Web services with SOAP over JMS in IBM WebSphere Process Server or IBM WebSphere Enterprise Service Bus v7.0.0.x using IBM WebSphere MQ JMS provider

Simon Holdsworth
Bo Wang

August 24, 2011

IBM® WebSphere® Process Server and IBM WebSphere Enterprise Service Bus products both include support for invoking and providing Web services using SOAP over Java™ Message Service (JMS) protocol. We can learn how to change the referenced Java Naming and Directory Interface (JNDI) resources of SOAP over JMS exports and imports so that they can receive and send SOAP messages using the IBM WebSphere MQ JMS provider rather than the SIBus JMS provider via an article covering WebSphere Process Server and WebSphere Enterprise Service Bus V6.02 and V6.1. However, there are changes for WebSphere Process Server and WebSphere Enterprise Service Bus V7.0, which are described in this article. To follow along with this article, you should know how to create modules in IBM WebSphere Integration Developer 7.0 and how to create WebSphere MQ resources.

Introduction

In WebSphere Integration Developer, when import or export bindings are generated for a module with the SOAP over JMS protocol selected, it is assumed that the embedded (SIBus) JMS provider is to be used and its resources are always generated. The WebSphere Process Server and WebSphere ESB runtime implementation uses the JMS API, so it is possible for an administrator to enable the use of other JMS providers, in particular the WebSphere MQ JMS provider, to transport SOAP messages over MQ JMS by replacing these generated JMS resources.

Find out how to configure imports and exports in WebSphere Process Server and WebSphere Enterprise Service Bus with Web service SOAP over JMS bindings to use WebSphere MQ JMS resources. This article provides example Service Component Architecture (SCA) applications that contain imports and exports with Web services-based SOAP over JMS bindings. It also describes how they are modified to use WebSphere MQ JMS resources. After reading this article, you can apply the steps described here to other such imports and exports.

The procedure is somewhat different between exports and imports. Exports make use of a message-driven bean (MDB) to receive requests, and they send responses to the destination specified by each request's JMSReplyTo header. You can modify the resources for an export
via the IBM WebSphere Application Server admin console. In WebSphere Process Server and WebSphere ESB V7.0, the MQ JMS provider is a Java 2 Platform, Enterprise Edition (J2EE) Connector architecture (JCA) resource adapter, which means that the MDB is configured by an activation specification object, rather than the listener port object that was used in previous releases.

Imports do not use an MDB to receive responses and can be configured to use a single reply queue or one temporary reply queue per request via the JMS URL specified for the import binding in the WebSphere Integration Developer tool. The relevant parts of the syntax of the JMS URL are described in this article.

To perform these configuration procedures, you need the following software:

- WebSphere MQ V7.0 with Refresh Pack 7.0.1.0
- WebSphere Integration Developer V7.0.0.x
- WebSphere Process Server V7.0.0.x or WebSphere Enterprise Service Bus V7.0.0.x

The application modules in this article have been developed and tested with versions 7.0.0.3 of WebSphere Process Server. These modules are examples to demonstrate how to create and configure SCA modules using SOAP over JMS in WebSphere Integration Developer and WebSphere Process Server or WebSphere Enterprise Service Bus. The downloadable project interchange files are provided in WebSphere Integration Developer V7.0.0.3.

Resources required for point-to-point messaging

This section describes a procedure for defining WebSphere MQ and WebSphere MQ JMS resources that can then be used by an import or export with a Web service SOAP over JMS binding. The names of the resources provided here are examples that you can change as required. First you find out how to work with the point-to-point messaging model using queue destinations. Later, in the Resources required for publish/subscribe messaging section, you learn how to work with the messaging model using topics. There are two ways to create a WebSphere MQ queue manager and queues:

- Create them from the WebSphere MQ Explorer.
- Create them from a command line window using WebSphere MQ commands.

For example, from a command line window, you enter `crtmqm QMGR` to create a queue manager and `strmqm QMGR` to start the queue manager. (See Related topics for a link to the WebSphere MQ V7.0 Information Center, which has details on how to create the required objects.)

This procedure requires the following WebSphere MQ queue manager and queues to be created:

- WebSphere MQ queue manager: QMGR
- WebSphere MQ queues: REQUESTQ and RESPONSEQ on QMGR for use with the export with Web service SOAP over JMS binding
- WebSphere MQ queues: REQUESTQ1 and RESPONSEQ1 on QMGR for use with the import with Web service SOAP over JMS binding
Corresponding to these WebSphere MQ resources, you need to create WebSphere MQ JMS Provider resources in the WebSphere Enterprise Service Bus or WebSphere Process Server admin console. You must create these resources under the WebSphere MQ JMS Provider at the node scope. Use the following steps to create the resources required for an export and import configuration:

1. Create a WebSphere MQ queue connection factory used for request connections for both export and import: `requestMQCF`, which uses QMGR.
2. Create a WebSphere MQ JMS queue connection factory used for reply connections for both export and import: `replyMQCF` using QMGR.
3. Create a WebSphere MQ JMS queue destination for receiving a request message at an export: `requestQ` using REQUESTQ.
4. Create a WebSphere MQ JMS queue destination for sending a request message from an import: `requestQ1` using REQUESTQ1.
5. Create an activation specification on the server: `MQAS`, which uses `requestQ`.
6. Create an activation specification on the server: `MQAS1`, which uses `requestQ1`.
7. Create a WebSphere MQ queue destination for sending a response message: `replyQ` using RESPONSEQ.

**Note:** This queue is used by a client application as the destination to which replies are to be sent, specified via the JMSReplyTo header in JMS messages. This destination is not configured against the web service SOAP over JMS export or import bindings.

### Application scenario for the point-to-point messaging model

**Figure 1. Example of scenario for point-to-point messaging model**

- **The WSMQJMSClient:** Web application that uses a JavaServer Pages (JSP) `sendMessage.jsp` to send SOAP messages.
- **WSMQJMSBackEndService:** Web service with SOAP over JMS using WebSphere MQ resources, whose implementation invokes another Web service.
- **WSMQJMSServiceProvider:** Another Web service provider using WebSphere MQ resources. Messages flow between these applications as follows:
  1. The client sends a SOAP message to a defined WebSphere MQ queue.
  2. The SOAP message is converted from the WebSphere MQ queue into a business object using an export with a Web service SOAP over JMS binding.
  3. A Java component handles the message and invokes another Web service provider via an import with a Web service SOAP over JMS binding.
  4. The Java component returns a response message to the defined WebSphere MQ queue via an export with Web service SOAP over JMS binding (when the request operation is a request/response operation).
Create an application with a SOAP over JMS export binding

This application is designed to have an export with a Web service SOAP over JMS binding and a Java component with the implementation of the interface of the Web service. This export receives SOAP messages sent by a JMS client application. The SOAP messages are then converted by the runtime into business objects for use by components within the application. This application is provided as an example, which is then modified to use WebSphere MQ JMS resources; you can apply the same procedure to your own exports with Web service SOAP over JMS bindings. Let’s break this down:

1. Create a library called MyLibrary, which contains a `CustomerService` interface with three functions, as shown in Listing 1.

Listing 1. CustomerService interface

```java
CustomerInfo getCustomerInfo(String id);
void createCustomer(String id);
int getQuantity(String id);
```

2. Create a module called `WSMQJMSServiceProvider`, which has an export with a Web service SOAP over JMS binding and a Java component from WebSphere Integration Developer. The `ProviderSOAPMQJMSExport` has the Web service SOAP over JMS binding as shown in Figure 2.

Figure 2. WSMQJMSServiceProvider application

3. Make sure this application depends on MyLibrary.

Build the Web service application

To build and install the project from WebSphere Integration Developer, perform the following steps:

1. Build the project `WSMQJMSServiceProvider`.
2. Choose `File > Export > Export project to integration module export` to export the project to an .ear file: `WSMQJMSServiceProvider.ear`.
3. Start the WebSphere Process Server or WebSphere Enterprise Service Bus runtime.
4. Install the .ear file from the WebSphere Application Server admin console.

Install and modify the application to use WebSphere MQ JMS

Some resources of the application should be modified after installation via the admin console. Since WebSphere Process Server and WebSphere Enterprise Service Bus V7.0, you can not modify the resources during installation time via the admin console.

Deploy steps:
1. Install the **WSMQJMSServiceProvider.ear** file as default without changing anything.
2. Change the resources after installation; go to the list of the installed applications.
3. Double-click the **WSMQJMSServiceProviderApp** application to the configuration, as shown in Figure 3.

**Figure 3. Application configuration**

![Application configuration](image)

4. Click **Message Driven Bean listener bindings** in the configuration as highlight 1 in Figure 3.
5. Change the **Activation Specification Target Resource JNDI Name** to **MQAS1**, as shown in Figure 4.

**Figure 4. Configure Activation Specification for SOAP/MQJMS export**

![Activation configuration](image)

6. Click **Resource references** in the configuration as highlight 2 in Figure 3.
7. Mapping JNDI names and WebSphere MQ resources by changing the **Target Resource JNDI Name** for export EJB MessageDriven_ProviderSOAPMQJMSExport_CustomerServiceJmsPort from jms/WebServicesReplyQCF to **replyMQCF**, as shown in Figure 5.

**Figure 5. Mapping JNDI names and WebSphere MQ resources**

8. Start the application.

**Create an application with import and export that has Web service SOAP over JMS bindings**

The application covered here includes:

- An export with a Web service SOAP over JMS binding, used for receiving SOAP messages and converting them into business objects.
- A Java component implementation, which prints the data field in the business object.
- An import with a Web service SOAP over JMS binding, which invokes the WSMQJMSServiceProvider.

The application is created in WebSphere Integration Developer and is shown in Figure 6.

**Figure 6. The WSMQJMSSBackEndService application**

After you create this application, perform the following steps:

1. Select **CustomerServiceImport1**, and modify the destination and connection factory values of the address property, as shown in Figure 7. The updated value for the address property is:

**Listing 2. The Web service import with SOAP over JMS binding address property**

```
jms://queue?destination=requestQ1&
connectionFactory=requestMQCF&
targetService=ProviderSOAPMQJMSExport_CustomerServiceJmsPort
```
2. Build the application.
3. Export the application into an .ear file: WSMQJMSBackEndService.ear.
4. Install the .ear file as default without changing anything.
5. Configure WSMQJMSBackEndServiceApp as already did for WSMQJMSServiceProviderApp.
6. Change the Activation Specification Target Resource JNDI Name as step 6 for WSMQJMSServiceProviderApp, this time the name is MQAS.
7. Change the map resource references to resources step in the same way as step 8 for WSMQJMSServiceProviderApp, and change the export EJB MessageDriven_BackEndSOAPMQJMSExport_CustomerServiceJmsPort JNDI name from jms/WebServicesReplyQCF to replyMQCF, and WebServiceReference JNDI name from jms/ProviderSOAPMQJMSExportQCF to requestMQCF.

Using a permanent queue for replies at an import

By default when a request/reponse operation is invoked via an import with a SOAP/JMS binding, the runtime creates a temporary queue for the response. This queue is passed as the replyTo queue in the JMS message sent to the SOAP/JMS service. The SOAP/JMS binding then waits for a response message to be put on the temporary reply queue, and finally discards the queue once the reply has been read. Temporary queues are not able to hold persistent messages, so if you need a permanent reply queue to be used by an import for this or other reasons, it is possible to identify the queue to be used via the URL configured on the import binding. An example of the URL syntax for specifying the reply queue is:

Listing 3. Example of URL syntax for specifying the reply queue

```
jms:/queue?destination=requestQ1&connectionFactory=requestMQCF&targetService=ProviderSOAPMQJMSExport_CustomerServiceJmsPort&replyToDestination=permanentReplyQ
```

You would need to ensure the permanentReplyQ JMS resource and corresponding MQ queue are created prior to using an import configured in this way.

Create a SOAP/MQ JMS client application

This section describes the required behaviour of client applications that send SOAP messages to the WebSphere MQ JMS destination, which then correspond to the reconfigured export from earlier steps. The format and the properties of the SOAP messages have to be provided in an appropriate way so that the messages are consumed correctly by the SOAP over JMS...
export. Listing 3 shows an example of a SOAP message for a one-way message using the CustomerService interface defined earlier.

**Listing 4. Example of a SOAP JMS message**

```xml
    xmlns:q0="http://MyLibrary/CustomerService"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <q0:createCustomer>
      <id>123</id>
    </q0:createCustomer>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following three JMS user properties must be set in messages sent to a SOAP over JMS export:

- **targetFunctionName**: The name of the operation (function) in the CustomerService interface to be invoked. In this example the value must be either `createCustomer` or `getCustomerInfo`.
- **JMSReplyTo**: The destination for the response message is set to `replyQ`.
- **targetService**: The Web service port name. The value must be `BackEndSOAPMQJMSExport_CustomerServiceJmsPort`. Figure 8 shows an example of a client application implemented as a JSP that sends a SOAP message to the requestQ and gets the response back from the replyQ.

**Figure 8. Client application sends a request message**
After clicking the `sendMessage` button, the runtime displays in a console window, as shown in Figure 9, and a response message is received back to the JSP client, as shown in Figure 10.

**Figure 9. Runtime result from console window**

![Runtime result from console window](image)

**Figure 10. Response message received from the client application**

![Response message received from the client application](image)

**Resources required for publish/subscribe messaging**

This section describes how to use the publish/subscribe model with imports and exports with Web service SOAP over JMS bindings, using the WebSphere MQ JMS provider.

**Create WebSphere MQ resources**

The publish/subscribe model is designed for an application (the publisher) to send messages to a set of interested applications (the subscribers). Subscribers register their interest in a particular topic, and publishers indicate the topic relevant to their messages. A broker sits in between the publisher and subscribers and sends a copy of each message under a certain topic to the appropriate subscribers. WebSphere MQ includes an embedded broker. To use the publish/
subscribe model with WebSphere MQ, you need to configure the following WebSphere MQ resources from the WebSphere Enterprise Service Bus or WebSphere Process Server admin console at the Node scope:

1. Create a WebSphere MQ JMS topic connection factory, as shown in Figure 11. The broker version is basic, letting you use the WebSphere MQ embedded broker.

**Figure 11. Configuration of the WebSphere MQ topic connection factory**

1. Create a WebSphere MQ JMS topic destination called `MQTopicDest` with a JNDI name, `MQTopicDest`, and a basic topic name, `Greetings`, as shown in Figure 12.
3. Create an activation specification called `MQTopicAS`, which uses `MQTopicCF`, as shown in Figure 13.
Figure 13. Configuration of the WebSphere MQ topic activation specification

Configure a WebSphere MQ broker service

To use the WebSphere MQ broker service, there's an initial setup step required only the first time you use the broker. This creates a number of queues, including SYSTEM.BROKER.CONTROL.QUEUE, which is used to handle subscription messages:

```bash
$WMQ_INSTALL\Java\bin runmqsc QMGR <MQJMS_PSQ.mqsc
```

An application scenario for the publish/subscribe model

This application scenario is designed for both a publisher and a subscriber so that they can use WebSphere MQ resources to communicate with SOAP messages. Figure 14 is an example of the application scenario. It includes a WSMQJMSPublisher application and a WSMQJMSSubscriber application. WSMQJMSPublisher is a client application for publishing messages to the
MQTopicDest topic. WSMQJMSSubscriber is a Web service application that subscribes to the MQTopicDest topic to receive SOAP messages.

Figure 14. Application scenario for the publish/subscribe model

Create an interface with a one-way operation

When using publish/subscribe with an import or export with Web service SOAP over JMS binding, only one-way operations are supported. So an interface with only one-way operations is required. Create the interface using the following steps:

1. Create a new interface called GreetingService under MyLibrary.
2. Create a one-way operation in the interface: void sayHello(String message);
3. Save and build MyLibrary.

Create a subscriber application

1. In WebSphere Integration Developer, create a new module called WSMQJMSSubscriber with an export with a Webservice SOAP over JMS binding called SubscriberSOAPMQJMSExport and a Java component called HandleTopic, as shown in Figure 15.

Figure 15. Example of a Web service application

This module illustrates the use of a WebSphere MQ JMS topic by an export with a Web service SOAP over JMS binding.

2. Open the WSDL file called WSMQJMSSubscriber_SubscriberSOAPMQJMSExport.wsdl in MyLibrary, and change the message type from queue to topic so the SubscriberSOAPMQJMSExport binding address property looks like Listing 4.

Listing 5. A web service export with SOAP JMS binding address property

```java
jms:/topic?destination=SubscriberSOAPMQJMSExport&
connectionFactory=SubscriberSOAPMQJMSExportQCF&
targetService=SubscriberSOAPMQJMSExport_GreetingServiceJmsPort
```
You can't change this from the SOAP over JMS export property window shown in Figure 16.

**Figure 16. WSMQJMS Export property**

![WSMQJMS Export property](image)

This ensures that the export knows to subscribe to a topic to receive request messages. The implementation of the `HandleTopic` method simply prints out a message in the method—`sayHello(String message)`—, which is passed by SubscriberSOAPMQJMSExport.

3. Save the changes to build the WSMQJMSSubscriber application.

**Deploy and install the subscriber application**

Export the WSMQJMSSubscriber application to an EAR file, and install it from the WebSphere Enterprise Service Bus or WebSphere Process Server admin console. The steps are the same as before and install and modify the application to use WebSphere MQ JMS section).

**Create a publisher application**

This section describes how to create an example publisher application, which makes use of an import with a Web service SOAP over JMS binding. We've chosen to create a module that contains only an import with Web service SOAP over JMS binding. The following steps show you how to create one:

1. Create a new module called WSMQJMSPublisher.
3. Drag the SubscriberSOAPMQJMSExport_GreetingServiceJmsPort onto the WSMQJMSPublisher Assembly Diagram to create an import with the Web service SOAP over JMS binding.
4. Change the import component name into GreetingServiceSOAPMQJMSImport.
5. Change the value of the Address property to what's shown in Listing 5.

**Listing 6. Web service import with SOAP over JMS binding address property for a topic destination**

```xml
jms:/topic?destination=SubscriberSOAPMQJMSExport&
connectionFactory=SubscriberSOAPMQJMSExportQCF&
targetService=SubscriberSOAPMQJMSExport_GreetingServiceJmsPort
```

You can directly change this from the import's Web service binding property window, shown in Figure 17.
6. Build the module and export it as an EAR file.

**Test the publisher application from WebSphere Integration Developer**

From WebSphere Integration Developer, you add the publisher application to the currently running server:

1. Right-click the project **TestWSMQJMSPub** and choose **Test > Test Module** (see Figure 18).

**Figure 18. Test publisher result**

2. Enter a string, such as **hello**, in the message field; the message is then published under the topic **Greetings**.

3. The subscriber receives the message and prints it out exactly as shown in the console window.

**Conclusion**

This article used both the point-to-point messaging model and the publish/subscribe messaging model to demonstrate how to configure WebSphere MQ resources used by Web service SOAP...
over JMS bindings. For the point-to-point messaging model, you change the export to use a WebSphere MQ JMS message activation specification and modify the export reply connection factory into a WebSphere MQ JMS connection factory after the application installation. For the import, you only need to modify the address property: change the destination to specify into a predefined WebSphere MQ JMS queue destination, and change the connection factory to specify a predefined WebSphere MQ JMS connection factory before the application installation. For a request/response message, the client application has to set the JMSReplyTo header in JMS messages.

For the publish/subscribe message model, you change the export binding address property to make the destination type **topic**. You also need to change the export activation specification after the application installation time. For the import, you change the destination to specify a WebSphere MQ JMS topic destination and change the connection factory to specify a WebSphere MQ JMS topic connection factory before the application installation.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications for point-to-point messaging</td>
<td>SOAPMQJMS2P.zip</td>
<td>41KB</td>
</tr>
<tr>
<td>Applications for publish subscribe messaging</td>
<td>SOAPMQJMSPubSub.zip</td>
<td>26KB</td>
</tr>
</tbody>
</table>
Related topics

- Building a JMS Web service using SOAP over JMS and WebSphere Studio
- Read the article, "Building a JMS Web service using SOAP over JMS and WebSphere Studio" (developerWorks, Feb 2004).
- Get more information in the article, "Web services client programming for WebSphere Process Server" (developerWorks, Aug 2006).
- Check out the IBM WebSphere Developer Technical Journal, "Deploying publish and subscribe applications into the Service Integration Bus" (developerWorks, Aug 2005).
- Evaluate IBM products in the way that suits you best.

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)