



---

# ***Joram - WebLogic Integration***

---

# Contents

<b>Contents.....</b>	<b>2</b>
<b>1.Using Joram in WebLogic.....</b>	<b>3</b>
<b>1.1.Integrating WebLogic server with Joram.....</b>	<b>3</b>
<b>1.1.1.Installation.....</b>	<b>3</b>
<b>1.1.2.Administration and Deployment.....</b>	<b>6</b>
<b>1.1.3.JMS Module / Foreign Server.....</b>	<b>10</b>
<b>1.1.4.JNDI view.....</b>	<b>14</b>
<b>1.1.5.Patch WSL92.....</b>	<b>15</b>
<b>1.2.Test with a simple MDB.....</b>	<b>15</b>
<b>1.2.1.Build the Message-Driven Bean.....</b>	<b>16</b>
<b>1.2.2.Deploy SimpleMDB.....</b>	<b>18</b>
<b>1.2.3.Results.....</b>	<b>19</b>

# 1. Using Joram in WebLogic

This chapter describes how to use a Joram server with a BEA WebLogic server, it has been tested with WebLogic 9.2 MP2 and Joram 4.3.32.

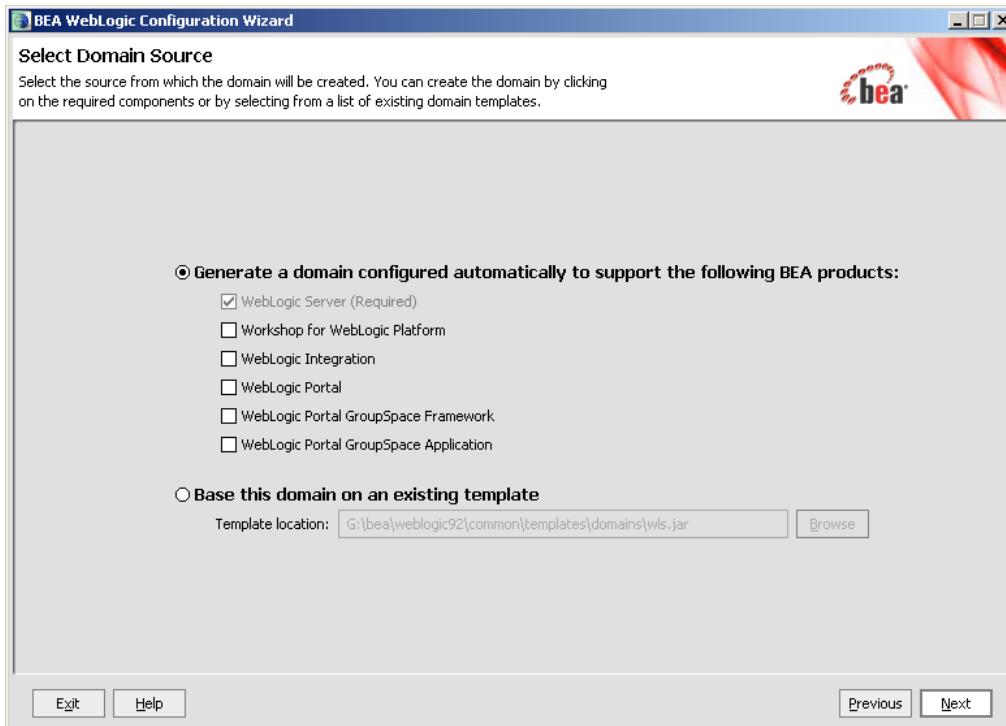
## 1.1. Integrating WebLogic server with Joram

This chapter shows how to create a new WebLogic domain and to connect it to a Joram's JMS provider.

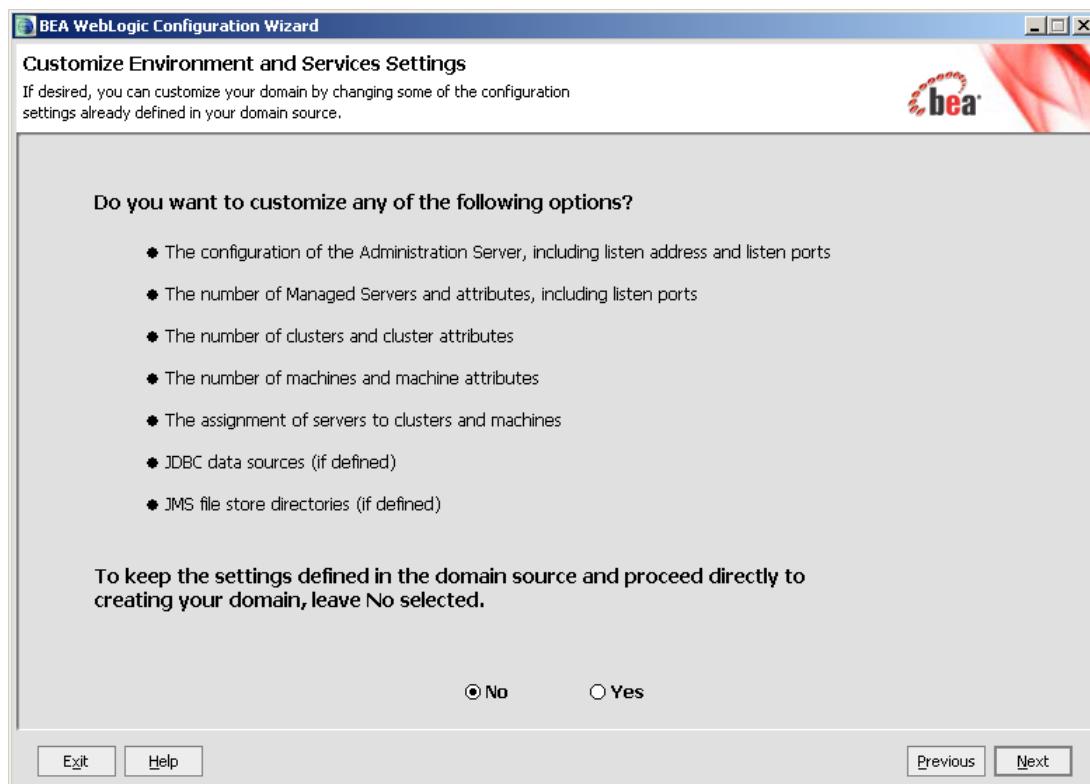
### 1.1.1. Installation

Create a simple WebLogic domain by using the wizard:

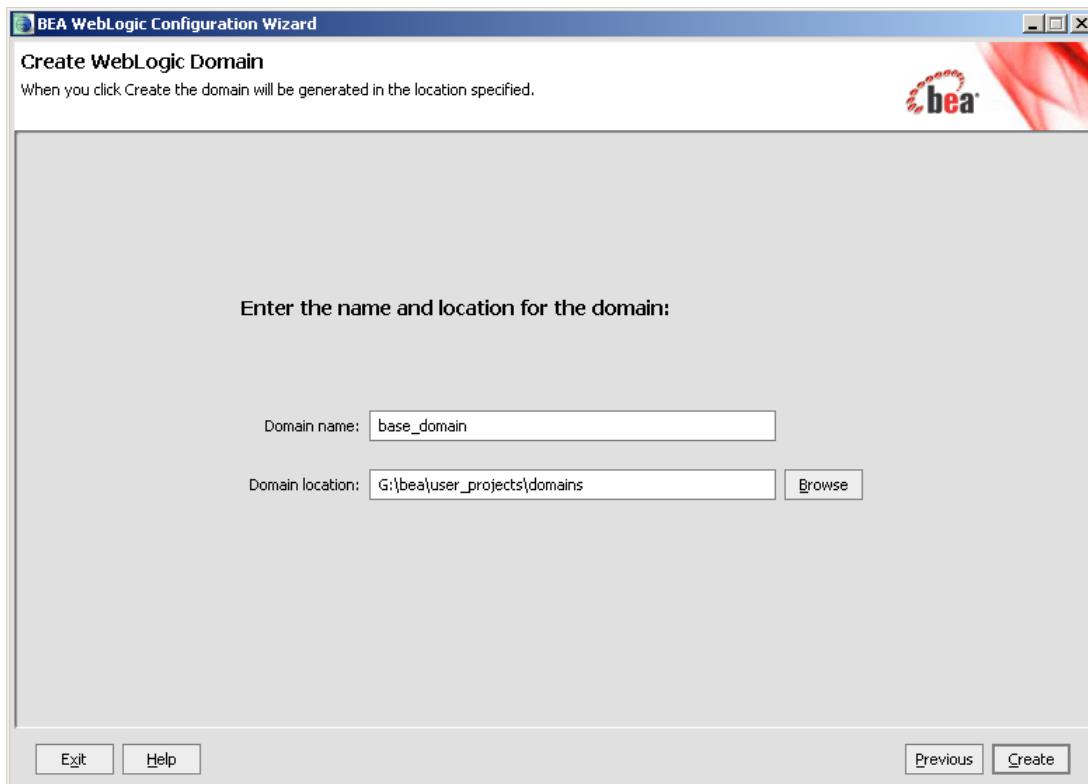
- Click **Start > All Programs > BEA Products > tools > Configuration Wizard**.
- The first form of the wizard allows to create a new domain or to extend an existing one. Choose **Create a new WebLogic domain** (default choice) then click the Next button.
- In the next form choose **Generate a domain configured automatically to support the following BEA products**, keep all the default options (see figure below) then click the Next button.



- The next form asks you to configure the administrator role. Enter the user name and password then click the **Next** button.
- The following form allow to configure server start mode and JDK : choose **Development mode** in the left pane, and select a JDK in the list of the right pane.



- The next form (see figure above) allows to customize environment and services settings of the created domain. Select **No** then click the **Next** button.
- The last form will create the WebLogic domain. Enter the name and the location where you want to install the server, in our example the domain name is **<DOMAIN\_NAME>** ('base\_domain' in our example) and the base directory **<DOMAIN\_DIR>** ('G:\bea\user\_projects\domains' in the example) then click the **Create** button.



After the above command has completed click the **Done** button, then execute the following actions:

1. Create the directory <DOMAIN\_DIR>\<DOMAIN\_NAME>\lib\joram.
  - In our example 'G:\bea\user\_projects\domains\base\_domain\lib\joram'.
2. Unzip the file **joram.rar** in the created directory.
3. Edit the file <DOMAIN\_DIR>\<DOMAIN\_NAME>\lib\joram\META-INF\MANIFEST.MF file and remove the following lines:

```
Implementation-Title: Joram
Implementation-Version: 5.0.6
Implementation-Vendor: ScalAgent D.T.
Specification-Version: 1.1
```

4. Copy the following files needed for Joram administration in the directory <DOMAIN\_DIR>\<DOMAIN\_NAME>.
  1. a3config.dtd
  2. a3debug.cfg
  3. a3servers.xml
  4. joramAdmin.xml

You are now ready to start the Joram server in WebLogic.

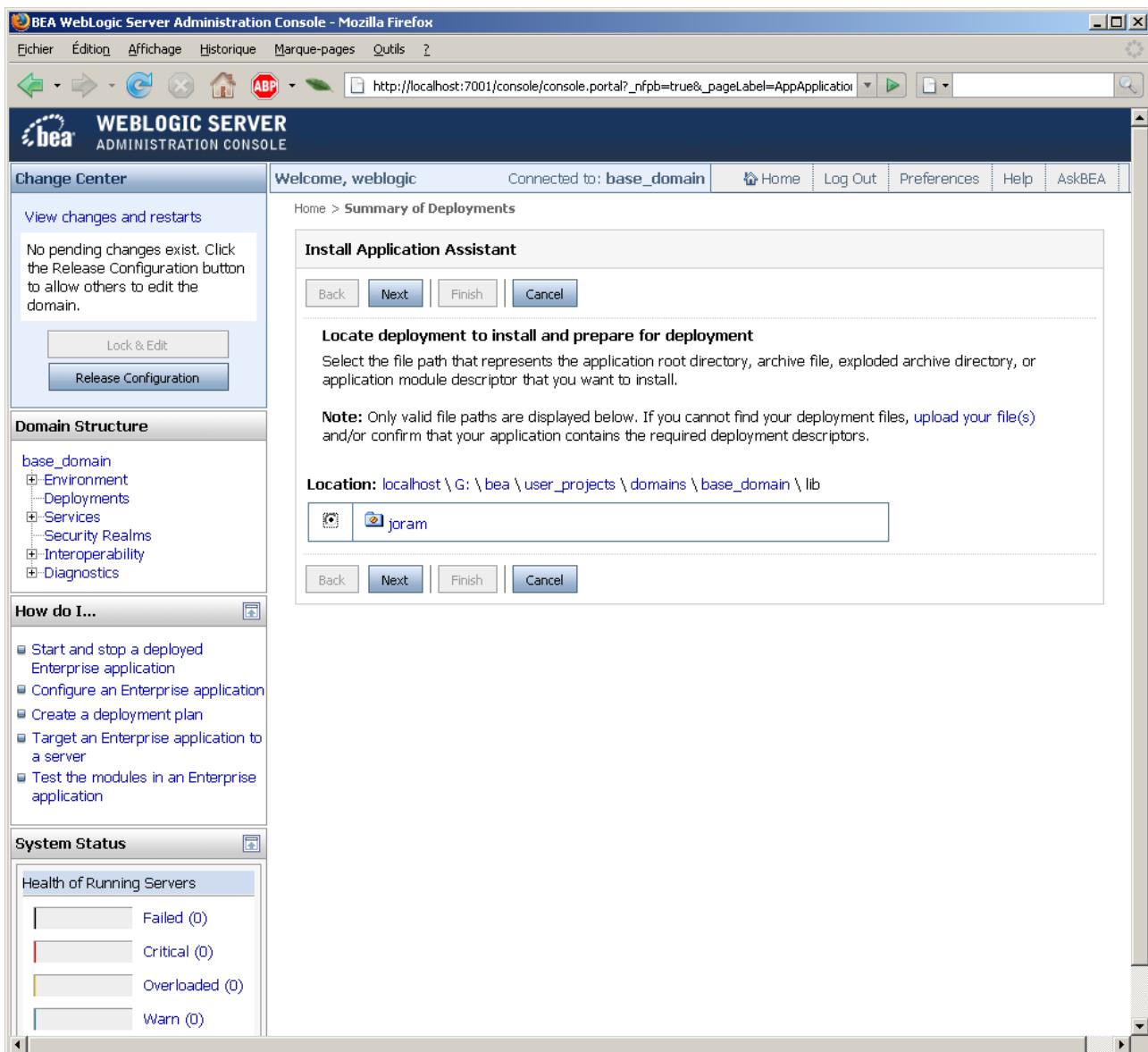
### **1.1.2. Administration and Deployment**

Start the new WebLogic domain:

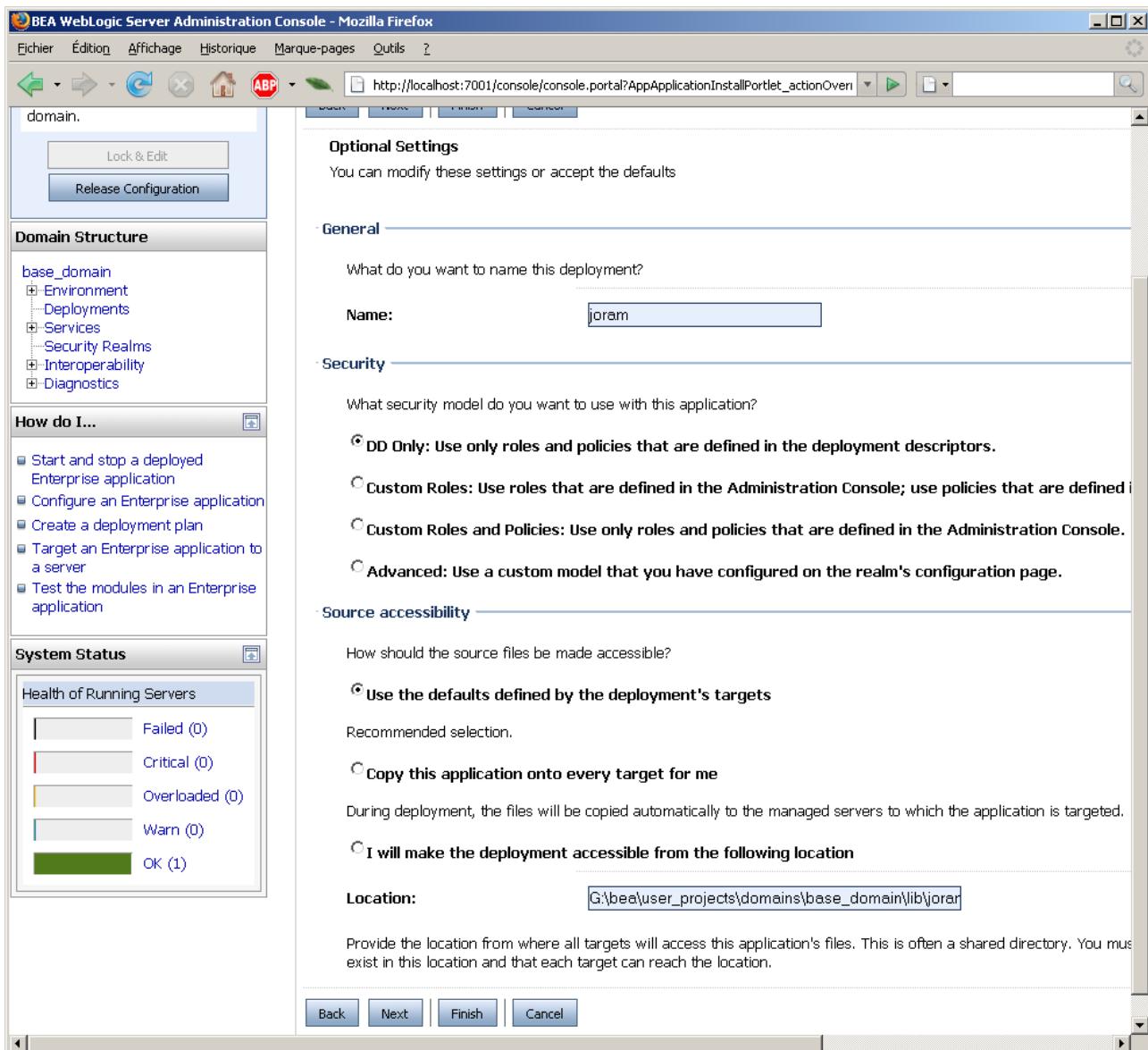
- Click **Start > All Programs > BEA Products > User Projects > <DOMAIN\_NAME>**.

Then connect to the administration console with your usual browser:

- Connect to <http://localhost:7001/console/console.portal>, enter the user name and password defined at the previous stage, then click **Log In**.
  - In the **Domain Configurations** panel select **Deployments** (under **Your Deployed Resources** subtitle).
  - Click **Lock & Edit** button in the left panel of the Administration Console.
  - Then in the **Control** tabpane of the right panel (**Summary of Deployments**) click the **Install** button; the right panel displays the **Install Application Assistant** (figure below).



- In the Right panel select the file path of the created domain:
  - <DOMAIN\_DIR>\<DOMAIN\_NAME> directory.
  - In our example 'G:\bea\user\_projects\domains\base\_domain\lib' directory.
- Then select **joram**, and click **Next**.
- In the next form choose "**Install this deployment as an application**" then click **Next**.
- The next form (see figure below) allows to fix optional settings, accept the defaults clicking **Next**, Then click **Finish**.

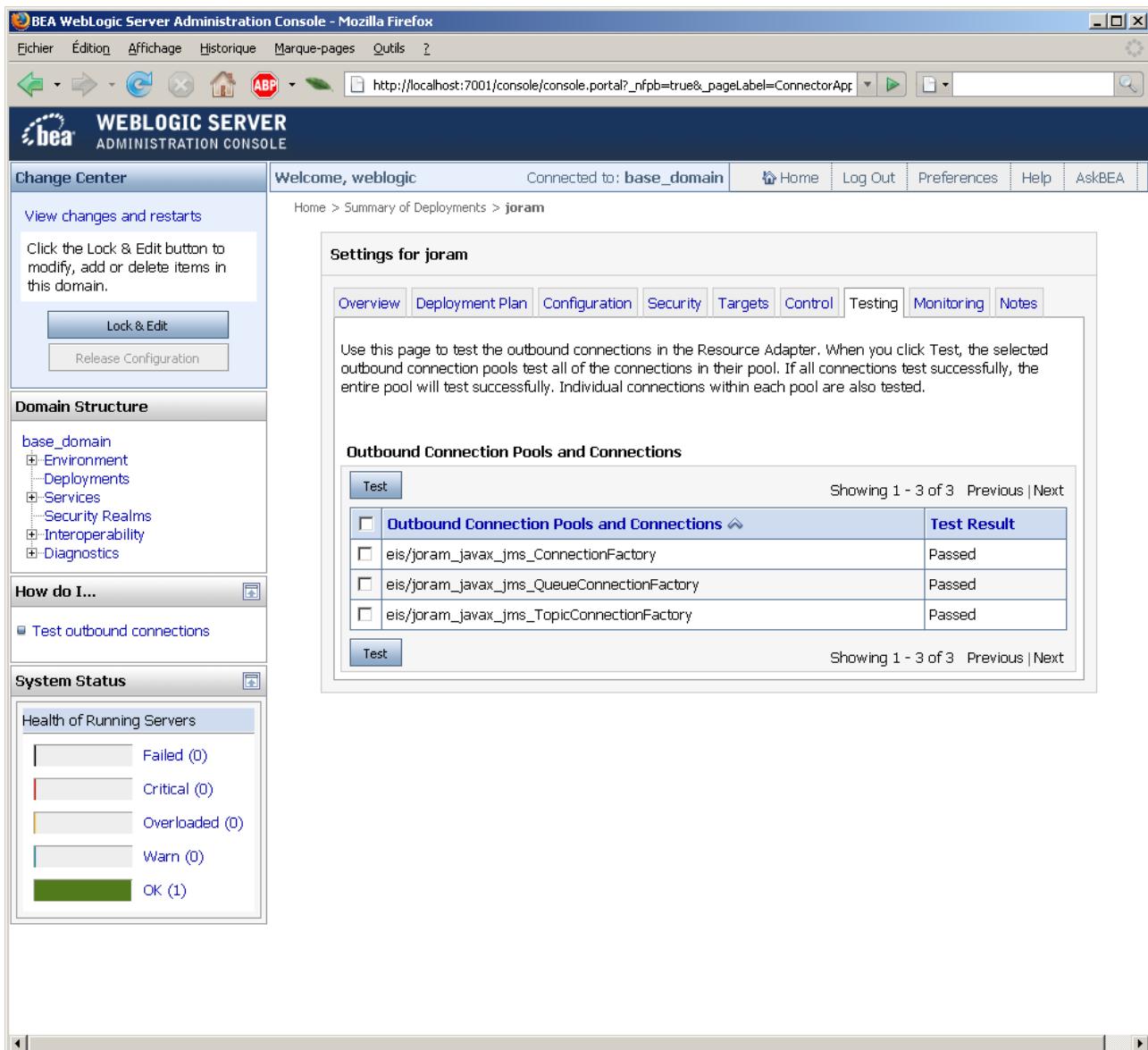


- The right panel displays basic information about the resource adapter deployment, click the **Activate Change** button in the left panel.
- In the right panel a message indicates "**All changes have been activated. No restarts are necessary**", at the top of this panel select **Summary of Deployments** in the path **Home > Summary of Deployments > joram**.
- The right panel displays (see Figure below) a summary of all modules that have been installed to this domain, select **joram** then click the **Start > servicing all requests** button.
- A confirmation is asked click the **Yes** button.

The screenshot shows the BEA WebLogic Server Administration Console interface. The left sidebar contains sections for Change Center, Domain Structure (with 'base\_domain' expanded to show Environment, Deployments, Services, Security Realms, Interoperability, and Diagnostics), How do I... (with options like Install an Enterprise application, Configure an Enterprise application, etc.), and System Status (Health of Running Servers). The main right panel displays the 'Summary of Deployments' page. At the top, it says 'Welcome, weblogic' and 'Connected to: base\_domain'. Below that is a breadcrumb trail: Home > Summary of Deployments. A tabs bar at the top of the main content area has 'Control' and 'Monitoring' selected. The main content area contains a message about the page displaying J2EE Applications and stand-alone application modules. It also says 'To install a new application or module for deployment to targets in this domain, click the Install button.' Below this is a table titled 'Deployments' showing one entry:

<input type="checkbox"/>	Name	State	Type	Deployment Order
<input type="checkbox"/>	joram	Prepared	Resource Adapter	100

- In the right panel a message indicates "**Start requests have been sent to the selected Deployments**". The Joram Resource Adapter is now Active, we will try to test it.
- In the control tabpane select select **joram**.
- The right panel now displays a summary of settings for the joram Deployments, choose the **testing** tab (see figure below), then select each outbound connection in the list and click the **Test** button. Normally the 'Test Result' column should display '**Passed**' for each outbound connection.



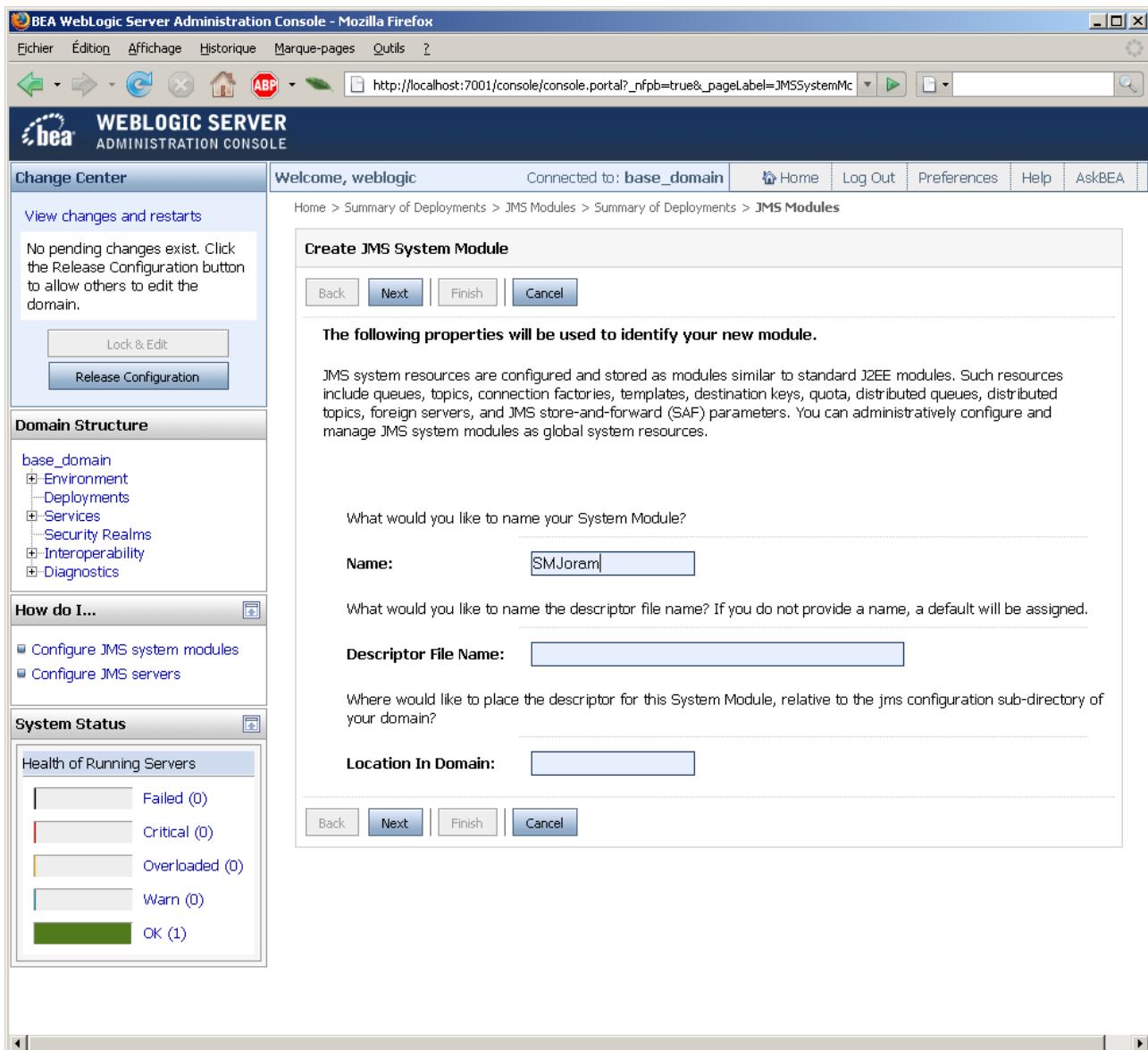
Joram RA tests view

### 1.1.3. JMS Module / Foreign Server

In order to allow a MDB to consume from a remote JMS provider we have to configure it with foreign JMS server definitions. We will now create a Foreign Server in "JMS Module" to wrap foreign factories and destinations.

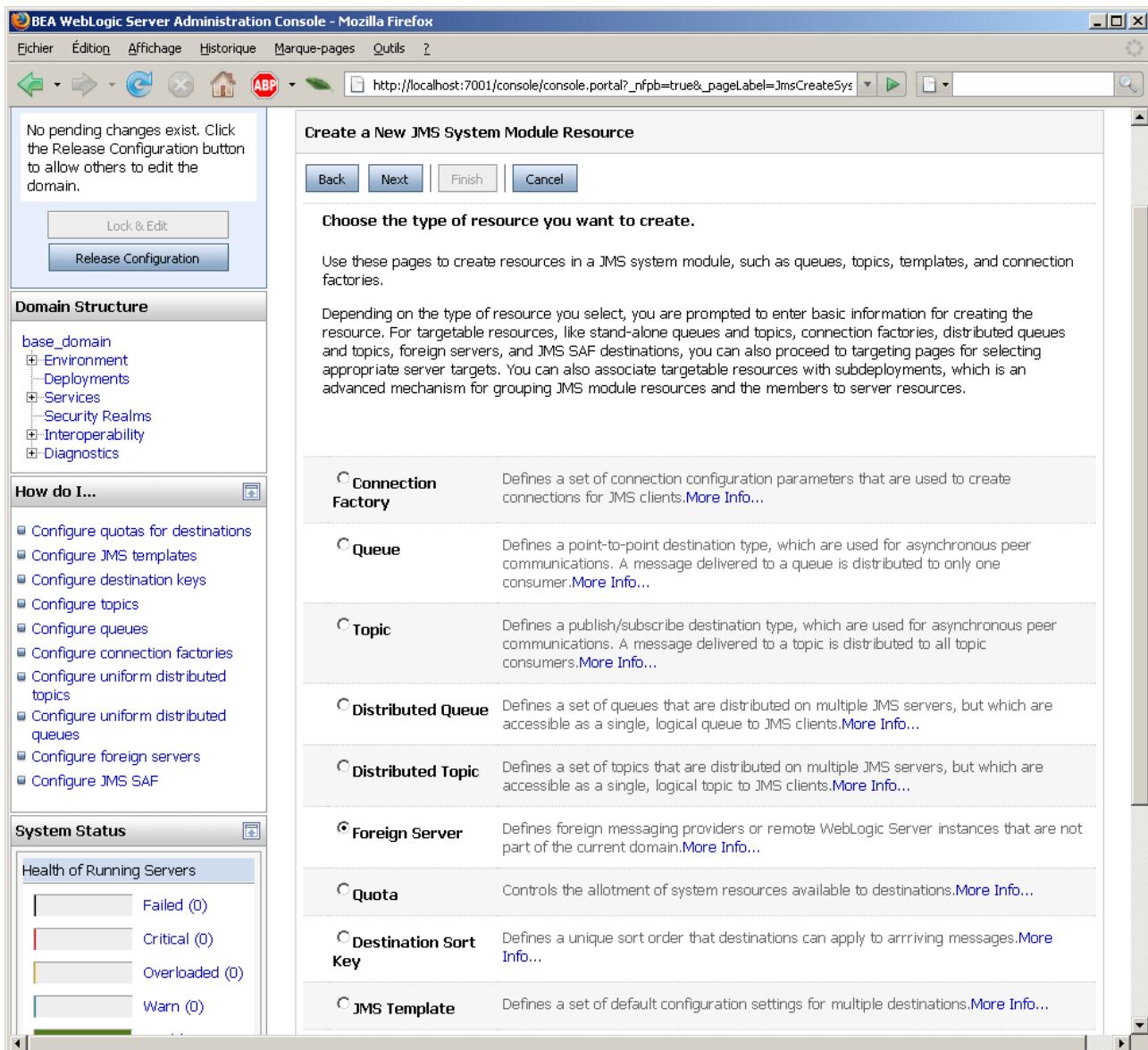
You do use wrappers, specify the Foreign Connection Factory and the Foreign Destination you set up in your local JNDI tree that corresponds the foreign JMS provider's connection factory.

- In the right panel the history path is displayed at the top of this panel: **Home > Summary of Deployments > joram...**, select **home**.
- In the **Domain Configurations** panel select **JMS Modules** (under **Messaging** subtitle).
- The next page summarizes the JMS system resources of the domain, click the **Lock & Edit** button in the left panel, then click the New button in the **JMS Modules** panel.
- The next page (figure below) shows the creation of a new JMS System Module Resource. Specify a name (**SMJoram** in our example) for the module then click the **Next** button.



### Creation of a new JMS module

- Follow the wizard selecting **AdminServer** then click the **Finish** button. At the top of the right panel a message indicates "**The JMS module was created successfully**".
- The messages at the top of the right panel indicates "**All changes have been activated. No restarts are necessary**". In the JMS Modules array select the created JMS Module (**SMJoram**) then click the **Lock & Edit** button in the left panel.
- The next page displays information about the JMS Module and its resources, in the right panel click the **New** button in **Summary of Resources**.
- In the new window select "Foreign Server" (see Figure below) and click the **Next** button.
- Give the name of the module you want to create (**ForeignServer-Joram** in our example) then click again the **Next** button, then the **Finish** one.
- The next page displays the message "**The foreign server was created successfully**", click the **Activate Change** button in the left pane.



### Configuration of a JMS module – step 1

The next view resumes the information about the new created JMS module. Now you are ready to create and configure the ConnectionFactory and Destination objects in the newly created foreign JMS provider.

- The message at the top of the next page indicates “**All changes have been activated. No restarts are necessary**”, select **ForeignServer-Joram** in the summary of resources then click the **Lock & Edit** button (see figure below).

The screenshot shows the BEA WebLogic Server Administration Console interface. The left sidebar contains tabs for 'Change Center' (selected), 'Domain Structure' (showing 'base\_domain' with sub-nodes like Environment, Deployments, Services, Security Realms, Interoperability, and Diagnostics), 'How do I...' (with links for JMS system modules, subdeployments, and resources), and 'System Status' (Health of Running Servers table with columns for Failed (0), Critical (0), Overloaded (0), Warn (0), and OK (1)). The main content area is titled 'Welcome, weblogic' and 'Connected to: base\_domain'. It shows the 'Messages' section with a green checkmark indicating 'All changes have been activated. No restarts are necessary.' Below this is the 'Settings for SMJoram' tab, which includes tabs for Configuration (selected), Subdeployments, Targets, Security, and Notes. The Configuration tab displays general information about the JMS system module, including its name ('SMJoram') and descriptor file ('jms/smjoram-jms.xml'). It also lists JMS resources such as queue and topic destinations, connection factories, JMS templates, destination sort keys, destination quota, distributed destinations, foreign servers, and store-and-forward parameters. A summary table for resources shows one entry: 'ForeignServer-Joram' (Type: Foreign Server, JNDI Name: N/A, Subdeployment: Default Targetting, Targets: AdminServer). Buttons for New and Delete are available for resource management.

## Configuration of a JMS module – step 2

- In the **Configuration** tabpane (**general** submenu) specify the JNDI Initial Context Factory and the JNDI connection URL, then click the **Save** button:
  - fr.dyade.aaa.jndi2.client.NamingContextFactory
  - scn://localhost:16400
- The next page displays the message “**Settings updated successfully**”, click the **Activate Change** button in the left pane.
- The message indicates now “**All changes have been activated. No restarts are necessary**”.
- Select **Destinations** at the top of the tabpane **Configuration** then click the **Lock & Edit** button. Click the New button in the **Foreign Destinations** array, enter the following values and click the **Ok** button:
  - **Name** = ForeignQueue
  - **Local JNDI Name** = eis/foreignQueue

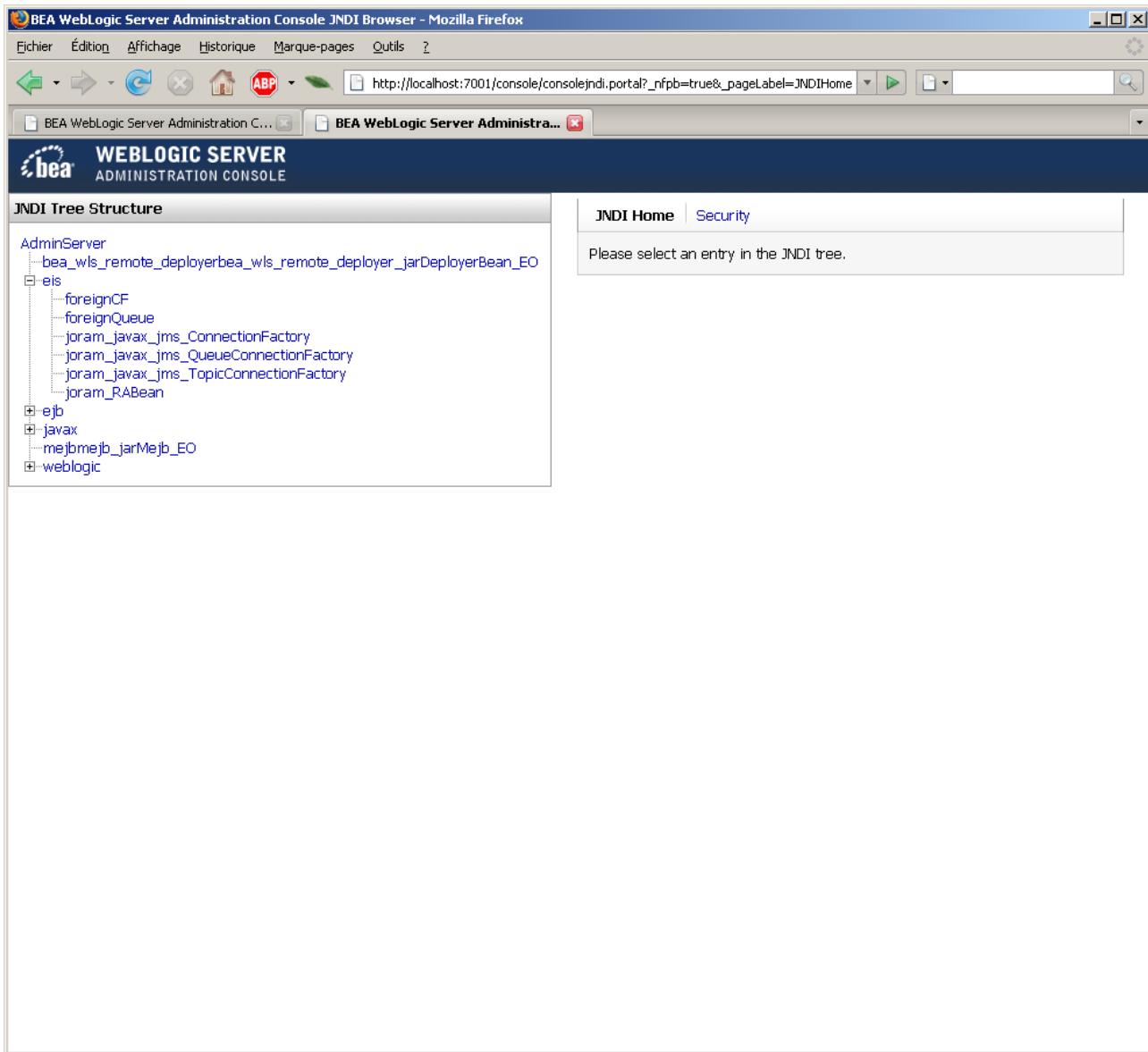
- **Remote JNDI Name** = queue
- Select **Connection Factories** at the top of the tabpane **Configuration** then click the New button in the **Foreign Connection Factories** array, enter the following values and click the **Ok** button :
  - **Name** = ForeignCF
  - **Local JNDI Name** = eis/foreignCF
  - **Remote JNDI Name** = cf
- click the **Activate Change** button in the left pane. The message indicates now “**All changes have been activated. No restarts are necessary**”.

#### 1.1.4. **JNDI view**

You can now see in JNDI the Joram Factories and Destinations (eis context). Use your usual Web browser and connect to the following URL:

- [http://localhost:7001/console/consolejndi.portal?  
\\_nfpb=true&\\_pageLabel=JNDIHomePage&server=AdminServer](http://localhost:7001/console/consolejndi.portal?_nfpb=true&_pageLabel=JNDIHomePage&server=AdminServer)

The figure above shows the administered objects at this step.



JNDI view

### 1.1.5. Patch WSL92

For testing MDB sample with **Weblogic 9.2 MP2**, you must apply the patch **7KCW**:

```
BEA_HOME\utils\bsu> bsu -prod_dir=BEA_HOME\weblogic92 -patchlist=7KCW
-patch_download_dir=BEA_HOME\utils\bsu\cache_dir -verbose -install
```

You can also use the '**Smart Update**' wizard:

- Click **Start > All Programs > BEA Products > Smart Update**.
- Then follow the instructions.

## 1.2. Test with a simple MDB

In order to test the Joram JMS module we will create and deploy a sample MDB. There are three steps:

1. Build and compile the MDB.
2. Deploy the MDB.
3. Send messages to observe the MDB behavior.

### **1.2.1. Build the Message-Driven Bean**

This simple MDB prints the JMS message content (JMS Message ID and text message) for each onMessage invocation<sup>1</sup>. The code of this sample MDB is in the figure below.

```
package mdb;

import javax.ejb.CreateException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

public class SimpleMDB implements MessageDrivenBean, MessageListener {
    private MessageDrivenContext context;

    // Required - public constructor with no argument
    public SimpleMDB() {
        System.out.println("==> SimpleMDB");
    }

    // Required - ejbRemove
    public void ejbRemove() {
        System.out.println("==> ejbRemove");
        context = null;
    }

    public void setMessageDrivenContext(MessageDrivenContext mycontext) {
        System.out.println("==> setMessageDrivenContext(" + mycontext + ')');
        context = mycontext;
    }

    // Required - ejbCreate() with no arguments
    public void ejbCreate() throws CreateException {
        System.out.println("==> ejbCreate");
    }

    // Implementation of MessageListener - throws no exceptions
    public void onMessage(Message msg) {
        try {
            System.out.println("SimpleMDB got JMS message: " + msg.getJMSMessageID());
            if (msg instanceof TextMessage)
                System.out.println("  msg = " + ((TextMessage) msg).getText());
        }
    }
}
```

<sup>1</sup> When a JMS Queue or Topic receives a message, the WebLogic Server calls the associated message-driven bean onMessage() method.

```
    } catch (Exception e) {
        // Catch any exception
        e.printStackTrace();
    }
}
```

## SimpleMDB.java

To write and compile this MDB you can use the Eclipse IDE for BEA WebLogic:

- Start > All Programs > BEA Products > Workshop for Weblogic Platform

The figures below display the XML descriptors 'ejb-jar.xml' and **weblogic-ejb-jar.xml**'

```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd" version="2.1">
  <enterprise-beans>
    <message-driven>
      <ejb-name>SimpleMDB</ejb-name>
      <ejb-class>mdb.SimpleMDB</ejb-class>
      <transaction-type>Container</transaction-type>

      <activation-config>
        <activation-config-property>
          <activation-config-property-name>destination</activation-config-
property-name>
          <activation-config-property-value>eis/foreignQueue</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>

    </message-driven>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>SimpleMDB</ejb-name>
        <method-name>*</method-name>
      </method>
      <trans-attribute>NotSupported</trans-attribute>
    </container-transaction>
    <message-destination>
```

```
<message-destination-name>destination</message-destination-name>
</message-destination>
</assembly-descriptor>
</ejb-jar>
```

**ejb-jar.xml**

```
<?xml version="1.0" encoding="UTF-8"?>

<weblogic-ejb-jar
    xmlns="http://www.bea.com/ns/weblogic/90"
    xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bea.com/ns/weblogic/90 http://www.bea.com/ns/weblogic/90/weblogic-ejb-jar.xsd">
    <weblogic-enterprise-bean>
        <ejb-name>SimpleMDB</ejb-name>
        <message-driven-descriptor>
            <pool>
                <max-beans-in-free-pool>1</max-beans-in-free-pool>
                <initial-beans-in-free-pool>1</initial-beans-in-free-pool>
            </pool>
            <destination-jndi-name>eis/foreignQueue</destination-jndi-name>
            <connection-factory-jndi-name>eis/foreignCF</connection-factory-jndi-name>
        </message-driven-descriptor>
    </weblogic-enterprise-bean>

    <message-destination-descriptor>
        <message-destination-name>destination</message-destination-name>
        <destination-jndi-name>eis/foreignQueue</destination-jndi-name>
    </message-destination-descriptor>
</weblogic-ejb-jar>
```

**weblogic-ejb-jar.xml**

### **1.2.2. Deploy SimpleMDB**

We have now to deploy the MDB in the server:

- Copy SimpleMDB.class in <DOMAIN\_DIR>\<DOMAIN\_NAME>\lib\simpleMDB\mdb, in our example: G:\bea\user\_projects\domains\base\_domain\lib\simpleMDB\mdb.
- Copy ejb-jar.xml and weblogic-ejb-jar.xml in <DOMAIN\_DIR>\<DOMAIN\_NAME>\lib\simpleMDB\META-INF, in our example: G:\bea\user\_projects\domains\base\_domain\lib\simpleMDB\ META-INF.
- Launch **Start > All Programs > BEA Products > User Projects > base\_domain > Start Admin Server for Weblogic Server Domain.**

You can then connect to the administration console with your usual browser using the following URL: <http://localhost:7001/console/console.portal>

- In the **Domain Configurations** panel select **Deployments** (under **Your Deployed Resources** subtitle).

- In the left panel, click the **Lock & Edit** button, then in the right panel click the **Install** button in the **Control** tabpane.
- In the Install Application Assistant panel select the installation path of the MDB:
  - <DOMAIN\_DIR>\<DOMAIN\_NAME>\lib directory,
  - G:\bea\user\_projects\domains\base\_domain\lib in our example.
- select **simpleMDB** and click the **Next** button.
- In the next form choose **Install this deployment as an application**" then click **Next**.
  - The next form (see figure below) allows to fix optional settings, accept the defaults clicking **Next**, Then click **Finish**.
  - The right panel displays basic information about the resource adapter deployment, click the **Activate Change** button in the left panel.
  - In the right panel a message indicates "**All changes have been activated. No restarts are necessary**", at the top of this panel select **Summary of Deployments** in the path **Home > Summary of Deployments > simpleMDB**.
  - The right panel displays a summary of all modules that have been installed to this domain, select **simpleMDB** then click the **Start > servicing all requests** button.
  - A confirmation is asked click the **Yes** button.

Now the simpleMDB waits messages from the Joram Destination named queue.

### 1.2.3. Results

You can use the sample classic to send 10 messages to this queue (see Joram user's guide).

10 messages sent.

#### Result from Classic.Sender

The console of the WebLogic server will display:

```
SimpleMDB got JMS message: ID:0.0.1026c5m0
msg = Test number 0
SimpleMDB got JMS message: ID:0.0.1026c5m1
msg = Test number 1
SimpleMDB got JMS message: ID:0.0.1026c5m2
msg = Test number 2
SimpleMDB got JMS message: ID:0.0.1026c5m3
msg = Test number 3
SimpleMDB got JMS message: ID:0.0.1026c5m4
msg = Test number 4
SimpleMDB got JMS message: ID:0.0.1026c5m5
msg = Test number 5
SimpleMDB got JMS message: ID:0.0.1026c5m6
msg = Test number 6
SimpleMDB got JMS message: ID:0.0.1026c5m7
msg = Test number 7
SimpleMDB got JMS message: ID:0.0.1026c5m8
msg = Test number 8
SimpleMDB got JMS message: ID:0.0.1026c5m9
msg = Test number 9
```

#### Results from SimpleMDB