Installing / Configuring Software AG’s Universal Messaging Resource Adapter on JBOSS EAP 6.x

Revision History:

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/9/2014</td>
<td>1.0</td>
<td>Initial Draft</td>
<td>Fabien Sanglier</td>
</tr>
<tr>
<td>6/24/2016</td>
<td>1.1</td>
<td>Updated with UM-provided RA + various JBOSS extra configurations</td>
<td>Fabien Sanglier</td>
</tr>
<tr>
<td>01/17/2017</td>
<td>1.2</td>
<td>Reformatting + adding new steps for extra completeness and clarity</td>
<td>Fabien Sanglier</td>
</tr>
<tr>
<td>11/10/2017</td>
<td>1.3</td>
<td>Minor corrections / additions</td>
<td>Fabien Sanglier</td>
</tr>
</tbody>
</table>

About the Author:
Fabien is Principal Technical and Solutions Architect at Software AG Government Solutions (http://www.softwareaggov.com).
Contact him at: fabien.sanglier@softwareaggov.com

Table of Contents:

Introduction.................................................................................................................3
JCA and Resource Adapters..............................................................................................3
Installing the Universal Messaging Resource Adapter..................................................4
Configuring the Universal Messaging Resource Adapter...............................................5
  Register JCA Resource Adapter .................................................................................5
  Check for any global definition of “java.naming.provider.url” ..................................5
UM resource Adapter Global Properties.......................................................................7
  Mandatory RA Settings ..............................................................................................7
  JndiProperties ..........................................................................................................7
  DeliveryType ..............................................................................................................7
Other Optional RA Settings ..........................................................................................8
UM Resource Adapter Configuration for Message Publishing (outbound) .....................8
  JCA Resource Adapter Connection Factories .............................................................8
  Object Creation ......................................................................................................8

©2017 Software AG Government Solutions. All rights reserved.
Introduction

This document outlines the steps involved in integrating SoftwareAG Universal Messaging platform with JBOSS Application Server via standard JCA Resource Adapter construct.

JCA and Resource Adapters

A Java EE Connector Architecture (JCA) resource adapter is a system-level software driver used by a Java application to connect to an Enterprise Information System (EIS). The resource adapter can be configured to use any protocol required by the EIS for connectivity. The resource adapter plugs into an application server (for example IBM Websphere Application Server, Red Hat JBOSS, or Oracle Fusion Middleware) and provides connectivity between an EIS (for example, a messaging infrastructure such as SoftwareAG Universal Messaging), the application server, and the enterprise application.

An application server that supports JCA can ensure seamless connectivity to multiple EISs. In the same way, any EIS with a JCA resource adapter can plug into an application server that supports JCA.

For details about the JCA 1.5 specification and additional JCA documentation, see:

http://www.oracle.com/technetwork/java/index.html

JCA was developed under the Java Community Process as JSR 16 (JCA 1.0), JSR 112 (JCA 1.5) and JSR 322 (JCA 1.6).
Installing the Universal Messaging Resource Adapter

Universal Messaging ships with a resource adapter package located in <UM_HOME>/j2ee/umra.rar

This Resource Adapter archive contains all the needed Universal Messaging libraries to connect your J2EE messaging application to Universal Messaging through standard JCA.

The UM Resource Adapter can be deployed to JBOSS using standard JBOSS deployment methods such as either one of the below:

- EITHER, copy to the JBOSS deployment folder (JBOSS-EAP-HOME>/standalone/deployments) and the RAR should be deployed automatically
  - If successful deployment, a file “umra.rar.deployed” should be created.
  - If nothing is created, or a file “um-jmsra-95.rar.failed” is created, an error occurred during deployment.
- OR, Use the JBOSS admin console to deploy the package just like you would do it for any other deployable resource (EAR, WAR, etc...)
  - Success or failure should be displayed in the console

You should now see the resource adapter listed in the deployed archives list in the JBOSS admin console.

From there, we will need to configure the Resource Adapter artifacts in JBOSS, as required by your application and business needs:

- Create the Resource Adapter Main Configuration parameters
- Create 1 or more Connection Factories (for message publishing and subscribing)
- Create 1 or more Admin Objects (for message publishing)

For message subscribing using the Resource Adapter, it’s usually configured in the application using Message Driven Beans and Activation Specs. We’ll go over an example later in this document.
Configuring the Universal Messaging Resource Adapter

Register JCA Resource Adapter

Using the JBOSS admin console, we first need to register the newly deployed Resource Adapter.

1) Navigate to the Resource Adapter screen (in “Profile>Connector”)

2) Create a new Resource Adapter entry, specifying the same name as the deployed adapter (umra.rar)
   a. At time of this writing (October 2017 -- UM 10.1), the UM resource adapter does not support XA transaction, so pick “not supported” in the “TX” dropdown.

3) When saved, new resource adapter should be shown in the resource adapter grid

Check for any global definition of “java.naming.provider.url”

If JBOSS application server defines the system property “java.naming.provider.url” globally for any of its internal components etc… it will introduce an issue with expected behavior of UM’s JNDI lookup.
This is because the Universal Messaging JMS client library checks for globally-set system properties first, as explained below.

Default UM JNDI Lookups flow:
1) Checks if “nirvana.provider.url” system property is defined globally, and use it if defined.
2) If not found, checks if “java.naming.provider.url” system property is defined globally, and use it if defined.
3) If both these system properties are not found, then the UM JMS client library looks at the JNDI InitialContext object for any of those 2 properties.
4) If no JNDI lookup URL for UM is found, an error will be returned.

To remediate this potential issue without having to remove JBOSS’s own usage of “java.naming.provider.url” property, 2 options are available.

**Option 1: Defines a Global Universal Messaging JNDI URL at JBOSS global level using Environment Variables / System Property**

This is useful if you want to enforce a global UM JNDI lookup URL for anything deployed on this JBOSS.

Start Jboss with following environment variable set (or add this value in the JBOSS system properties admin screen):

-Dnirvana.provider.url=<Universal Messaging JNDI url>

**IMPORTANT:** By specifying this environment variable, all UM JNDI lookups on this JBOSS will use this JNDI URL globally defined – and any JNDI lookup urls defined at the Resource Adapter level or in your Java JMS code will be essentially ignored, overridden by this setting. That may be appropriate, or not based on your requirements.

**Option2: Change the default UM JNDI Lookups flow (useful if you want to**

This is useful if you want to let applications deployed on this JBOSS decide what UM JNDI lookup URL they need.

Start Jboss with following environment variable set (or add this value in the JBOSS system properties admin screen):

-Dnirvana.provider.urlpref.sysprops=N

With this property in place, the JNDI lookups specified at the Resource Adapter level would take precedence over any globally-set JNDI system properties.

Details on JNDI lookup with this property applies:
1) JMS client library FIRST looks for any of those 2 properties (“nirvana.provider.url”, “java.naming.provider.url”) in the JMS InitialContext JNDI lookup object.
2) If these properties are NOT in the InitialContext JNDI lookup object, it checks if the “nirvana.provider.url” system property is defined globally.
3) If not, it then checks if the “java.naming.provider.url” system property is defined globally.
UM resource Adapter Global Properties

To modify the UM Resource adapter global properties:

1) Go to the resource adapter detail screen.
2) Click on Custom Properties.

For the most part, 2 properties must be updated, and some others can be optionally modified. Let’s review each group.

**Mandatory RA Settings**

**JndiProperties**

The JndiProperties property must be updated since it specifies how to connect to the UM JNDI
Set it as follow:

```
JndiProperties=java.naming.factory.initial=com.pcbsys.nirvana.nSpace.NirvanaContext
Factory,java.naming.provider.url="@UM_server_connection_url@",java.naming.security.authentication=simple
```

Where:
- Use commas ("," as the JNDI property delimiters.
- `@UM_server_connection_url@` is the UM connection url (also known as RNAMES), identified by the following pattern: protocol://host1:port1 (eg. nsp://localhost:9000)

If UM is setup in a cluster, the `@UM_server_connection_url@` URL becomes:
- **Comma separated** (protocol://host1:port1,protocol://host2:port2,etc...) to try and reach the first JNDI available in the order defined.
- **Semi-column separated** (protocol://host1:port1;protocol://host2:port2;etc...) to try and reach the first JNDI available randomly selected in that list.

**CAUTION:** Since the `@UM_server_connection_url@` can contain commas with the cluster format, be careful to surround that UM connection url with straight quotes (highlighted in red in the above example) in the to “escape” the comma characters in the UM URL.

**NOTE:** the property “java.naming.provider.url” may not be needed, based on how you have set the UM JNDI lookup ordering – see previous section

**DeliveryType**
For **DeliveryType**, at time of this writing (October 2017 -- UM 10.1) the UM Resource Adapter supports “**Synchronous**” or “**Asynchronous**”. Pick accordingly.

**Other Optional RA Settings**

A couple of other settings to optionally change based on your needs:
1) LogLevel (=FINE, FINEST, etc...)
2) UserName / Password
3) Etc...

**UM Resource Adapter Configuration for Message Publishing (outbound)**

**JCA Resource Adapter Connection Factories**

In order to send messages through the resource adapter, we will need to create a RA “connection factory” and configure it based on the actual connection factory we want to use in Universal Messaging.

**Object Creation**

1) Go to the resource adapter detail screen.
2) Click on New Connection to create a new connection factory
3) 2 values must be specified, as follows:
   a)  Jndi Name for this connection factory (which will be registered in JBOSS's JNDI and should be used by your code to lookup and acquire a reference to this managed connection factory)
      i)  Name Example: `java:/jms/MyJmsConnectionFactory`
   b)  Connection factory interface (no other choice)
      i)  `com.sun.genericra.outbound.ManagedJMSConnectionFactory`
4) Click on Apply, OK and save directly to master configuration

**Configuration**

Now the Connection Factory object is created, we need to set the inner-properties for that object.
1) Click on “Properties” for that Connection Factory, and set at least the following 2 properties:
   a)  “ConnectionFactoryJndiName”
      i)  Set to the actual name of the connection factory in UM (eg. “**MyUMConnectionFactory**”)
   b)  “ConnectionValidationEnabled” to true (will make the application server aware of any connection issues)
2) Click on save

**Connection Pooling**

Select Connection pool properties under Additional properties for the connection factory. Especially set min / max pool size as appropriate for the application.
Also, set “Flush Strategy” to “FailingConnectionOnly”.
Enabling the connection

Finally, let’s enable the connection by going back to the “Attributes” tab, clicking “edit” button, check “Enabled”, and then save (alternatively, you can select the connection factory row in the list, and press the “Enable” button on the top right corner of the UI)

If everything is successful, the connection should now show as “Enabled” in the connection list:
JCA Resource Adapter Admin Objects

In order to send messages to specific queues / topics via the UM resource adapter implementation, it’s a good idea to create the appropriate queue/topic as Admin Objects so your applications can easily lookup and acquire references to these JCA destinations. Alternatively, your code could also lookup and/or create queue/topic destinations at runtime as well.

Object Creation

1) Go to Resources-> Resource Adapters and select the Resource Adapter created.
2) Select “Admin Objects” from the top tab
3) Click New button
4) Enter the “jndi name”. For example: Jndi name=java:/jms/JmsQueueProxy
5) For the “classname” field, chose the following 2 options based on type of admin object needed:
   a) If Queue: “com.sun.genericra.outbound.QueueProxy”
   b) If Topic: “com.sun.genericra.outbound.TopicProxy”
6) Click on Apply, OK and save directly to master configuration

Configuration

Now the destination object is created, we need to set the inner-properties for that object.

1) Select the “Admin Object” just created
2) Click on “Properties” tab and set the following:
   a) DestinationJndiName=<the Jndi name of the actual UM Queue/Topic in UM>
   b) For example, if you have a queue defined in UM at “JMSSamples/SimpleQueue”, then set:
i) DestinationJndiName=JMSSamples/SimpleQueue

3) Click on Apply, OK and save directly to master configuration.
Client Application Resource Adapter-Specific Setup

UM Resource Adapter Configuration for Message Consumption (MDBs)

To consume message from a queue or topic through resource adapter construct, the J2EE application will commonly use Message Driven Beans (MDBs) that are bound to specific JCA Activation Specifications. In order to enable that functionality on JBOSS, the J2EE application will need to configure Activation Specs properties via standard configuration files like the standard EJB descriptor (ejb-jar.xml) or via the extended jboss-specific descriptor file (jboss-ejb3.xml)… As shown in example below:

```xml
<message-driven>
  <ejb-name>SimpleConsumerBean</ejb-name>
  <ejb-class>com.softwareaggov.messaging.simplejmsconsume.ejb.subscribe.SimpleConsumerBean</ejb-class>
  <activation-config>
    <activation-config-property>
      <activation-config-property-name>destinationJndiName</activation-config-property-name>
      <activation-config-property-value>JMSSamples/SimpleQueue</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>connectionFactoryJndiName</activation-config-property-name>
      <activation-config-property-value>JBOSSSHCF</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>batchSize</activation-config-property-name>
      <activation-config-property-value>1</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>maxPoolSize</activation-config-property-name>
      <activation-config-property-value>8</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>maxWaitTime</activation-config-property-name>
      <activation-config-property-value>300</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>redeliveryAttempts</activation-config-property-name>
      <activation-config-property-value>50</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>redeliveryInterval</activation-config-property-name>
      <activation-config-property-value>1</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>reconnectAttempts</activation-config-property-name>
      <activation-config-property-value>50</activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name>reconnectInterval</activation-config-property-name>
      <activation-config-property-value>20</activation-config-property-value>
    </activation-config-property>
  </activation-config>
</message-driven>
```
Set UM Resource Adapter as Application Dependency

Your application will need to make sure the UM resource adapter gets loaded and used. Refer to JBOSS documentation to create a deployment descriptors that enforces the usage of the Resource Adapter.

With JBOSS EAP 6.1, a good practice is to specify the UM Resource Adapter as a deployment dependency in the EAR jboss deployment descriptor (“META-INF/jboss-deployment-structure.xml”)

```xml
<?xml version="1.0"?>
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.2">
  <ear-subdeployments-isolated>false</ear-subdeployments-isolated>
  <deployment>
    <dependencies>
      <module name="deployment.umra.rar" optional="false" export="true"/>
    </dependencies>
    <exclusions>
      <module name="org.apache.commons.logging"/>
      <module name="org.apache.log4j"/>
      <module name="org.jboss.logging"/>
      <module name="org.jboss.logging.jul-to-slf4j-stub"/>
      <module name="org.jboss.logmanager"/>
      <module name="org.jboss.logmanager.log4j"/>
      <module name="org.slf4j"/>
      <module name="org.slf4j.impl"/>
    </exclusions>
  </deployment>
</jboss-deployment-structure>
```
Enforce the MDBs to use the UM Resource Adapter and Resource Pool

Option 1: Set at the application level using JBOSS EJB descriptor

To ensure that your application MDBs use the right UM Resource Adapter and related MDB pool, JBOSS provides an “assembly-descriptor” section in the “jboss-ejb3.xml” descriptor. Specify the UM resource adapter and resource pool to use as detailed in the following sample.

```xml
<assembly-descriptor>
  <r:resource-adapter-binding>
    <ejb-name>*</ejb-name>
    <r:resource-adapter-name>umra.rar</r:resource-adapter-name>
  </r:resource-adapter-binding>
  <p:pool>
    <ejb-name>*</ejb-name>
    <p:bean-instance-pool-ref>umra-strict-max-pool</p:bean-instance-pool-ref>
  </p:pool>
</assembly-descriptor>
```

Please refer to JBOSS documentation (eg. https://docs.jboss.org/author/display/AS71/jboss-ejb3.xml+Reference) for more details on the JBOSS EJB descriptor

Option 2: Set it at the application level using JBOSS-Specific Annotations

To take a configuration-less approach, JBOSS provides annotations to achieve the same. You can specify which resource adapter and which pool you want to use by specifying the JBOSS-specific @ResourceAdapter and @Pool annotations, as follow:

```java
@ResourceAdapter("umra.rar")
@Pool("umra-strict-max-pool")
```

The main drawback of this option is that these annotations are purely JBOSS-specific, and as such, using them makes your code / application tightly-coupled with JBOSS implementation (in other words, deploying it on other application server platform will not work as-is)

Option 3: Set it at the JBOSS global level

If appropriate, it’s also possible to enforce the use of the UM Resource Adapter at the global JBOSS level, essentially forcing all applications and MDBs deployed on this JBOSS to use the UM Resource Adapter (provided the applications don’t specify another resource adapter using application-specific descriptors or annotations as described in option 1 and 2)

To do so, we’ll edit the JBoss configuration file (standalone.xml, domain.xml) following these steps:
• Stop JBOSS if running
• Edit JBOSS configuration file (standalone.xml, domain.xml)
• Navigate to subsystem section “urn:jboss:domain:ejb3:1.4”
• In that section, there should be a “<mdb></mdb>” section.
• Edit the “resource-adapter-ref” line and set the attribute “resource-adapter-name” to the name of the UM resource adapter registered in the previous section (usually default to “umra.rar”)
• The final result should be as follow:

```xml
...<subsystem xmlns="urn:jboss:domain:ejb3:1.4">...
  <MDB>
    <resource-adapter-ref resource-adapter-name="umra.rar"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </MDB>
  ...
</subsystem>
...```

• Restart JBOSS

Now, any MDBs deployed to JBOSS will use the UM resource adapter by default.