Web Services JMS Gateway for WMQI

Shahryar Sedghi
IBM Software Group Services for Websphere
ssedghi@us.ibm.com

Table of Contents

Date	Version	Name	Descrition
12/8/2002	Draft 1.0	Shahryar Sedghi	**Initial release**
12/18/2002	Draft 1.1	Shahryar Sedghi	**Tomcat setup**

1. Who this document is for

This document is for WMQI, developers and administrators who want to enable WMQI to receive SOAP messages over HTTP and some other protocols that are not currently supported by WMQI but supported by Axis.

Familiarity with the following concepts is helpful:

- JMS administration: JMSAdmin Utility
- Web Services Concept and Axis architecture
- WSDL and XML Schema definitions (using WSAD-IE editors are helpful for both beginners and advanced users)
- Java programming or familiarity with Java runtime environment

2. Introduction

Web Services JMS gateway is a front-end multithreaded process for a WMQI broker. The Gateway:

- Converts incoming SOAP Request over HTTP or other Axis transports to JMS, and sends it to a WMQI input queue
- On return receives the outgoing SOAP responses from a WMQI output queue based on JMS correlation ID of the request message and sends it back using the same Axis instance.
- Supports both RPC and document SOAP transactions.
- Uses JNDI for JMS administrative objects to provide maximum flexibility for setting different JMS parameters.
- Although the Gateway is designed to work as a front end for WMQI which requires Websphere MQ as a JMS provider, the gateway itself does not have any dependency on any JMS provider.
- Can use either Websphere MQ server or Client, depending on JNDI settings for QCF.

3. Installation

This section explains requirements and procedures of gateway installation and setup.

2.1 Prerequisites

The following components are required to run the gateway:

- A JDK 1.3.1 or higher (using the latest JDK 1.3 is recommended)
- Axis (1.0) Jar files
- A Standard XML parser (Xerces is recommended)
- An Application Server if and transport other than HTTP is required (see Jakarta Tomcat setup procedure in Section 2.6)
- Websphere MQ (V5.3 is strongly recommended)
- Websphere MQI(V2.1 CSD3 is required)

2.2 Where to get the prerequisites

Download:

- Xerces and Axis from www.apache.org
- JDK from sun.java.com
- Jakarta Tomcat from www.apache.org

Note: Websphere MQ® and Websphere MQI® are licensed programs

2.3 Files in this package

The following files are packaged in a compressed format (zip, gz or executable)

com.ibm.axis.jms jar file	contains the code required for running the gateway
IA81Echo.xml	the message flow to test the gateway
hello.wsdl	WSDL file to test the gateway
SayHello.java	A sample client java program to test the package
Global_config.wsdd	Deployment Descriptor for JMS global parameters
hello_server.wsdd	Sample Deployment descriptor for the Hello program
IA81JmaScript.txt	A JMSAdmin utility input script to create the JMS administrative objects
JMSAdmin.config	A JMSAdmin configuration file matching other examples
setCp.bat	A windows batch file to set the required classpath
simpleStartup.bat	A Windows batch file to set the classpath and start simple http listener

2.4 Download and setup

Notes:

1. This procedure assumes Win2000 as the environment, minor differences can be found in UNIX procedures
2. Websphere MQ must have installed and configured prior to this configuration

1. Set an environment variable MQ_HOME to the path of the directory into which you have installed Websphere MQ

- **AXIS and Xerces, Required. Xerces also comes with Tomcat**

2. Create a directory for Axis
3. Download Axis 1.0 compressed file from http://www.apache.org and extract the files into the directory which you created in the previous step
4. Set an environment variable AXIS_HOME to the path of the directory into which you have installed Axis 1.0
5. Create a directory for Xerces
6. Download Xerces 2.0 compressed file from http://www.apache.org and extract the files into the directory which you created in the previous step
7. Set an environment variable XERCES_HOME to the path of the directory into which you have installed Axis

- **JDK, If you already have a JDK 1.3.1 or over go to step 9**

8. Download JDK from http://java.sun.com/j2se and install it. It will create a directory during the installation
9. Set an environment variable JAVA_HOME to the path of the directory into which you have installed JDK

- **Tomcat, If you want to use the gateway with Tomcat, otherwise go to step 13**

10. Create a directory for Jakarta Tomcat 4.11
11. Download Jakarta Tomcat 4.11 compressed file http://www.apache.org and extract the files into the directory which you created in the previous step
12. Set an environment variable CATALINA_HOME to the path of the directory into which you have installed Tomcat 4.11

- **The package itself, required**

13. In this step you may either create a directory for the gateway package extraction or it is better to use %CATALINA_HOME%\webapps\axis\WEB-INF directory than creating one from scratch. In this case Tomcat and simple HTTP listener can share the same Deployment descriptive file for the unit test and system test environment. Tomcat and simple
14. Extract the gateway files into the directory of the previous step
15. Create an environment variable WJGWY_HOME to the path of the directory into which you have extracted the gateway
16. Copy the content of %AXIS_HOME%\webapps to %CATALINA_HOME%\webapps
17. Copy **every file** in %MQ_HOME% \java\lib to %CATALINA_HOME%\webapps\axis\lib **except**:
    - com.ibm.mqbind.jar -- back level
    - postcard.jar -- not required
    - jndi.jar -- comes with Tomcat

2.5 Setup for Simple HTTP Server

Axis Simple HTTP server is a single threaded server used for demo and development purposes. If you want to use this gateway in production, you need to setup Axis under an application server like Tomcat.

- Change the current directory to WJGWY_HOME(i.e cd %WJGWY_HOME% in DOS command prompt)
- Modify the supplied batch file"setCp.bat" to reflect the AXIS_HOME and MQ_HOME, XERCES_HOME and WJGWY_HOME variables. If you have already created the environment variables you can delete them from batch file.
- Run the batch file "simpleStartup.bat" to start the simple HTTP Server. This command creates a separate windows for HTTP listener
- To test the server, enter the command "java org.apache.axis.client.AdminClient list" from the original window. It must return the standard deployed services and configuration settings.

2.6 Setup for Jakarta Tomcat

Simple HTTP Server is relying on system "classpath" and its working directory while Tomcat is using standard classloaders. To know how Tomcat classloader works, refer to: %CATALINA_HOME%\/webapps/tomcat-docs/class-loader-howto.html

If you have followed the procedure in section 3.4 no additional setup is required. The following rule applies to the gateway code and every class that is generated by WSDL2Java:

- Copy the Jar files to %CATALINA_HOME%\webapps\axis\lib
- Copy the directory hierarchy of the classes to CATALINA_HOME\webapps\axi\classes

2.6.1 Startup and Testing

1. To start Tomcat either go to Windows services and start the "Apache Tomcat" service or run the batch file "startup.bat" in %CATALINA_HOME%\bin
2. To test the server, enter the command "java org.apache.axis.client.AdminClient list" from the original window. It must return the standard deployed services and configuration settings.

2.7 Deploying Global Parameters

- Change the current directory to WJGWY_HOME

- Modify global_config.wsdd to reflect your JNDI settings, if you are not familiar with JNDI leave the parameters unchanged. Default parameters assume you are using a file context JNDI, with the objects stored in the directory c:\JMSNames. If you are using default settings, create the mentioned directory before accessing any services.
- Run “java org.apache.axis.client.AdminClient global_config.wsdd” to deploy global options

4. How it works

Web services JMS Gateway is implemented as an Axis provider. Like RPC and Message providers receives the control on the server side:

- Instead of invoking the java object that is already loaded by Axis server, retrieves the message from message context
- Converts it to JMS Text message and send to a JMS request queue according to the definition in the service deployment.
- Waits for the reply on a response queue, with a correlation Id equal to the message Id of the sent message
- On return receives the message from response queue and return it to the Axis engine.

All the required parameters for JMS are set as parameters in the server side deployment descriptor:

- JMS Global parameters such as QCF (jmsQcf), ICF(jmsIcf) , URL(jmsUrl) and timeout (jmsTimeOut) are set as global configuration parameters.
- For each service tow parameters jmsReqQueue and jmsRespeQueue are used to set the JNDI names for each service.
- Although the server side classes are not required for the gateway operation, Axis needs them to validate the method signature of the incoming SOAP request. Therefore the skeleton of the server side classes along with the serialization classes are required in the same place as normally Axis server objects reside.

5. Service Deployment

Web Services JMS Gateway relies on Axis tooling for service deployment. The main tool for this purpose is WSDL2Java tool that generates necessary files required by both Axis server and client. Files generated by WSDL2Java used for four different purposes:

- Stub Java classes to be used by client side
- Data Type Java classes to be used by both server and clients
- Server side skeleton java class, which can be used to implement a real server side service. We need to keep the skeleton accessible to the server, but gateway does not require it.
- Web Services Deployment Descriptor (WSDD) file that is used for service deployment.

WSDD file, which is generated for the service needs some modifications to reflect the gateway JMS requirement before deployment. The modifications are as follows:

- The “Provider” attribute in the “service” element must be changed to “Handler” from “java:RPC” or “java:MSG”
- Three parameters must be added as follows:

```
<parameter name="handlerClass"
            value="com.ibm.axis.jms.JMSProvider"/>
<parameter name="jmsReqQueue" value="ia81EchoIn"/>
<parameter name="jmsRespQueue" value="ia81EchoOut"/>
```

Where ia81Echoin and ia81EchoOut are request and response JNDI queues names for sample Echo program comes with the package. These queue names must be set for each service created.

6. Testing the Gateway with WMQI

A “hello world” sample is provided in this package that:

- Send a SOAP message over HTTP to the gateway
- Gateway receives the message and sends it over JMS to WMQI
- WMQI echoes back the message
- Gateway sends it back over HTTP to the service requestor

The following procedure applies not only to this sample but also to every service that is using the gateway with Simple HTTP server or Tomcat to reach WMQI. The following procedure applies to both Simple HTTP Server and Jakarta Tomcat:

1. Create two queues in your queue manager named “IA81.REQUEST.QUEUE” and .REQUEST.QUEUE
2. Import IA81Echo.xml into WMQI control center, check-in, assign and deploy it
3. Copy setCp.bat file to somewhere in the system “PATH”
4. run “setCp.bat”
5. Rename the original JMSAdmin.config file in “%MQ_HOME%\Java\bin”
6. Copy the supplied JMSAdmin.config file to the same directory
7. Create the directory “C:\JMSNames”
8. Modify IA81JmaScript.txt to reflect your environment. Follow the instruction
9. CD to “%MQ_HOME%\Java\bin”
10. Run “JMSAdmin < %WJGWY_HOME%\IA81JmaScript.txt “
11. CD back to “%WJGWY_HOME%”

6.1 Simple HTTP Server

1. Run “java org.apache.axis.wsdl.WSDL2Java hello.wsdl –o. –p ia81.stubs –s”
2. Compile the classes generated by WSDL2Java, they are located at ia81/stubs, run “javac ia81/stubs/*.java”
3. Modify the “ia81\stubs\deploy.wsdd “WSDD file as mentioned in section 5, it must be similar to hello.wsdd which comes with the package
4. run “java org.apache.axis.client.AdminClient ia81\stubs\deploy.wsdd”
5. Compile the client program “javac SayHello.java”
6. run the program “java SayHello”, you must see the “Hello World” on the console

6.2 Jakarta Tomcat

1. Run “java org.apache.axis.wsdl.WSDL2Java hello.wsdl –o %CATALINA_HOME%\webapps\axis\classes –p ia81.stubs –s”
2. Compile the classes generated by WSDL2Java, they are located at ./classes/ia81/stubs, run “javac classes/ia81/stubs/*.java”
3. Modify the “%CATALINA_HOME%\webapps\axis\classes\ia81\stubs\deploy.wsdd “WSDD file as mentioned in section 5, it must be similar to hello.wsdd which comes with the package
4. run “java org.apache.axis.client.AdminClient classes\ia81\stubs\deploy.wsdd”
5. Compile the client program “javac SayHello.java”
6. run the program “java SayHello”, you must see the “Hello World” on the console

Notes: To know more about Axis Utilities refer to “%AXIS_HOME%\docs\ user-guide.html”