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

An Oracle White Paper
May 2008

ORACLE
SIEBEL
CUSTOMER RELATIONSHIP MANAGEMENT

ORACLE

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

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

Reliable Messaging is a cornerstone of many companies' integration strategies.

This paper documents an approach to provide reliable messaging between Siebel CRM 7.8 (or later) and Oracle SOA Suite 10.1.3.3 using JMS and Oracle AQ.

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.

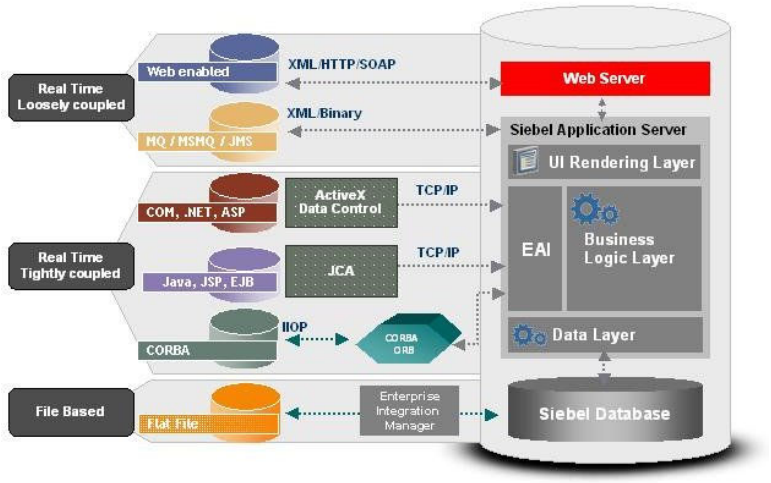

Figure 1 – Siebel Integration capabilities

System	Version
Siebel	Siebel 7.8.2.7
Siebel DB	Oracle 10.2.0.3
Siebel DB OS	Sun Solaris 8.0
Siebel Server OS	Microsoft Windows 2003 SP1
Oracle SOA Suite	10.1.3.3.0 or 10.1.3.3.1
Oracle SOA Suite DB	Oracle 10.2.0.3
Oracle App Server OS	Sun Solaris 10
Oracle SOA Suite DB OS	Sun Solaris 10

Table 1 - System Versions

CONFIGURATION

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:

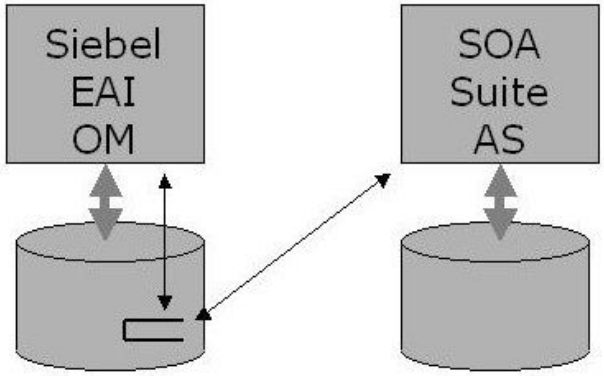

Figure 2 - Deployment Architecture

Create JMSUSER User

Separating the JMSUSER from other operations on the database is best practice and allows tighter control of the overall integration.

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

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.

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 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)

The JVM that is installed on the Siebel Application Server executes the calls to the JMS provider.

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:

```
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 allows the Siebel JMS Transport to locate the target queue for its operations.

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 inluded in the CLASSPATH, discussed later in this document.

An example jndi.properties file is given below:

```
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:

Profile	JAVA
Alias	JAVA
Subsystem Type	JVMSubSys
Description	Subsystem that includes the path to jvm dll file, Classpath & JVM Options.

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\jre\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

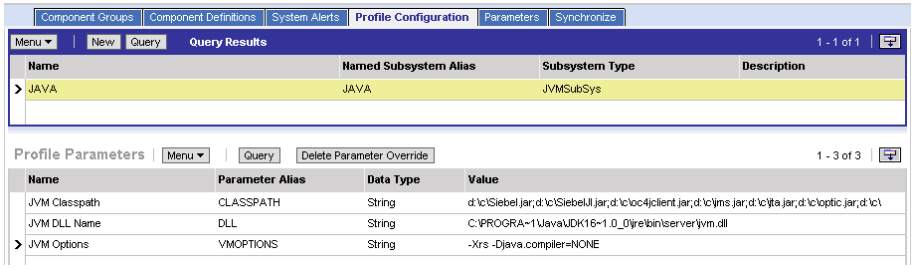

Figure 3 - Creating JAVA – JMSSubSys on Siebel Client

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:\c\Siebel.jar;d:\c\SiebelJI.j
ar;d:\c\oc4jclient.jar;d:\c\jms.jar;d:\c\j
ta.jar;d:\c\optic.jar;d:\c\", VMOPTIONS
="-Xrs -Djava.compiler=NONE"
```

Verify the values of the 3 Profile Properties:

Note: The classpath value is restricted to 100 characters on the Siebel Server Admin UI. Use the srvrmgr utility to run the "change param" command to update the CLASSPATH. Additionally there is a "." at the end of the classpath. This must be present; it explicitly refers to the "current" directory.

Name	JVM Classpath
Alias	CLASSPATH
Data Type	String
Value	Path to jar files seperated by a ; followed by the path to the jndi.properties file followed by a ; and . c:\c\jms.jar; c:\c\jta.jar; c:\c\oc4jclient.jar; c:\c\optic.jar; c:\c\javaee.jar; c:\c\jmxri.jar; c:\c\dms.jar; c:\c\j2ee_1.3.01.jar; c:\c\Siebel.jar; c:\c\SiebelJI.jar; c:\c\jndi.properties;.
Description	JVM Classpath

Name	JVM DLL Name
Alias	DLL
Data Type	String
Value	C:\PROGRA~1\Java\JDK16~1.0_0\jre\bin\server\jvm.dll
Description	JVM DLL Name

Name	JVM Options
Alias	VMOPTIONS
Data Type	String
Value	-Xrs -Xusealtsigs -Djava.compiler=NONE -Djms.log=siebjvm.log
Description	JVM Options

Configure JMS Transport and Receiver

The Receiver component is configured in order to allow the JMS Transport to receive messages.

Using the Siebel Server Manager (command-line)

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

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

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

```
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 screenshot below shows what the subsystem should look like after the configuration

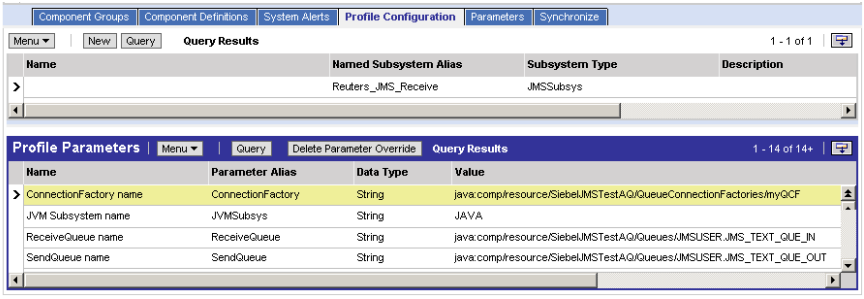

Figure 4 - Customer_JMS_Receive subsystem

To create a JMS Receiver subsystem using the Siebel Web Client a user should follow the steps below:

Each queue that sends messages to the Siebel application will have a different receiver component configured for it..

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-AQ_Receive_Message

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

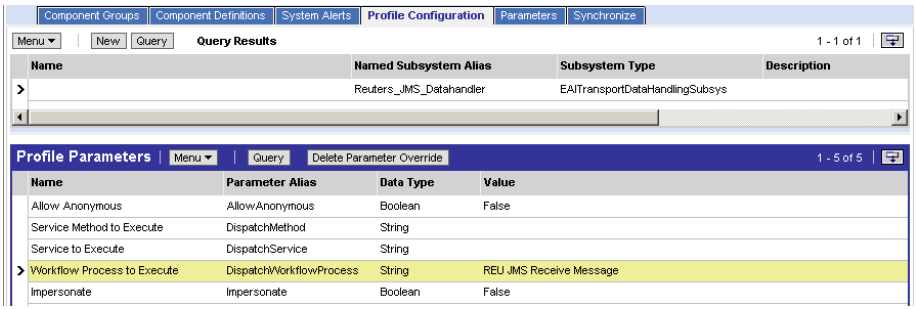

Figure 5 - Customer_JMS_Datahandler subsystem

Parameters

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

Parameter	Notes
ConnectionFactory	Connection Factory required by the JMS transport. This information is supplied by the messaging team
ReceiveQueue	The fully-qualified name of the queue that Siebel will receive messages from
SendQueue	The fully-qualified name of the queue that Siebel will Send messages to
ReceiveTimeout	Receive Timeout (in milliseconds)

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

Parameter	Notes
DispatchWorkflowProcess	Workflow Process to Execute.

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).

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.

1. Define the workflow as shown in the following figure

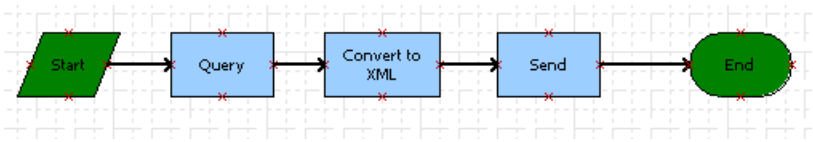

Figure 6 - The JMS send workflow

2. Create the following process properties in the Process Properties applet:

Name	Data Type	In/ Out	Default String	Comments
OrderXML	String	In		
JMSConnectionFactory	String	In	java:comp/resource/SiebelJMSTestAQ/QueueConnectionFactories/myQCF	JNDI name of the JMS connection factory
JMSSendQueue	String	In	java:comp/resource/SiebelJMSTestAQ/Queues/JMSUSER.JMS_TEXT_QUE_OUT	JNDI name of the queue
Order Message	IO	In		

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:

Input Argument	Type	Value	Property Name
Primary Row Id	Process Property		Object Id

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

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:

Input Argument	Type	Value	Property Name
SiebelMessage	Process Property		Order Message
GenerateProcessingInstructions	Literal	False	

Property Name	Type	Value	Output Argument
OrderXML	Output Argument		<Value>

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:

Input Argument	Type	Value	Property Name
<Value>	Process Property		OrderXML
ConnectionFactory	Process Property		JMSConnectionFactory
SendQueue	Process Property		JMSSendQueue

Property Name	Type	Value	Output Argument
OrderXML	Output Argument		<Value>

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

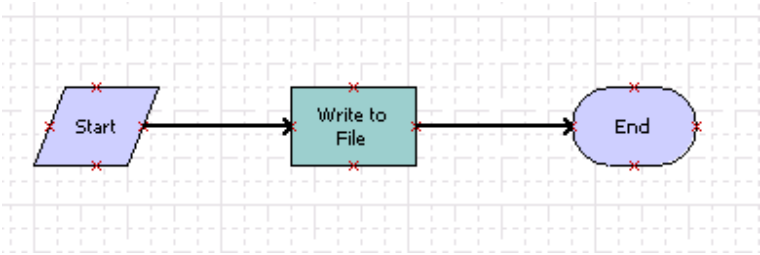

Figure 7 - Workflow executed by the Data Dispatcher component

3. Create the following process properties in the Process Properties applet:

Name	Data Type	In/ Out	Default String	Comments
<Value>	Binary	In	<Value>	Will contain the received message

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

Input Argument	Type	Value	Property Name
<Value>	Process Property		<Value>
FileName	Expression	" JMS_Incoming_" + [&Process Instance Id] + ".xml"	

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:

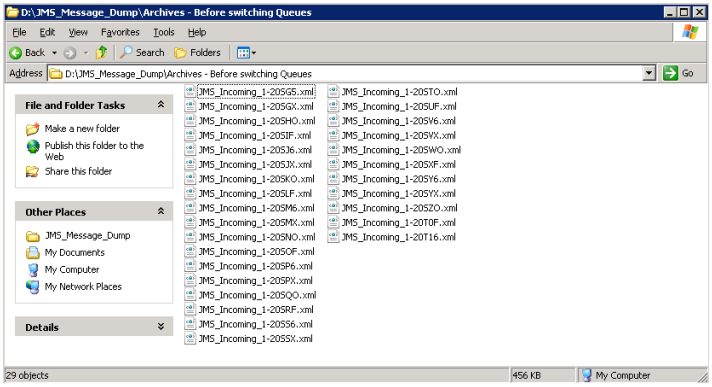

Figure 8 - XML messages in the destination Directory

Parameters

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

Parameter	Notes
ReceiverConnectionSubsystem	The name of the JMS Transport subsystem configured earlier
ReceiverDataHandlingSubsystem	The name of the JMS Receiver Subsystem configured earlier

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

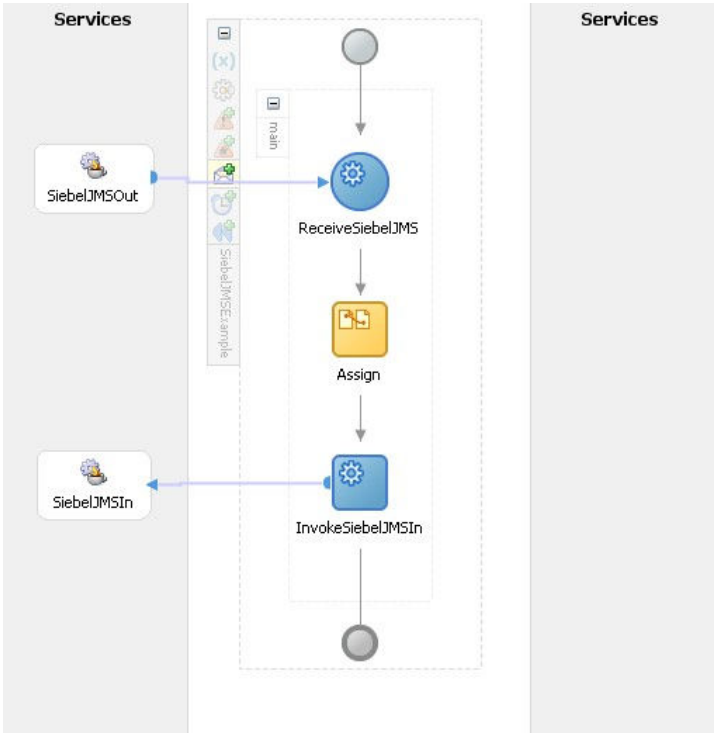

Figure 9 - BPEL test Stub

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, 5 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

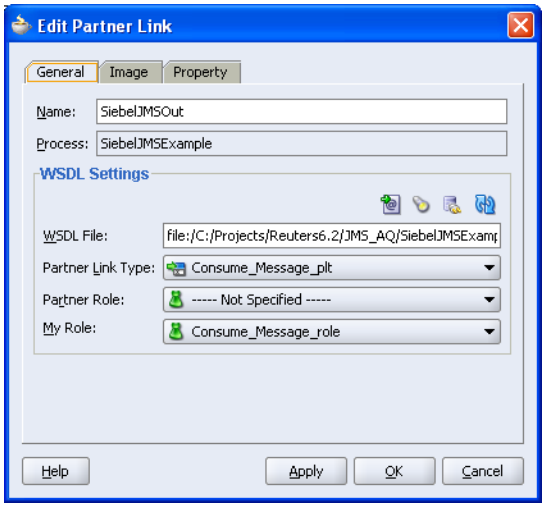

Figure 10 - SiebelJMSOut Queue

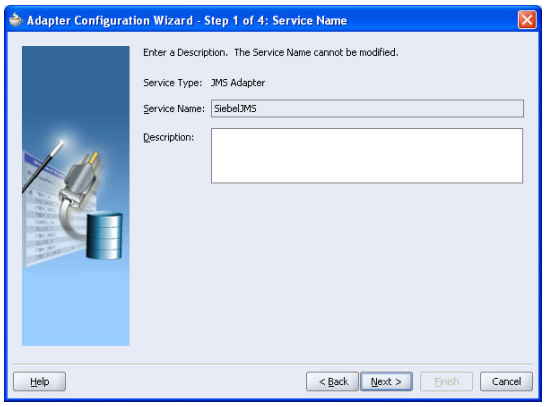

Figure 11 - SiebelJMSOut Queue

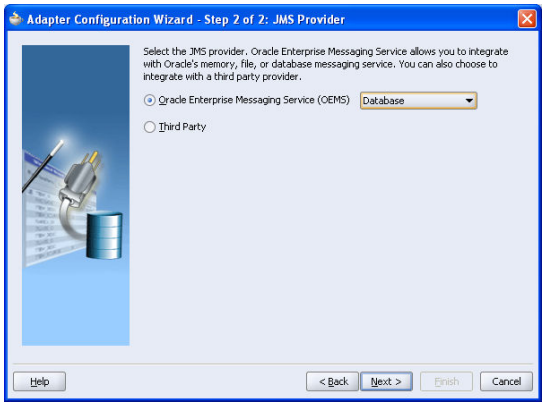

Figure 12 - SiebelJMSOut Queue

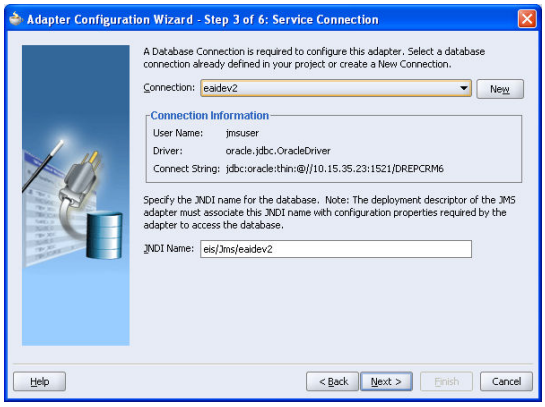

Figure 13 - SiebelJMSOut Queue

Figure 14 - SiebelJMSOut Queue

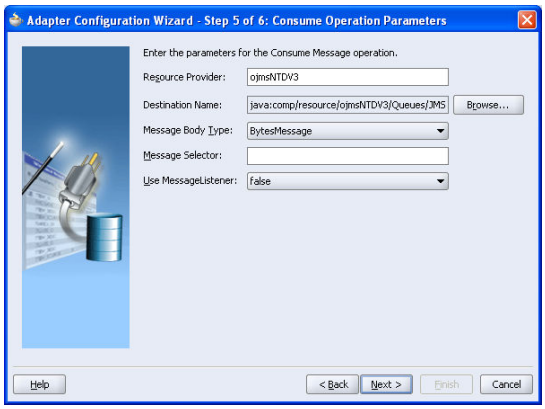

Figure 15 - SiebelJMSOut Queue

Configure the "Inbound" Queue

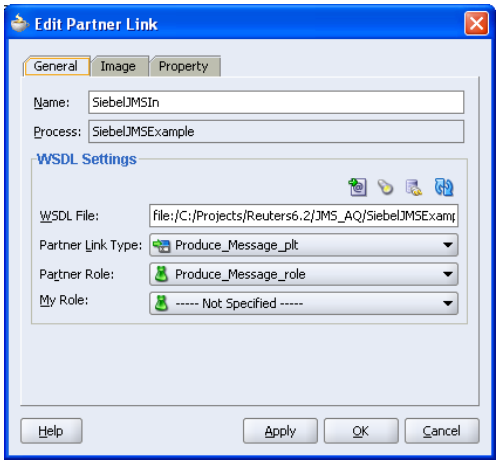

Figure 16 - SiebelJMSIn Queue JMS Adapter

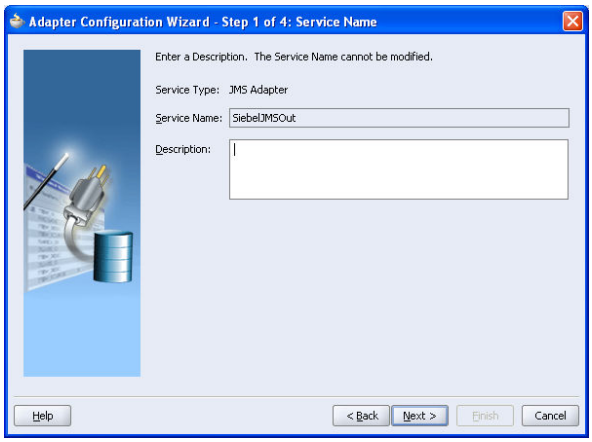

Figure 17 - SiebelJMSIn Queue JMS Adapter

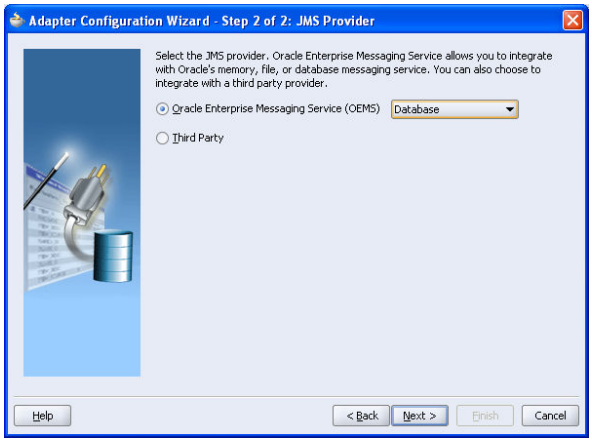

Figure 18 - SiebelJMSIn Queue JMS Adapter

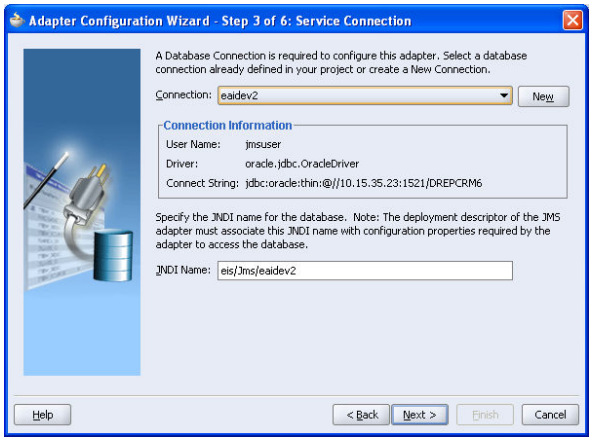

Figure 19 - SiebelJMSIn Queue JMS Adapter

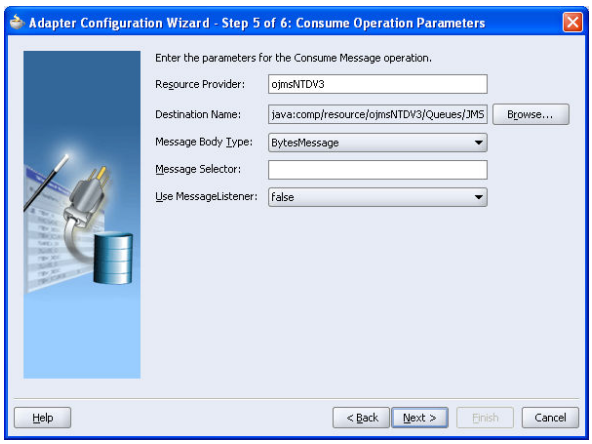

Figure 20 - SiebelJMSIn Queue JMS Adapter

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

```
<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>
```

In the oc4j-ra.xml file (in $SOA_HOME/ j2ee/<OC4J container>/application-deployments/default/JmsAdapter/), add the following code segments:

```
<connector-factory location="eis/jms/<QueueConnectionName>" connector-name="Jms
Adapter">
    <config-property name="connectionFactoryLocation" value="
         java:comp/resource/<ResourceProvider>/QueueConnectionFactories/myQCF" />
    <config-property name="factoryProperties" value="" />
    <config-property name="acknowledgeMode" value="AUTO ACKNOWLEDGE" />
    <config-property name="isTopic" value="false" />
    <config-property name="isTransacted" value="true" />
    <config-property name="username" value="jmsuser" />
    <config-property name="password" value="jmsuser" />
</connector-factory>
```

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.

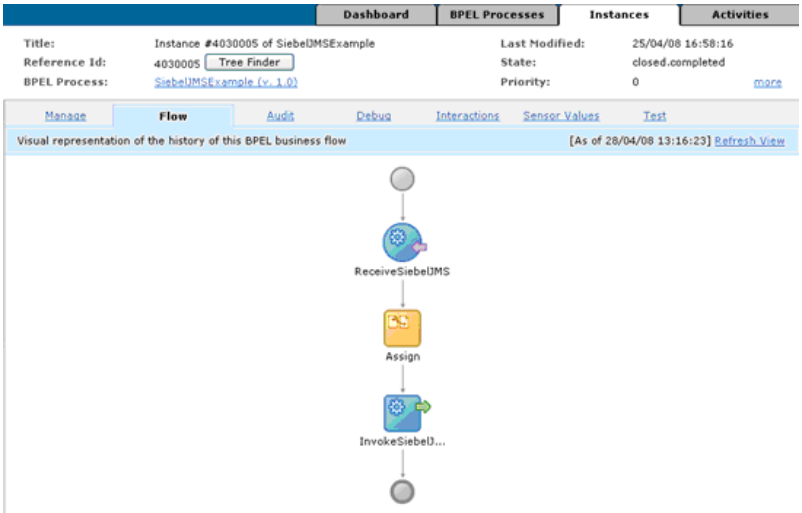

Figure 21 - The BPEL stub instance

Note: It is essential to ensure that all the database character sets are set the same otherwise you may encounter several endoding 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.

Configuring JMS/AQ messaging between Oracle Siebel CRM and Oracle SOA Suite
May 2008

Author: Nitin Manoharan
Contributing Authors: Graham Nicol
Mark McMellon
Cord Oestmann
Mayuresh Kadu

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com