



An Oracle White Paper  
June 2011

# ActiveMQ - Oracle Enterprise Gateway Integration Guide

## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

---

1. Introduction .....	4
1.1 Overview .....	4
1.2 JMS Architecture.....	4
1.3 Prerequisites .....	4
2. Setting Up The ActiveMQ Environment .....	5
2.1 Download ActiveMQ .....	5
2.2 Installation of ActiveMQ .....	5
2.3 Start ActiveMQ.....	5
3. Setting Up The OEG Environment.....	6
3.1 Load ActiveMQ Provider Files Onto The OEG Gateway .....	6
3.1.2 Instructions for Software install:.....	6
4. Configuring The Gateway To Place Messages On ActiveMQ Queue .....	6
4.1 Creating A JMS Session: .....	6
4.2 Create A Policy That Will Place Message On The ActiveMQ Queue. ....	7
4.2.1 Configuring the Messaging System Filter .....	8
4.3 Ensure Policies Are Updated On The Gateway .....	11
4.4 Test The Configuration To Place Message On ActiveMQ Queue .....	11
5. Configuring The Gateway To Read From An ActiveMQ Queue. ....	14
5.1 Create Policy That Will Be Invoked When A Message Is Read From ActiveMQ .....	14
5.2 Creating A JMS Session .....	14
.....	15
5.3 Ensure Policies Are Updated On The Gateway .....	15
5.4 Testing to Read Messages From A Queue.....	16
6. Conclusion .....	18

# 1. Introduction

## 1.1 Overview

This document describes how to configure the Gateway to perform protocol translation. This will be demonstrated by the following:

1. The Gateway will listen for messages on a HTTP interface. Messages read from this interface will be placed on a message queue.
2. The Gateway will listen for messages on a message queue. Messages read from the queue will be logged to a file.

This guide applies to OEG software products, from version 6.x upwards. In this guide the message queuing system that will be used is **ActiveMQ**.

## 1.2 JMS Architecture

The Gateway utilises JMS (Java Message Service) for sending and receiving messages from messaging systems. JMS API which was developed by Sun defines a common set of interfaces and associated semantics that allow the Gateway to communicate with various messaging applications in a standard way.

Messaging system products (IBM WebSphere MQ, JBossMQ, SonicMQ, Fiorano, OpenJMS and ActiveMQ) provide implementations of JMS which can be plugged into the Gateway.

The Gateway has been designed to allow 3rd party JMS providers to be "plugged in" for easy integration. To plug in a new JMS provider, you must install the JMS provider on the machine where the Gateway software resides on. The messaging system vendors normally provide implementations of the JMS provider which is normally in the form of jar files. The configuration settings for the JMS integration are configured in the OEG Policy Studio.

## 1.3 Prerequisites

- ActiveMQ available from <http://activemq.apache.org>
- OEG's Gateway Software available from [www.oracle.com](http://www.oracle.com)
- Java Developer Kit (JDK) 1.5.x or greater. Java home should be set to this directory.

## 2. Setting Up The ActiveMQ Environment

### 2.1 Download ActiveMQ

ActiveMQ is available from: <http://activemq.apache.org>

The most current version (at time of writing) is ActiveMQ v5.3.2

### 2.2 Installation of ActiveMQ

#### **For Windows:**

Extract the ActiveMQ files to a desired directory

#### **For Linux:**

Extract the ActiveMQ files to a desired directory

**NOTE:** For more advanced configuration options please see the ActiveMQ documentation available from <http://activemq.apache.org>

### 2.3 Start ActiveMQ

Start the ActiveMQ by completing the following steps:

#### **For Windows:**

1. JAVA\_HOME environment variable must be set to the directory where the JDK is installed

*Example:*

```
> set JAVA_HOME=c:\java\jdk1.5.0_22
```

2. From the command prompt, browse to the ActiveMQ\_root\bin directory.
3. Start ActiveMQ by typing **activemq** at the prompt.

*Example:*

```
Activemq\bin>activemq
```

4. Check that ActiveMQ started successfully by running the following command from a Windows command prompt:  
**netstat -an|find "61616"** (port 61616 is the default ActiveMQ listening port)

#### **For Unix:**

JAVA\_HOME environment variable must be set to the directory where the JDK is installed

*Example:*

```
> export JAVA_HOME=/opt/java/jdk1.5.0_22
```

1. From the terminal prompt, browse to the ActiveMQ\_root/bin directory.

2. Start ActiveMQ by typing **activemq** at the prompt.

*Example:*

3. Activemq/bin> ./activemq
4. Check that ActiveMQ started successfully by running the following command from a Unix command shell  
netstat -an|grep 61616 (port 61616 is the default Act

### 3. Setting Up The OEG Environment

#### 3.1 Load ActiveMQ Provider Files Onto The OEG Gateway

ActiveMQ provides a particular JMS provider that the Gateway will use to connect. The JMS provider takes the form of Java archive files (i.e. JAR files). Once ActiveMQ is installed it is a simple matter to drop the JMS provider JAR files onto the OEG Gateway.

#### 3.1.2 Instructions for Software install:

Copy the following jar file from the /activemq\_install\_dir/lib directory to the OEG\_product\_dir/ext/lib directory

- activemq-all-5.3.0.jar

### 4. Configuring The Gateway To Place Messages On ActiveMQ Queue

The gateway will be configured to place messages it receives on a queue (i.e. destination) named 'example.A' (default ActiveMQ queue) in ActiveMQ.

#### 4.1 Creating A JMS Session:

1. Start the **Gateway** and **Policy Studio** (for more details refer to Getting Started in the Help Configuration Guide.)
2. Click on the **External Connections** navigation panel in Policy Studio
3. Right click on **JMS Services** and click on **Add a JMS Service**
4. Configure the following fields for the **JMS Service**:  
**Name:** ActiveMQ  
**Provider URL:** tcp://ip\_of\_activemq\_host:61616  
**Initial ContextFactory:** org.apache.activemq.jndi.ActiveMQInitialContextFactory  
**Connection Factory:** QueueConnectionFactory

**Username:** can be left blank by default

**Password:** can be left blank by default

5. Click **Add** at the bottom of the **Custom Message Properties** section and enter:

**Name:** queue.A

**Value:** example.A

6. Click on **OK**.

7. Click on **OK**.

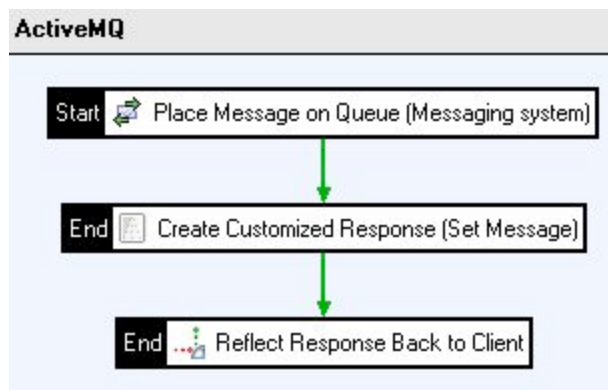
**NOTE:** ActiveMQ is the registering queues in JNDI using the form:  
queue.[jndiName] = [physicalName]

For more information please see ActiveMQ documentation at <http://activemq.apache.org>

#### 4.2 Create A Policy That Will Place Message On The ActiveMQ Queue.

##### The policy:

1. The policy will consist of a **Messaging System** filter that is responsible for communicating with a JMS system and placing messages on the listening queues.
2. Once the message has successfully been processed by the Messaging System filter it will pass through a **Set Message** filter that will create a customized response to send back to the client. This is done for demonstration purposes; the policy can be customized to preference with the exception that a Messaging System filter needs to be present.
3. The last filter in the policy is a Reflect filter that will send the customized response back to the client.



### Create a test policy to route messages on to the ActiveMQ Queue:

- Open **Policy Studio** and select **Policies** navigation panel. Create a new Policy titled **Route to ActiveMQ** by right clicking on **Policies** and select **Add Policy**.
- Create a new relative path on the Gateway Process called **/ToActiveMQ** by selecting **Services** navigation panel, then expand **OEG Gateway** and right click on **Default Services** to **Add Relative Path**.
- Map the **/ToActiveMQ** path to the policy called **Route to ActiveMQ** with a tick. This means that when a message is received by the Gateway on the path **/ToActiveMQ**, it will be passed to the **Route to ActiveMQ** policy, which will then process the message.

#### 4.2.1 Configuring the Messaging System Filter

When a policy that routes to a JMS provider (such as ActiveMQ) is created, the policy must contain a **Messaging System** filter, which can be found under the **Routing** filter category in Policy Studio. To configure this filter, complete the following steps:

1. Go back to the “Route to ActiveMQ” policy under Policies. (the TAB should still be open)
2. Drag a **Messaging System** from the **Routing** filter category onto the policy canvas.
3. Under the **Request** tab select the ActiveMQ JMS Service that has been configured above (titled **ActiveMQ**) from the JMS Session dropdown.
4. Set the Destination to **A**, which was configured during the ActiveMQ destination configuration i.e example.**A**
5. Specify the **Message Type**. For example, in **Message Type**, select **Use content.body attribute to create a message adhering to the “SOAP over JAVA Message Service proposal”**.
6. All other settings may be left at default.
7. Click on the **Response tab** and select **No Response** and click on **Finish**.
8. Once configured right click on the filter and select **Set as Start**.

The Messaging System filter configured as described above



**Configure "Place Message on Queue (Messaging system)"**

**Send message to messaging system**

Send message to messaging system.

Name: Place Message on Queue (Messaging system)

Request | Response

JMS Service: ActiveMQ

Destination: A

Delivery Mode: Persistent

Priority Level: 4

Time to Live: 0 (ms)

Message ID: \$\${id}

Correlation ID:

Message Type: Use content.body attribute to create a message adhering to the "SOAP over Java Message Service" proposal

Attribute Name:

Use shared JMS session

Custom Message Properties:

Name	Value

Add Edit Delete

Help < Back Next > Finish Cancel

### Note on Message Type:

Explanation of the various serializations (from OEG message to JMS message)

**- Use content.body attribute to create a message in the format specified in the "SOAP over Java Messaging Service" proposal:**

If this option is selected, messages will be formatted according to the SOAP over JMS proposal. This is the default option since, in most cases; it is the message body that is to be routed to the messaging system. This will result in a ByteMessage being sent to the queue/topic and JMS a property will contain the Content-Type (i.e. text/xml)

**- Create a MapMessage from the java.lang.Map in the attribute named below:**

Select this option to create a javax.jms.MapMessage from the OEG message attribute named below that consists of name-value pairs.

**- Create a ByteMessage from the attribute named below:**

Select this option to create a javax.jms.ByteMessage from the OEG message attribute named below.

**- Create an ObjectMessage from the java.lang.Serializable in the attribute named below:**

This option can be selected in order to create a javax.jms.ObjectMessage from the OEG message attribute named below.

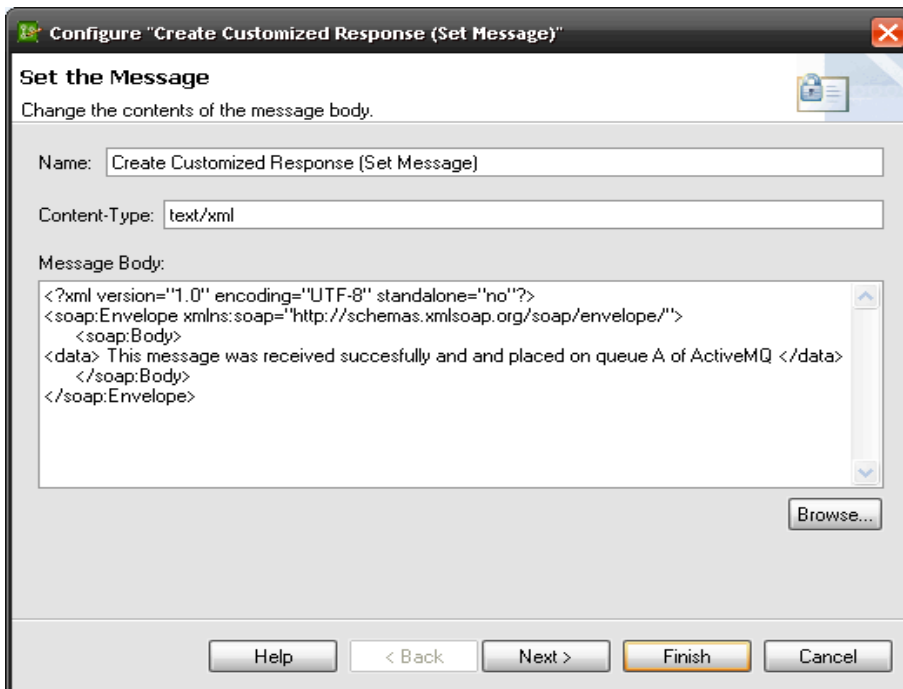
**- Create a TextMessage from the attribute named below:**

A javax.jms.TextMessage can be created from the message attribute named below by selecting this option from the dropdown.

---

**To complete the test policy create the following flow:**

1. **Messaging System Filter:** This filter should be configured as described above. This is a mandatory filter in the policy.
2. **Set Message Filter:** Used to set the content of an XML response message that can be returned to the client to acknowledge that the message has been placed on the ActiveMQ queue. This step is not mandatory, but is useful for testing purposes. This filter can be found in **the Conversion** filter category. Add a custom XML message in the **Message Body** section and for **Content-Type** enter **text/xml**. Connected from the **Messaging System** filter via a success path.
3. **Reflect Filter:** Used to reflect the response back to the client. The filter can be found in the Utility filter category. Can be left default. Connected from the **Set Message** filter via a success path.



#### 4.3 Ensure Policies Are Updated On The Gateway

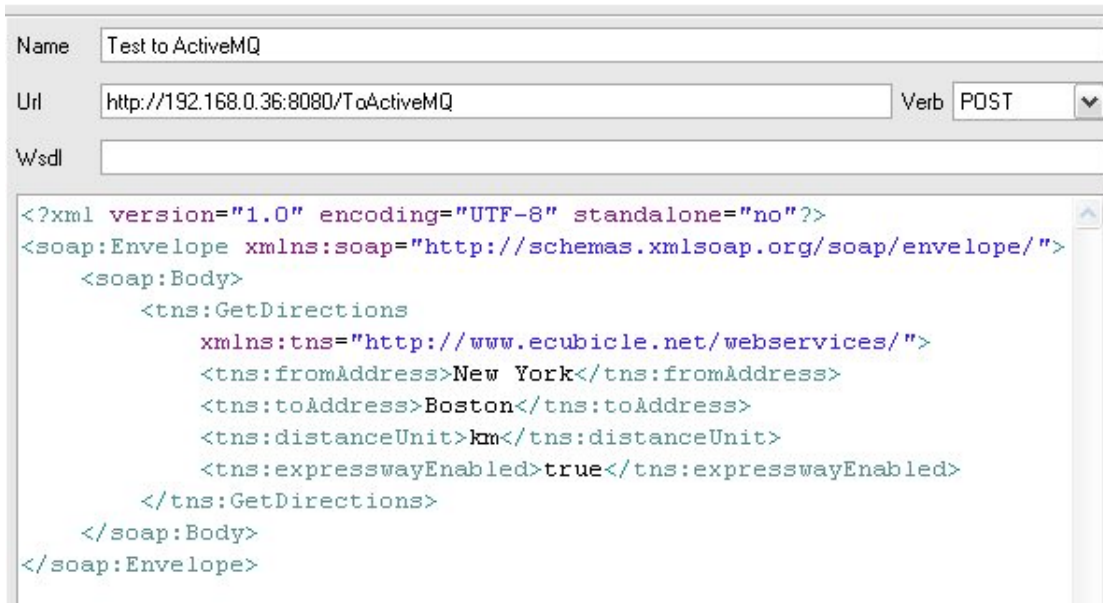
1. Open the **Policy Studio**.
2. Click on **Settings**.
3. Select **Deploy** to ensure that the changes made are propagated to the running Gateway
4. Click **Yes** to Deploy.

#### 4.4 Test The Configuration To Place Message On ActiveMQ Queue

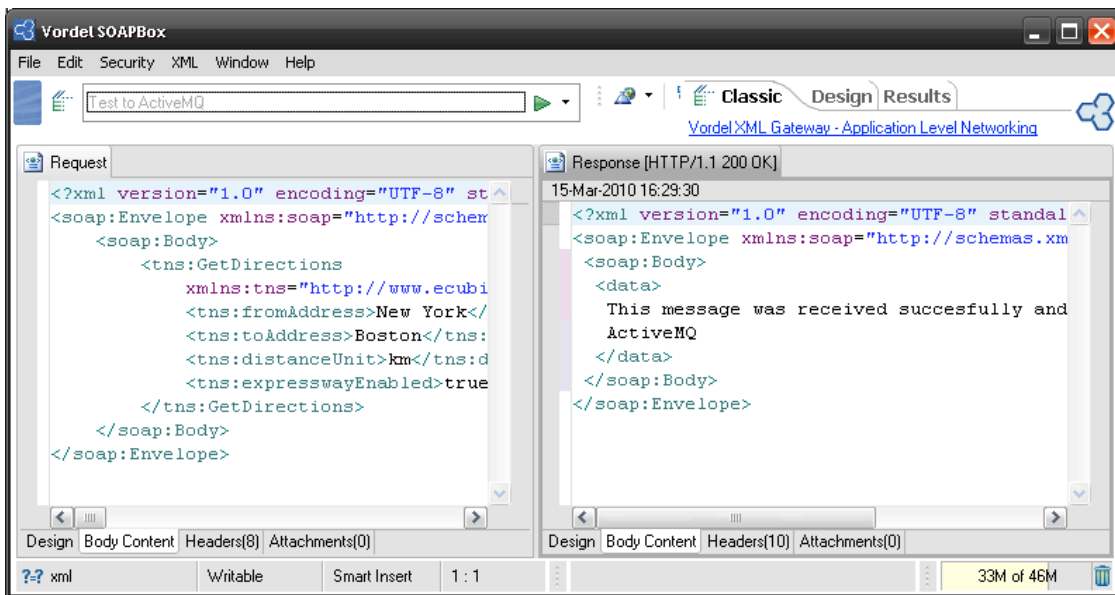
OEG Service Explorer will be used as the client to test the integration. OEG's Service Explorer test client is available to download for free from [www.oracle.com](http://www.oracle.com)

1. Open **Service Explorer** and load a request in the **Request** window.
2. Ensure that the URL field in SOAPbox points to the Gateway and the **/ToActiveMQ** path on the Gateway. This is set in the Request Settings window accessed from small black triangle to the right of the green triangle button.

*Example:* http://ip\_of\_xmlgateway:8080/ToActiveMQ



3. Click on the **Send** button and the response will appear in the right hand pane:









4. The messages can be viewed by using the ActiveMQ web console which is can be accessed by default on [http://ip\\_of\\_activemq:8161/admin/](http://ip_of_activemq:8161/admin/)
5. Click on the **Queues** tab to see how many messages are on the queues.

Home | Queues | Topics | Subscribers | Connections | Send

Queue Name

### Queues

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
ActiveMQ.DLQ	1	0	0	0	Browse Active Consumers  	Send To Purge Delete
example.A	0	2	4	4	Browse Active Consumers  	Send To Purge Delete
example.B	3	0	3	0	Browse Active Consumers  	Send To Purge Delete

6. It can be seen that 4 requests has been sent to queue example. A from the Gateway.

Here is a truncated extract of the transaction in the Gateway trace file:

```

-----
DEBUG 12:08:50:198 [0dcc] Incoming HTTP request: method=POST, host=(unset),
port=(unset), path=/ToActiveMQ, query=(unset), version=1.1
DEBUG 12:08:50:198 [0dcc] attaching current thread to VM
DEBUG 12:08:50:198 [0dcc] done: env=0705B4F4
DEBUG 12:08:50:198 [0dcc] handle type text/xml with factory class
com.vordel.mime.XMLBody$Factory
DEBUG 12:08:50:198 [0dcc] run circuit "ActiveMQ"
DEBUG 12:08:50:198 [0dcc] run filter [Place Message on Queue (Messaging system)] {
DEBUG 12:08:50:198 [0dcc]   Fire and forget about response
DEBUG 12:08:50:198 [0dcc]   JMSProcessor message ID is: Id-0000012766e05516-
0000000001123eb0-1
DEBUG 12:08:50:198 [0dcc]   Creating JMS byte message using automime
DEBUG 12:08:50:198 [089c] start thread 06A3E280 in set "netshvc threadpool":
count=7, busy=6, idletarget=4, max=1024
DEBUG 12:08:50:213 [0dcc] } = 1, in 15 milliseconds
DEBUG 12:08:50:213 [0dcc] run filter [Create Customized Response (Set Message)] {
DEBUG 12:08:50:213 [0dcc]   ConversionProcessor.setConvertedMessage:The
contentype of the converted message is text/xml
DEBUG 12:08:50:213 [0dcc]   ConversionProcessor.setConvertedMessage: coverted
message is added to the the pipeline
DEBUG 12:08:50:213 [0dcc]   ChangeMessageProcessor.convert: finished
DEBUG 12:08:50:213 [0dcc] } = 1, in 0 milliseconds
DEBUG 12:08:50:213 [0dcc] run filter [Reflect Response Back to Client] {

```

DEBUG 12:08:50:213 [0dcc] } = 1, in 0 milliseconds

DEBUG 12:08:50:213 [0dcc] ActiveMQ

DEBUG 12:08:50:213 [0dcc] add header Server:Vordel

---

## 5. Configuring The Gateway To Read From An ActiveMQ Queue.

The Gateway will be configured to read the messages from a queue from ActiveMQ.

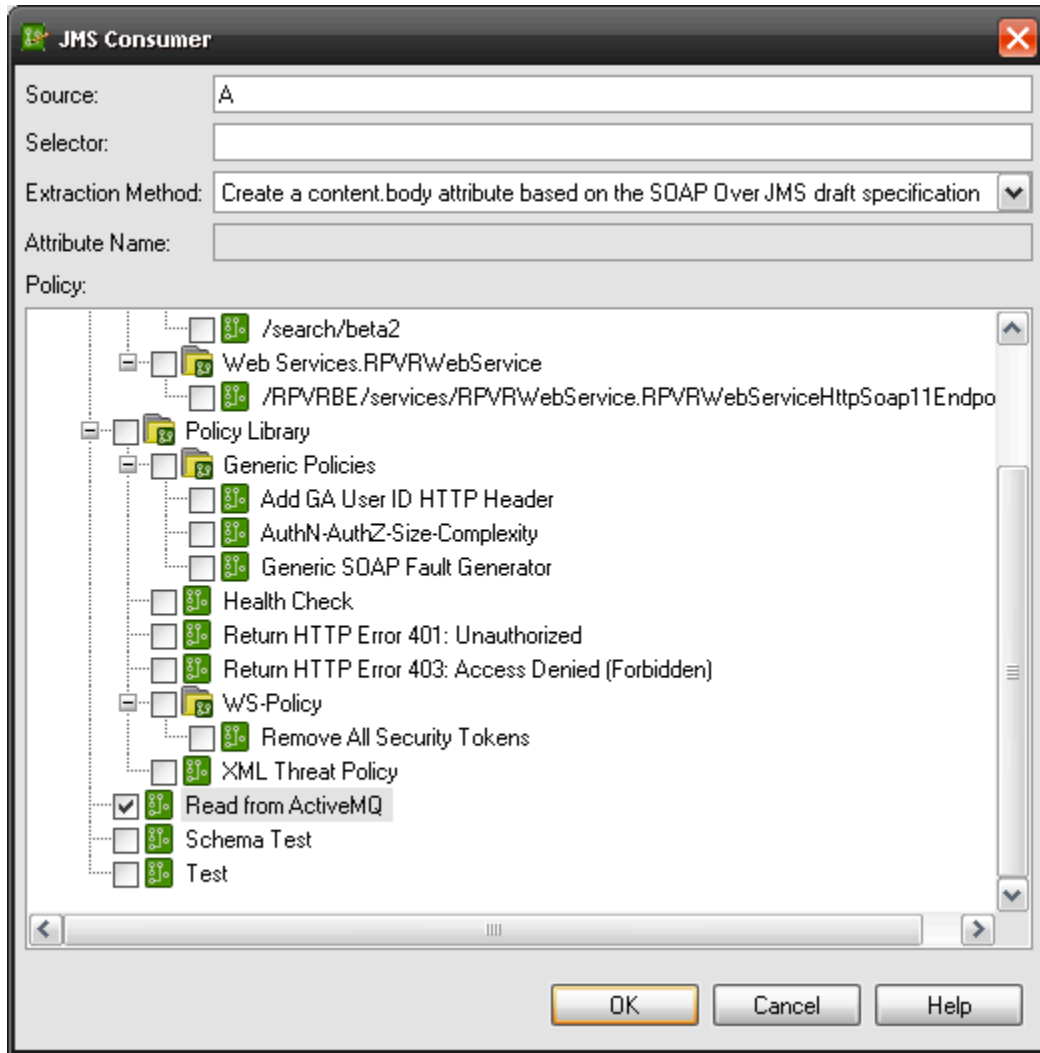
### 5.1 Create Policy That Will Be Invoked When A Message Is Read From ActiveMQ

To create the second policy that the JMS Consumer (Part of JMS Session in Gateway) will point to:

1. Select the **Policies** navigation panel and right click on **Policies**
2. Click **Add Policy** and create a new policy titled **Read from ActiveMQ**.
3. Drag a **Log Message Payload** filter onto the policy canvas from **the Monitoring** filter category. The filter configuration can be left default.
4. Click on **Finish** then right click on the filter and select **Set as Start**.
5. Click back to the **Services** module enable logging from the Services module by right clicking on the **OEG Gateway Process** and select **Logging->Custom**.
6. Select the **Text File** tab and select the **Enable Logging to file** check box.
7. Click **OK**

### 5.2 Creating A JMS Session

1. Open **Policy Studio**, select the **Services** navigation panel then right click on the **OEG Gateway** process and select **Messaging System->Add JMS Session**.
2. In the popup window by the **JMS Service** drop down select the ActiveMQ service that was created earlier.
3. Do not select **Allow Duplicates**. Click on **OK**.
4. Right click on **JMS Session** and select **Add JMS Consumer**.
5. Configure as follows:
  - Source:** A
  - Selector:** Can be left blank by default.
  - Extraction Method:** Create a content.body attribute based on the SOAP Over JMS Draft specification.
6. Point the JMS Consumer to the **Read from ActiveMQ** policy.



### 5.3 Ensure Policies Are Updated On The Gateway

Complete the following steps to refresh the policies:

1. Open the **Policy Studio** and click on **Settings**.
2. Select **Deploy** and **Yes** to the popup window to ensure that the changes made are propagated to the running Gateway.

### 5.4 Testing to Read Messages From A Queue

The Gateway can be configured to let the JMS service consume a message on the queue and to forward it to a log file

By creating the JMS consumer and the policy that it points to (i.e. Read from ActiveMQ Queue) that contains a Log Message Payload filter, the messages have been read from **example.A queue** and the message logged to a file called vordel.log that is located in the oeg\_gateway\_root/logs directory.

Once the Gateway has been configured to consume messages on an ActiveMQ queue, the log files and trace can be checked to see the transactions.

Here is a truncated extract of the transaction in the Gateway trace file after the message has been consumed from the ActiveMQ queue:

```
-----  
DEBUG 12:20:38:546 [040c] JMSConsumer.onMessage from destination:  
queue://example.A  
DEBUG 12:20:38:562 [040c] New message read from JMS queue, Vordel message ID is:  
Id-0000012766eb2412-000000000031ff23-1  
DEBUG 12:20:38:609 [040c] JMS header: name=contentType,  
value=text/xml;charset="utf-8"  
DEBUG 12:20:38:609 [040c] JMS Message ID: ID:schoemang-4842-1268741233963-  
0:0:11:1:1  
DEBUG 12:20:38:625 [040c] JMS Correlation ID: Unknown  
JMS Correlation ID: Tue Mar 16 12:20:38 GMT 2010  
DEBUG 12:20:38:625 [040c] JMS Type: Unknown JMS Type: Tue Mar 16 12:20:38  
GMT2010  
DEBUG 12:20:38:625 [040c] JMS Destination: queue://example.A  
DEBUG 12:20:38:640 [040c] Extracting message with automime  
DEBUG 12:20:38:640 [040c] handle type text/xml with factory class  
com.vordel.mime.XMLBody$Factory  
DEBUG 12:20:38:640 [040c] relay from a class com.vordel.dwe.jms.BytesMessageIn  
putStream to a class com.vordel.dwe.BufferingContentSource$FeedStream  
DEBUG 12:20:38:656 [040c] run circuit "Read from ActiveMQ"  
DEBUG 12:20:38:656 [040c] run filter [Log Message Payload] {  
DEBUG 12:20:39:750 [040c] relay from a class com.vordel.dwe.NativeContentS  
ource$IS to a class java.io.ByteArrayOutputStream  
DEBUG 12:20:39:750 [040c] close content stream  
DEBUG 12:20:39:750 [040c] } = 1, in 1094 milliseconds  
DEBUG 12:20:39:750 [040c] Read from ActiveMQ
```

---



Contents of the vordel.log log file:

```
-----  
03.16.2010 12:07:31,218 Id-0000012766df1fe6-  
000000000083d37a-1 Log Message Payload  
contentType: text/xml;charset="utf-8"  
jms.header.JMSCorrelationID: Unknown JMS Correlation ID: Tue Mar 16 12:07:31 GMT  
2010  
jms.header.JMSDeliveryMode: 2  
jms.header.JMSMessageID: ID:schoemang-4842-1268741233963-0:0:4:1:1  
jms.header.JMSPriority: 4  
jms.header.JMSTimestamp: 1268741254823  
jms.header.JMSType: Unknown JMS Type: Tue Mar 16 12:07:31 GMT 2010  
contentType: text/xml;charset="utf-8"  
jms.header.JMSCorrelationID: Unknown JMS Correlation ID: Tue Mar 16 12:07:31 GMT  
2010  
jms.header.JMSDeliveryMode: 2  
jms.header.JMSMessageID: ID:schoemang-4842-1268741233963-0:0:4:1:1  
jms.header.JMSPriority: 4  
jms.header.JMSTimestamp: 1268741254823  
jms.header.JMSType: Unknown JMS Type: Tue Mar 16 12:07:31 GMT 2010  
<?xml version="1.0" encoding="utf-8" standalone="no"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
<soap:Body>  
<tns:GetDirections xmlns:tns="http://www.ecubicle.net/webservices/">  
<tns:fromAddress>New York</tns:fromAddress>  
<tns:toAddress>Boston</tns:toAddress>  
<tns:distanceUnit>km</tns:distanceUnit>  
<tns:expresswayEnabled>true</tns:expresswayEnabled>  
</tns:GetDirections>  
</soap:Body>  
</soap:Envelope>  
-----
```

## 6. Conclusion

This document is a simplistic demonstration on how to setup the connection from a OEG Gateway to the ActiveMQ provider using the JMS Service and filter options in the Gateway.

This configuration can be part of a larger policy, including features such as XML threat detection and conditional routing, features which are out of the scope of this document but are covered in other documents which can be obtained from Oracle at <http://www.oracle.com>.



Oracle Enterprise Gateway  
May 2011  
Author:

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



| Oracle is committed to developing practices and products that help protect the environment

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 0410

**SOFTWARE. HARDWARE. COMPLETE.**