name>
           joram-mdb-invoker
        </invoker-proxy-binding-name>
     </invoker>
   </invoker-bindings>
</message-driven>
```

Then define the invoker into the JBoss configuration file jboss/server/<server name>/conf/standardjboss.xml.

```
<invoker-proxy-bindings>
...
<invoker-proxy-binding>
     <name>joram-mdb-invoker</name>
     <invoker-mbean>does-not-matter</invoker-mbean>
     <proxy-factory>
         org.jboss.ejb.plugins.jms.JMSContainerInvoker
     </proxy-factory>
     <proxy-factory-config>
       <JMSProviderAdapterJNDI>JoramJMSProvider</JMSProviderAdapterJNDI>
       <ServerSessionPoolFactoryJNDI>StdJMSPool</ServerSessionPoolFactoryJNDI>
       <MinimumSize>1</MinimumSize>
       <KeepAliveMillis>30000</KeepAliveMillis>
       <MaximumSize>15</MaximumSize>
       <MaxMessages>1</MaxMessages>
      <MDBConfig>
         <ReconnectIntervalSec>10</ReconnectIntervalSec>
         <DLQConfig>
           <DestinationQueue>collector-converter-queue</DestinationQueue>
           <MaxTimesRedelivered>10</MaxTimesRedelivered>
           <TimeToLive>0</TimeToLive>
         </DLQConfig>
      </MDBConfig>
     </proxy-factory-config>
   </invoker-proxy-binding>
```

The next step is to define the JMS provider “JoramJMSProvider” in the file jboss/server/<server name>/jms/joram-jms-ds.xml

```
<server>
...
<mbean code="org.jboss.jms.jndi.JMSProviderLoader"
       name="jboss.mq:service=JoramJMSProviderLoader,name=JoramJMSProvider">
  <attribute name="ProviderName">JoramJMSProvider</attribute>
  <attribute name="ProviderAdapterClass">
     org.jboss.jms.jndi.JNDIProviderAdapter</attribute>
  <attribute name="FactoryRef">JoramXAConnectionFactory</attribute>
  <attribute name="QueueFactoryRef">JoramXAQueueConnectionFactory</attribute>
  <attribute name="TopicFactoryRef">JoramXATopicConnectionFactory</attribute>
  <attribute name="Properties">
    java.naming.factory.initial=fr.dyade.aaa.jndi2.client.NamingContextFactory
    java.naming.factory.host=localhost
    java.naming.factory.port=16400
  </attribute>
</mbean>
```

It relies on the use of XA connection factories which are resolved through the Joram JNDI. These factories must have been created and bound into Joram JNDI before starting the Jboss server.

You can create and bind these factories either manually with the Joram administration GUI (JAMT) or automatically with the XML file joramAdmin.xml.

JoramAdmin.xml example:

```
<ConnectionFactory
    className="org.objectweb.joram.client.jms.tcp.XATcpConnectionFactory">
    <tcp host="localhost"
         port="16010"/>
    <jndi name="JoramXAConnectionFactory"/>
  </ConnectionFactory>

  <ConnectionFactory
    className="org.objectweb.joram.client.jms.tcp.XATopicTcpConnectionFactory">
    <tcp host="localhost"
         port="16010"/>
    <jndi name="JoramXATopicConnectionFactory"/>
  </ConnectionFactory>

  <ConnectionFactory
     className="org.objectweb.joram.client.jms.tcp.XAQueueTcpConnectionFactory">
    <tcp host="localhost"
```

```
         port="16010"/>
    <jndi name="JoramXAQueueConnectionFactory"/>
  </ConnectionFactory>
```

Remarks:

- The default user/password is chosen by Joram connection factories: anonymous/anonymous. It seems that the JMSContainerInvoker doesn't allow to modify the user identity.

1.2.2. Joram ActivationSpec

The JCA ActivationSpec is implemented by Joram resource connector (RAR). It provides the same basic functionalities as the JMSContainerInvoker, i.e. threads and sessions pooling. The difference is that the ASF interfaces are implemented by the JCA connector and that the activation mechanisms (and roles) are specified by the JCA API.

It implies that you installed Joram with its RAR (exhaustive or remote) and not with the JARs.

However notice that the Jboss specific features (provided by the JMSContainerInvoker) are not provided by the JCA ActivationSpec. For example the reconnection policy is currently not implemented by Joram connector. The DLQ mechanism is provided by Joram but is configured in another way.

First, link the MDB to an invoker called “joram-mdb-invoker” in the deployment descriptor jboss.xml:

```
<message-driven>
  <ejb-name>MyMDB</ejb-name>
  <destination-jndi-name>myDestination</destination-jndi-name>
  <resource-adapter-name>joram.rar</resource-adapter-name>
  <invoker-bindings>
     <invoker>
        <invoker-proxy-binding-name>
           joram-mdb-invoker
        </invoker-proxy-binding-name>
     </invoker>
   </invoker-bindings>
 </message-driven>
```

The resource adapter used to activate the MDB is defined in the file jboss.xml.

Then define the invoker into the JBoss configuration file jboss/server/<server name>/conf/standardjboss.xml.

```
<invoker-proxy-binding>
      <name>joram-mdb-invoker</name>
      <invoker-mbean>default</invoker-mbean>
      <proxy-factory>
        org.jboss.ejb.plugins.inflow.JBossJMSMessageEndpointFactory
      </proxy-factory>
      <proxy-factory-config>
        <activation-config>
```

```
          ...
        </activation-config>
         <endpoint-interceptors>
           <interceptor>org.jboss.proxy.ClientMethodInterceptor</interceptor>
           <interceptor>
               org.jboss.ejb.plugins.inflow.MessageEndpointInterceptor
           </interceptor>
           <interceptor>org.jboss.proxy.TransactionInterceptor</interceptor>
           <interceptor>org.jboss.invocation.InvokerInterceptor</interceptor>
         </endpoint-interceptors>
       </proxy-factory-config>
     </invoker-proxy-binding>
```

The activation-config can be defined above or/and can be overridden into the MDB deployment descriptor ejb-jar.xml.

```
 <activation-config-property>
   <activation-config-property-name>
     destinationType</activation-config-property-name>
   <activation-config-property-value>
      javax.jms.Topic</activation-config-property-value>
</activation-config-property>
<activation-config-property>
   <activation-config-property-name>
        messageSelector</activation-config-property-name>
   <activation-config-property-value></activation-config-property-value>
</activation-config-property>
<activation-config-property>
  <activation-config-property-name>
       acknowledgeMode</activation-config-property-name>
  <activation-config-property-value>
          Auto-acknowledge</activation-config-property-value>
</activation-config-property>
<activation-config-property>
  <activation-config-property-name>
     subscriptionName</activation-config-property-name>
  <activation-config-property-value>
     mySubscription</activation-config-property-value>
</activation-config-property>
<activation-config-property>
  <activation-config-property-name>
    subscriptionDurability</activation-config-property-name>
  <activation-config-property-value>
    Durable</activation-config-property-value>
    </activation-config-property>
 <activation-config-property>
   <activation-config-property-name>
     maxNumberOfWorks</activation-config-property-name>
   <activation-config-property-value>
     0</activation-config-property-value>
 </activation-config-property>
 <activation-config-property>
    <activation-config-property-name>
         userName</activation-config-property-name>
    <activation-config-property-value>
         foo</activation-config-property-value>
  </activation-config-property>
  <activation-config-property>
  <activation-config-property-name>
      password</activation-config-property-name>
  <activation-config-property-value>
      foo</activation-config-property-value>
  </activation-config-property>
</activation-config>
```

Remarks:

- The property 'maxMessages' is currently not provided by Joram connector (always set to 1).
- The property 'maxNumberOfWorks' is actually the maximum number of sessions (maxSessions) allowed in the pool.

1.2.3. Jboss ActivationSpec

The Jboss JCA connector can be used on top of a Joram client.

You just have to configure the MDB to be activated by the jms-ra.rar adapter and add the following property in the activation configuration.

```
<activation-config-property>
  <activation-config-property-name>
         providerAdapterJNDI</activation-config-property-name>
  <activation-config-property-value>
         JoramJMSProvider</activation-config-property-value>
</activation-config-property>
```

1.3. Send and receive JMS messages

If you need to send and receive (explicitly, in a synchronous way) JMS messages then you have to get a connection factory.

There are several ways to get a Joram connection factory. You can get a non-managed one by simply querying the Joram JNDI. You can also get a managed connection factory, i.e. A connection factory that handles a pool of connections.

A Joram connection can be managed by the Jboss connector or by the Joram connector.

1.3.1. Jboss managing Joram connections

You can define in the file deploy/jms/joram-jms-ds.xml an XA connection factory called " JoramJmsXA". This is a Jboss managed connection factory (it uses Jboss jms-ra.rar). But it actually uses Joram as specified by the property 'JmsProviderAdapterJNDI'.

```
<tx-connection-factory>
  <jndi-name>JoramJmsXA</jndi-name>
  <xa-transaction/>
  <track-connection-by-tx/>
  <rar-name>jms-ra.rar</rar-name>
  <connection-definition>
    org.jboss.resource.adapter.jms.JmsConnectionFactory
  </connection-definition>
  <config-property name="SessionDefaultType"
                type="java.lang.String">javax.jms.Topic</config-property>
  <config-property name="JmsProviderAdapterJNDI"
                type="java.lang.String">java:/JoramJMSProvider</config-property>
  <config-property name="UserName" type="java.lang.String">sqp</config-property>
  <config-property name="Password" type="java.lang.String">sqp</config-property>
  <max-pool-size>20</max-pool-size>
</tx-connection-factory>
```

1.3.2. Joram managed connection factory

You can define in the file deploy/jms/joram-jms-ds.xml an XA connection factory called " JoramJmsXA". This is a Joram managed connection factory (it uses joram.rar).

```
<tx-connection-factory>
  <jndi-name>JoramJmsXA</jndi-name>
  <xa-transaction/>
  <track-connection-by-tx/>
  <rar-name>joram.rar</rar-name>
  <connection-definition>
    javax.jms.ConnectionFactory
  </connection-definition>
  <config-property name="UserName" type="java.lang.String">sqp</config-property>
  <config-property name="Password" type="java.lang.String">sqp</config-property>
  <max-pool-size>20</max-pool-size>
</tx-connection-factory>
```

1.3.3. Resolving the connection factory

You can resolve this connection factory with the following line code in your EJB:

```
ConnectionFactory cf = (ConnectionFactory)ctx.lookup("java:/JoramJmsXA");
```

1.3.4. Resolving the destinations

If the destinations have been bound into the Joram JNDI then you have to mount it into Jboss JNDI. The file deploy/joram/joram-service.xml defines a Joram JNDI external context.

```
<server>
...

<mbean code="org.jboss.naming.ExternalContext"
       name="jboss.jndi:service=ExternalContext,jndiName=external/joram">
  <attribute name="JndiName">external/joram</attribute>
  <attribute name="CacheContext">true</attribute>
  <attribute name="Properties">
    java.naming.factory.initial=fr.dyade.aaa.jndi2.client.NamingContextFactory
    java.naming.factory.host=localhost
    java.naming.factory.port=16400
  </attribute>
  <attribute name="InitialContext">javax.naming.InitialContext</attribute>
  <attribute name="RemoteAccess">true</attribute>
</mbean>
```

Remark:

- If you installed Joram with the RAR then you will have to add the Joram JNDI client JAR into jboss libraries. If you installed Joram with the JARs then the JNDI client is already packaged into joram-client.jar.

If the destination is bound in Joram JNDI with the name 'myDestination' (at the root context) then you can resolve it with the following code line:

```
Queue q = (Queue)ctx.lookup("/external/joram/myDestination");
```