Create a Simple HTTP Web Services Gateway Service with WebSphere Application Server V6

Bob O’Hanlon  
July 26, 2005  
(First published February 23, 2005)

This article is describes how to create a Web services gateway, configure existing back-end Web services, and then test their invocation over the SOAP/HTTP protocol in IBM® WebSphere® Application Server V6.

Overview

The Web services gateway available in WebSphere Application Server V5, has been re-based on the Service Integration Bus (hereafter referred to as Bus) and Web services enablement in V6. The gateway is essentially a tool that gives administrators an easy way to map Web services across the Bus. It enables the mapping of different protocols. In this article, one protocol (SOAP/HTTP) is used for the Web service invocation. A SOAP/HTTP request comes into the gateway and is then re-targeted to an external service, also over SOAP/HTTP. This article guides you through the process of creating a gateway service.

The configuration

The planned configuration is shown in Figure 1. The diagram depicts how request and response messages flow through the Bus, and highlights the different steps required for configuring a gateway service, as follows:

1. Create an endpoint listener and reply destination and connect it to the Bus.
2. Create a gateway instance on the Bus.
3. Create a gateway service on the gateway instance, including creating and configuring:
   - Request and response destinations for the gateway
   - An outbound service and an outbound service destination
   - An outbound port and an outbound port destination
   - An inbound port
We’ll complete the configuration using the Application Server administration console. This article assumes you’ve already configured a Bus and completed the steps in the SIBWS - Completing the installation in the WebSphere Application Server Information Center. In this article, we’ll use DeveloperWorksBus as the SIBus name.

Create an endpoint listener

The endpoint listener is the entry point for your SOAP/HTTP message into the gateway. You’ll need to configure the listener before you create your gateway service.

1. From the administration console navigation pane, select **Servers => Application servers**.
2. In the content pane, select **server_name**.
3. Under **Additional Properties**, select **Endpoint Listeners => New**.
Create a Simple HTTP Web Services Gateway Service with WebSphere Application Server V6

4. On the Endpoint Listeners page, specify the following values:
   - **Name**: SOAPHTTPChannel1
   - **URL Root**: http://host_name:port_number/wsgwsoaphttp1/ This will be the context route of the service endpoint.
   - **WSDL serving HTTP URL Root**: http://host_name:port_number/sibws. This will be the context route of the WSDL address.

5. Click **OK**.
6. Select the endpoint listener **SOAPHTTPChannel1**.
7. Under **Additional Properties**, select **Connection Properties**.
8. Select **New**, then select **DeveloperWorksBus**
9. Click **OK**, then save the configuration.

Create a gateway instance

The gateway instance provides you with a single point of administration for your Web service requests, where access to Web services can be enabled and controlled for selected users. Before you can create a gateway service, you have to create the gateway instance.

1. From the administration console navigation pane, select **Service integration => Buses**.
2. In the content pane, select **DeveloperWorksBus**.
3. Under **Additional Properties**, select **Web service gateway instances**.
4. On the Web service gateway instances page, select **New**.
Create a gateway service

When you create a gateway service, you need to associate it with a single external service. You do this by supplying the external service WSDL when creating the gateway service. The gateway service is created from within the administrative functions of a gateway instance. If you are using the sample supplied with this article, you'll need to install the application StringService.ear.

1. From the navigation pane, select **Service integration => Buses**.
2. In the content pane, select **DeveloperWorksBus**.
3. Under **Additional Properties**, select **Web service gateway instances**.
4. On the Web service gateway instances page, select **DeveloperWorksGateway**.
5. On the DeveloperWorksGateway page, under **Additional Properties**, select **Gateway services**.
6. On the Gateway services page, select **New**.
7. Select **WSDL-defined Web service provider** as the type of target service.
8. Click **Next**. The New Gateway Service wizard opens. The following sections describe how to complete the wizard to create a gateway service.

**Step 1: Specify gateway service and destination names**

In this step, we'll specify names for the gateway service and destinations.

**Figure 4. Specify gateway service and destination names**

1. Enter these values:
   - **Gateway service name**: `DeveloperWorksGatewayService`. (This is an arbitrary name; you can name your gateway service as you wish.)
   - **Gateway request destination name**: `DWGSRequestDestination`. This destination processes request messages for the gateway.
   - **Gateway response destination name**: `DWGResponseDestination`. This destination processes response messages for the gateway. (Gateway response and request destination names are arbitrary.)

   **Note**: In Application Server, a destination is defined as a virtual location within a service integration bus, to which applications attach as producers, consumers, or both to exchange messages.

   We're not using mediations, so only the above values are required.

2. Click **Next**.

**Step 2: Locate the target service WSDL**

In this step, we'll supply the external service WSDL address.
Figure 5. Locating the target service WSDL

1. Enter these values:
   - **WSDL location type**: URL
   - **WSDL location**: The WSDL URL or the WSDL file that you want to use. If you're using the sample application, the URL is: http://host_name:port_number/StringService/wsdl/stringService.wsdl

2. Click **Next**.

**Step 3: Select the service**

In this step, we'll select the service in the WSDL we want to configure. The sample application WSDL has only one service, which is selected by default.
Click **Next**.

**Step 4: Select ports**

In this step, we'll select the port in the WSDL we want to configure. The sample application WSDL has only one port.
Select the default port and click **Next**.

**Step 5: Name the service and port destinations**

In this step, we'll set up the external service as an outbound service, and associate its destination with the gateway request and reply destinations we created in step 1. The wizard sets some defaults for you on this page based on the WSDL. You can override these values as desired.
**Figure 8. Name service and port destinations**

1. If you are using the sample application, enter these values:
   - **Outbound service name**: StringServiceOutboundService
   - **Service destination name**: StringServiceOutboundServiceDestination
   - **Port destination name**: StringServiceOutboundPortDestination

2. Click Next.

**Step 6: Select points for service and port destinations**

In this step, we'll select the Bus member to use. (When you initially created your Bus, you would have added the server to the Bus in order for it to become a Bus member.)
Select the Bus member and click Next.

**Step 7: Select endpoint listeners**

In this step, we'll specify the endpoint listener we created earlier, SOAPHTTPChannel1.

**Figure 10. Select endpoint listeners**
Click **Next**.

**Step 8: Define UDDI publication properties**

If you want to publish the service to a UDDI, you can do so in this step.

**Figure 11. Define UDDI publication properties**

Select **Finish** and save your changes.

**Test the gateway service**

We've now finished configuring our gateway service and are ready to test it. As in previous releases, the gateway has generated some new WSDL that can be exposed to service requesters. If you’re using the sample, you can view the WSDL interface at this address: http://host_name:port_number/sibws/wsdl/DeveloperWorksBus/DeveloperWorksGatewayService

You can view the WSDL binding and port type definitions by appending ?wsdl=bindings or ?wsdl=porttypes to this URL respectively.

If you’re using the sample client, you can derive from the WSDL interface and bindings the required parameters for a Dynamic Invocation Interface (DII) JAX-RPC invocation of this gateway service, as follows:

- **Operation namespace**: http://service.string
- **Service namespace**: http://developer.works/Service
- **Endpoint**: http://host_name:port_number/sgwssoaphttp1/soaphttpengine/DeveloperWorksBus_DeveloperWorksGatewayService/SOAPHTTPChannel1InboundPort
Test the gateway service using the stringClient with these parameters:

```java
java stringClient <Endpoint_Address> <Operation_Namespace> <Service_Namespace> <string>
```

The client sends a string, and the service should successfully return the string you sent.

**Conclusion**

We've now used the Web services gateway to enable an existing Web service as a destination in the Bus and then exposed that destination as a gateway Web service using the wizard. As you worked through the wizard, all the configuration artifacts were created and configured for you. You can achieve the same results by separately configuring outbound and inbound services and ports in the Application Server administration console, but the method shown here is much simpler. In addition you now have the benefits of using a gateway service.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample code</td>
<td>DevWorks_Gateway_Sample.zip</td>
<td>1.6 KB</td>
</tr>
</tbody>
</table>
Related topics

- For information on JAX-RPC, see Java API for XML-Based RPC (JAX-RPC).
- For more information about the benefits of gateway services, check out the WebSphere Application Server InfoCenter.
- Building an Enterprise Service Bus with WebSphere Application Server V6 -- Part 2
- WebSphere Application Server InfoCenter SIBWS - Completing the installation

© Copyright IBM Corporation 2005
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)