Configuring JMS/AQ messaging between Oracle Siebel CRM and Oracle SOA Suite

An Oracle White Paper
May 2008
Introduction........................................................................................................3
Configuration.....................................................................................................4
    CRM DB - JMS/AQ Schema and Queue creation........................................4
        Create JMSUSER User ........................................................................4
        Create JMS queues ..........................................................................5
    Siebel Application JMS adapter to Siebel DB AQ ....................................6
        Installation of a Java Virtual Machine (JVM) ......................................6
        Files required on the Siebel application server .................................6
    Subsystem Setup .....................................................................................7
        Configure JMS Transport and Receiver ...........................................10
    Sending a Message .................................................................................12
    Receiving messages ..............................................................................14
    Oracle SOA Suite Application JMS Adapter to EAI DB AQ .................16
        Create jndi.properties file ...............................................................16
        Create BPEL Test stub ....................................................................17
        Configure the “Outbound” queue ......................................................18
        Configure the “Inbound” Queue .........................................................20
    Configure the Application Server for OJMS ......................................21
Conclusion........................................................................................................23
INTRODUCTION

Changing markets, increasing competitive pressures and evolving customer needs are placing greater pressure on IT to deliver greater flexibility, speed and reliability. Oracle SOA Suite, a component of Oracle Fusion Middleware, is best in class for rapidly achieving total business integration using the principles of service-oriented computing.

There are several ways to achieve connectivity between Siebel and Oracle SOA Suite. Current Integration technologies include EAI Adapters, EIM, VBC, EBC, Adapter, and Connectors.

Market trends are leaning toward SOA, Web Services and BPEL. However currently there is no web based reliable messaging mechanism between Siebel and Oracle SOA Suite that can be achieved using Oracle technology; This paper captures an approach taken to establish Siebel to Oracle SOA Suite integration using JMS and Oracle AQ.

Figure 1 – Siebel Integration capabilities
In order to configure JMS messaging using AQ between the two systems the following areas require configuration:

- Siebel CRM DB - Schema and Queue creation
- Siebel Application JMS adapter to Siebel DB AQ
  - Send a message (Outbound Connectivity)
  - Receive a message (Inbound Connectivity)
- Oracle SOA Suite Application JMS Adapter to EAI DB AQ
  - Send a message (Outbound Connectivity)
  - Receive a message (Inbound Connectivity)

CRM DB - JMS/AQ Schema and Queue creation

The first configuration steps required are to create the user and the queues on the Siebel CRM DB. While a number of deployment topologies are possible, each offering different benefits and limitations, for the purposes of this white paper a simple deployment of both the Siebel CRM system and SOA Suite accessing queues stored on the Siebel CRM database was selected:

Create JMSUSER User

The script below creates the JMSUSER database user (which will be used for queue access for this scenario). The commands must run with a user that has system privileges on the designated database (e.g. SYSTEM).

```
DROP USER jmsuser CASCADE;
CREATE USER jmsuser IDENTIFIED BY jmsuser;
GRANT CONNECT, RESOURCE,
AQ_ADMINISTRATOR_ROLE, AQ_USER_ROLE to
jmsuser;
GRANT EXECUTE ON DBMS_AQADM TO jmsuser;
GRANT EXECUTE ON DBMS_AQ TO jmsuser;
```
Create JMS queues

This script creates and starts the JMS queues that will be used to function as the source and destination, respectively, for the JMS messages being transferred from Siebel. This script should be executed as the JMSUSER created above.

```
Send Queue: JMSUSER.JMS_TEXT_QUE_OUT
Receive Queue: JMSUSER.JMS_TEXT_QUE_IN

BEGIN DBMS_AQADM.CREATE_QUEUE_TABLE
  (Queue_table => 'jmsuser.jms_qtt_text_in',
   Queue_payload_type => 'SYS.AQ$_JMS_TEXT_MESSAGE',
   compatible => '8.1.0');
END;

BEGIN DBMS_AQADM.CREATE_QUEUE (Queue_name => 'jmsuser.jms_text_que_in',
   Queue_table => 'jmsuser.jms_qtt_text_in');
END;

BEGIN DBMS_AQADM.START_QUEUE (Queue_name => 'jmsuser.jms_text_que_in');
END;

BEGIN DBMS_AQADM.CREATE_QUEUE_TABLE
  (Queue_table => 'jmsuser.jms_qtt_text_out',
   Queue_payload_type => 'SYS.AQ$_JMS_TEXT_MESSAGE',
   compatible => '8.1.0');
END;

BEGIN DBMS_AQADM.CREATE_QUEUE (Queue_name => 'jmsuser.jms_text_que_out',
   Queue_table => 'jmsuser.jms_qtt_text_out');
END;

BEGIN DBMS_AQADM.START_QUEUE (Queue_name => 'jmsuser.jms_text_que_out');
END;
```

The number and type of queues created will depend on the specific requirements of the deployment. The Siebel JMS Transport supports both TextMessage and BytesMessage JMS messages.
The following details of the Message queues involved should be noted as they are required for the next section of configuration:

- The fully qualified queue name
- Message Type (Text or Binary)
- If required, access credentials (username and password) for of the queues.

**Siebel Application JMS adapter to Siebel DB AQ**

The following are prerequisites for messaging over JMS for Siebel servers/clients:

**Installation of a Java Virtual Machine (JVM)**

A JVM must be setup on both any Siebel Servers and any Siebel Mobile/Developer Client machines (JRE 1.5.0_12 was used for this example).

Following this installation it is important to verify the location of the jvm library file, as this will be used in the configuration of the Siebel Java subsystem later. An example from the Windows Server used in this scenario is:

```plaintext
C:\PROGRA~1\Java\JDK16~1.0_0\jre\bin\server\jvm.dll
```

**Files required on the Siebel application server**

The following JAR files are required for communicating with SOA Suite.

**Siebel JMS JAR files**

- Siebel.jar
- SiebelJI_lang.jar (lang corresponds to the default language of the Siebel installation)
**Additional JAR files required for accessing the JMS provider.**

For the OC4J JMS Provider supplied with SOA Suite the following files are required.

- Jms.jar
- Jta.jar
- Oc4jclient.jar
- Optic.jar
- Javaee.jar
- Jmxri.jar
- Dms.jar
- J2ee_1.3.01.jar

**jndi.properties**

The jndi.properties file in the file system pointing to SOA Suite must be accessible. The full path of the jndi.properties file should be included in the CLASSPATH, discussed later in this document.

An example jndi.properties file is given below:

```Properties
java.naming.factory.initial=oracle.j2ee.rmi.RMIInitialContextFactory
java.naming.provider.url=ormis://10.15.35.22:12701/default
java.naming.security.principal=oc4jadmin
java.naming.security.credentials=Customer123
```

This provides the JMS subsystem on the Siebel Server with a connection point (the RMIInitialContextFactory) that it can use to look up AQ Queues and their Connection Factories dynamically.

**Subsystem Setup**

A Java subsystem must be set up on the Siebel Server(s). Login to the Siebel application and navigate to SiteMap → Administration-Server Configuration → Profile Configuration View; Create a new subsystem with the following properties:

<table>
<thead>
<tr>
<th>Profile</th>
<th>JAVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>JAVA</td>
</tr>
<tr>
<td>Subsystem Type</td>
<td>JVMSubSys</td>
</tr>
<tr>
<td>Description</td>
<td>Subsystem that includes the path to jvm dll file, Classpath &amp; JVM Options.</td>
</tr>
</tbody>
</table>
The following steps can be used to create the JVM subsystem using the Siebel Web Client.

1. Start any Siebel Business Application and navigate to Site Map → Administration → Server Configuration → Enterprises.
2. In the top list applet, select the Enterprise Server that you want to configure.
3. In the middle applet, click the Profile Configuration tab.
4. Click New to create a new component profile and set the following parameters:
   a. Profile = JAVA
   b. Alias = JAVA
   c. Subsystem Type = JVMSubsys
5. In the Profile Parameters list applet (the bottom applet), set the following values:
   a. Set the Value of the JVM Classpath parameter to contain the following:
      i. The location of the JNDI.properties
      ii. The JMS provider JAR files.
      iii. The Siebel.jar and SiebelJI_lang.jar files.
   b. Set the Value of the JVM DLL Name parameter to the path where you have the jvm.dll file installed. For example,
      
      C:\PROGRA~1\Java\JDK16~1.0_0\bin\server\jvm.dll
   c. Set the Value of the JVM Options record to any JVM-specific options that you would like to enable. For example,
      
      -Xrs -Djava.compiler=NONE

The screenshot below shows what the subsystem should look like after the configuration:

![Image of the subsystem configuration](Image)

Figure 3 - Creating JAVA – JMSSubSys on Siebel Client
When updating large numbers of servers, using Siebel Server Manager can allow administrators to create scripts to perform administrative tasks repeatedly.

Alternatively the following command-line can be used to create the JVM subsystem using Siebel Server Manager:

```
create named subsystem JAVA for subsystem JVMSubSys with
  DLL="C:\PROGRA~1\Java\JDK16-1.0_0\jre\bin\server\jvm.dll",
  CLASSPATH="d:\Siebel.jar;d:\SiebelJII.jar;d:\oc4jclient.jar;d:\jms.jar;d:\jta.jar;d:\optic.jar;d:\", VMOPTIONS ="-Xrs -Djava.compiler=NONE"
```

Verify the values of the 3 Profile Properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>JVM Classpath</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>CLASSPATH</td>
</tr>
<tr>
<td>Data Type</td>
<td>String</td>
</tr>
</tbody>
</table>
| Value            | Path to jar files separated by a ; followed by the path to the jndi.properties file followed by a ; and .
                 | c:\\\ms.jar; c:\\ta.jar; c:\\ojms.jar; c:\\optic.jar; c:\\jms.jar; c:\\jta.jar; c:\\dms.jar; c:\\\j2ee_1.3.0.jar; c:\\Siebel.jar; c:\\SiebelJII.jar; c:\\jndi.properties; |
| Description      | JVM Classpath                  |

<table>
<thead>
<tr>
<th>Name</th>
<th>JVM DLL Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>DLL</td>
</tr>
<tr>
<td>Data Type</td>
<td>String</td>
</tr>
<tr>
<td>Value</td>
<td>C:\PROGRA~1\Java\JDK16-1.0_0\jre\bin\server\jvm.dll</td>
</tr>
<tr>
<td>Description</td>
<td>JVM DLL Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>JVM Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>VMOPTIONS</td>
</tr>
<tr>
<td>Data Type</td>
<td>String</td>
</tr>
<tr>
<td>Value</td>
<td>-Xrs -Xusealtsigs -Djava.compiler=NONE -Djms.log=siebjvm.log</td>
</tr>
<tr>
<td>Description</td>
<td>JVM Options</td>
</tr>
</tbody>
</table>
Configure JMS Transport and Receiver

**Using the Siebel Server Manager (command-line)**

The following command-line can be used to create the JMS Transport using the Siebel Server Manager

```plaintext
create named subsystem Customer_JMS_Receive for subsystem JMSSubsys with 
  ConnectionFactory="java:comp/resource/SiebelJMSTestAQ/QueueConnectionFactories/myQCF",
  ReceiveQueue="java:comp/resource/SiebelJMSTestAQ/Queues/JMSUSER.JMS_TEXT_QUE_IN",
  SendQueue="java:comp/resource/SiebelJMSTestAQ/Queues/JMSUSER.JMS_TEXT_QUE_OUT",ReceiveTimeout=20000
```

The following command-line can be used to create the JMS Receiver using the Siebel Server Manager

```plaintext
create named subsystem Customer_JMS_Datahandler for subsystem EAITransportDataHandlingSubsys with 
  DispatchWorkflowProcess="Customer JMS-AQ Receive Message"
```

**Using the Siebel Web Client (GUI)**

The following procedure can be used for creating the JMS Transport subsystem using the Siebel Web Client.

1. Start any Siebel Business Application and navigate to Administration → Server Configuration → Enterprises.
2. In the top list applet, select the desired Enterprise Server that you want to configure.
3. In the middle applet, click the Profile Configuration tab.
4. Click New to create a new component profile and set the following parameters:
   a. Profile = Customer_JMS_Receive
   b. Alias = Customer_JMS_Receive
   c. Subsystem Type = JMSSubsys

5. In the Profile Parameters list applet (the bottom applet), specify the following parameters
   a. ConnectionFactory name = java:comp/resource/SiebelJMSTestAQ/QueueConnectionFactories/myQCF
   b. JVM Subsystem name = JAVA
   c. ReceiveQueue name = java:comp/resource/SiebelJMSTestAQ/Queues/JMSUSER.JMS_TEXT_QUE_IN
   d. SendQueue name = java:comp/resource/SiebelJMSTestAQ/Queues/JMSUSER.JMS_TEXT_QUE_OUT
   e. Receive Timeout = 20000

The Receiver component is configured in order to allow the JMS Transport to receive messages.
To create a JMS Receiver subsystem using the Siebel Web Client a user should follow the steps below:

1. Start any Siebel Business Application and navigate to Administration → Server Configuration → Enterprises.
2. In the top list applet, select the desired Enterprise Server that you want to configure.
3. In the middle applet, click the Profile Configuration tab.
4. Click New to create a new component profile and set the following parameters:
   a. Profile = Customer_JMS_Datahandler
   b. Alias = Customer_JMS_Datahandler
   c. Subsystem Type = EAITransportDataHandlingSubsys
5. In the Profile Parameters list applet (the bottom applet), specify the following parameters
   a. Workflow Process to Execute = Customer_JMS-AO Receive Message

The screenshot below shows what the subsystem should look like after the configuration.
Parameters

The following table lists the details of the parameters being supplied to the JMS Transport subsystem.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory</td>
<td>Connection Factory required by the JMS transport. This information is supplied by the messaging team</td>
</tr>
<tr>
<td>ReceiveQueue</td>
<td>The fully-qualified name of the queue that Siebel will receive messages from</td>
</tr>
<tr>
<td>SendQueue</td>
<td>The fully-qualified name of the queue that Siebel will send messages to</td>
</tr>
<tr>
<td>ReceiveTimeout</td>
<td>Receive Timeout (in milliseconds)</td>
</tr>
</tbody>
</table>

The following table lists the details of the parameters being supplied to the JMS Receiver.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DispatchWorkflowProcess</td>
<td>Workflow Process to Execute.</td>
</tr>
</tbody>
</table>

Sending a Message

Configure and deploy the following workflow process for sending a message out, using Siebel tools or the Siebel web client (Site Map → Administration → Business Process → Workflow Processes).

1. Define the workflow as shown in the following figure

![Figure 6 - The JMS send workflow](image-url)

This is a very simple example process used purely to “prove” the concept of transport using this method. “Real world” scenarios will likely be more complex.
2. Create the following process properties in the Process Properties applet:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>In/Out</th>
<th>Default String</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderXML</td>
<td>String</td>
<td>In</td>
<td>java:comp/resource/Siebel/JMSTestAQ/QueueConnectionFactories/myQCF</td>
<td>JNDI name of the JMS connection factory</td>
</tr>
<tr>
<td>JMSConnectionFactory</td>
<td>String</td>
<td>In</td>
<td>java:comp/resource/Siebel/JMSTestAQ/QueueConnectionFactories/myQCF</td>
<td>JNDI name of the queue</td>
</tr>
<tr>
<td>JMSSendQueue</td>
<td>String</td>
<td>In</td>
<td>JMS_USER.JMS_TEXT_QUEUE_OUT</td>
<td>JNDI name of the queue</td>
</tr>
<tr>
<td>Order Message</td>
<td>IO</td>
<td>In</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Set up the first step, after the start step, to use the “Siebel Order” ASI business service with the QueryById method to query the information from the Siebel database using the following input and output arguments:

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Type</th>
<th>Value</th>
<th>Property Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Row Id</td>
<td>Process Property</td>
<td>Object Id</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Value</th>
<th>Output Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Message</td>
<td>Output Argument</td>
<td>SiebelMessage</td>
<td></td>
</tr>
</tbody>
</table>

4. Setup the second step, after the start step, of the workflow to use the “EAI XML Converter” with the “IntObjHierToXMLDoc” method to convert the data extracted from the above step to XML. Use the following input and output arguments:

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Type</th>
<th>Value</th>
<th>Property Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiebelMessage</td>
<td>Process Property</td>
<td>Order Message</td>
<td></td>
</tr>
<tr>
<td>GenerateProcessingInstructions</td>
<td>Literal True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Value</th>
<th>Output Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderXML</td>
<td>Output Argument</td>
<td>&lt;Value&gt;</td>
<td></td>
</tr>
</tbody>
</table>
5. Set up the third step, after the start step, to use the “EAI JMS Transport” business service with the Send method using the following input and output arguments:

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Type</th>
<th>Value</th>
<th>Property Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Value&gt;</td>
<td>Process Property</td>
<td>OrderXML</td>
<td></td>
</tr>
<tr>
<td>ConnectionFactory</td>
<td>Process Property</td>
<td>JMSCOnnectionFactory</td>
<td></td>
</tr>
<tr>
<td>SendQueue</td>
<td>Process Property</td>
<td>JMSSendQueue</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Value</th>
<th>Output Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderXML</td>
<td>Output Argument</td>
<td>&lt;Value&gt;</td>
<td></td>
</tr>
</tbody>
</table>

6. The last step requires no setup.

7. Save and deploy the workflow process.

It is recommended that the workflow simulator be used for testing purposes.

**Note:** In order for this scenario to test adequately, the messaging application needs to be configured to accept the message.

**Receiving messages**

Configure and deploy the following workflow process for sending a message out, using Siebel tools or the Siebel web client (Site Map → Administration → Business Process → Workflow Processes).

1. Create a new workflow named “Customer JMS-AQ Receive Message”. This name needs to match the one specified in the “Workflow Process to Execute” property of the JMS Receiver subsystem configuration.

2. Define the workflow as shown in the following figure

![Figure 7 - Workflow executed by the Data Dispatcher component](image)

3. Create the following process properties in the Process Properties applet:
<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>In/ Out</th>
<th>Default String</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Value&gt;</td>
<td>Binary</td>
<td>In</td>
<td>&lt;Value&gt;</td>
<td>Will contain the received message</td>
</tr>
</tbody>
</table>

4. Setup the second step, after the start step, of the workflow to use the “EAI File Transport” business service with the “Send” method to write the received message to a file. Use the following input and output arguments.

<table>
<thead>
<tr>
<th>Input Argument</th>
<th>Type</th>
<th>Value</th>
<th>Property Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Value&gt;</td>
<td>Process Property</td>
<td>&quot;JMS_Incoming_&quot; + [&amp;Process Instance Id] + &quot;_xml&quot;</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>FileName</td>
<td>Expression</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. The last step requires no setup.

6. Save, deploy and activate the workflow process.

Finally, use the following command-line to start a task on Siebel server using the Siebel Server Manager:

```
start task for comp JMSReceiver with
ReceiverConnectionSubsystem=Customer_JMS_Receive,
ReceiverDataHandlingSubsystem=Customer_JMS_Datahandler,
ReceiverMethodName=ReceiveDispatch
```

As a result of the above, any messages available on the external message queue should be picked up and written to the disk. See illustrative screenshot below:

![Figure 8 - XML messages in the destination Directory](image-url)
Parameters

The following table lists the details of the parameters used in the command-line to start the receiver task on the Siebel server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReceiverConnectionSubsystem</td>
<td>The name of the JMS Transport subsystem configured earlier</td>
</tr>
<tr>
<td>ReceiverDataHandlingSubsystem</td>
<td>The name of the JMS Receiver Subsystem configured earlier</td>
</tr>
</tbody>
</table>

Oracle SOA Suite Application JMS Adapter to EAI DB AQ

This section details the steps that need to be carried out in SOA Suite Database and Apps Server in order to create a stub BPEL process.

Create jndi.properties file

The jndi.properties file contains the information required to access the JMS provider on the Oracle Application server. The client (the Siebel Application Server in this scenario) uses it in order to access the correct part of the server.

From version 10.1.3 the package names for these context factories have been renamed and you should start using oracle.j2ee.rmi.RMIInitialContextFactory

Provider URL

Incorrect provider URL is the cause of the most of the errors encountered by users.

The value for `java.naming.provider.url` should be of format `ormi://<hostname>:<ormi port>/<appName>`

The ORMI port is configured using the rmi.xml. The default port is **23791**.

The appName is the application name that you used while deploying the application and can be found in the server.xml. In the instance used for this scenario the providerURL is `ormis://10.15.35.22:12701/default`
Create BPEL Test stub

Figure 9 - BPEL test Stub

This example is based on the instructions contained in the Oracle® Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide 10g Release 3 (10.1.3.1.0), Part Number B28994-02, *Oracle Application Server Adapter for Java Message Service*.

Follow the steps in section 5.2 JMS Adapter Use Cases for Oracle BPEL. Process Manager.
Configure the “Outbound” queue

Figure 10 - SiebelJMSOut Queue

Figure 11 - SiebelJMSOut Queue

Figure 12 - SiebelJMSOut Queue
Configure the “Inbound” Queue

Figure 16 - SiebelJMSIn Queue JMS Adapter

Figure 17 - SiebelJMSIn Queue JMS Adapter

Figure 18 - SiebelJMSIn Queue JMS Adapter
Configure the Application Server for OJMS

Configure the OJMS provider within the resource-provider element in the global application.xml file (in $SOA_HOME/ j2ee/<OC4J container>/config/). It is possible to configure the resource provider with a URL property. The following demonstrates a URL configuration:

Replace the following tokens with the correct name for the scenario

- `<ResourceProvider>`, eg. OjmsSiebel
- `<QueueConnectionName>`, eg. SiebelDB
- `<connectionDetails>`, eg. localhost:1521:my
In the oc4j-ra.xml file (in SSO_HOME/j2ee/OC4J/container)/application-deployments/default/JmsAdapter/), add the following code segments:

```xml
<resource-provider class="oracle.jms.OjmsContext" name="ResourceProvider">
    <description>OJMS/AQ</description>
    <property name="url" value="jdbc:oracle:thin:@connectionDetails" />
    <property name="username" value="jmsuser" />
    <property name="password" value="jmsuser" />
</resource-provider>
```

Restart the OC4J container concerned before running the test to clear the current container cache.

When the installation has been completed successfully the example BPEL process will pick up this message and deliver it to the simulated Siebel Inbound queue.

**Figure 21 - The BPEL stub instance**

**Note:** It is essential to ensure that all the database character sets are set the same otherwise you may encounter several encoding and formatting issues with the messages.
CONCLUSION

This document provided a process for enabling communication between the Siebel application and Oracle SOA Suite using Oracle AQ as a persistent store for messages and JMS as the transport mechanism.

The steps outlined should allow a user competent in both Siebel administration and SOA Suite administration to set up a full end to end process, alternatively the tasks can be divided between the database, SOA Suite and Siebel application in order to make best use of existing skill sets.