Configure secure Web services communications through an ESB

Web services security using WebSphere Service Integration technologies

Skill Level: Advanced

**Tendai Chinoda** (tendai@us.ibm.com)
Advisory Software Engineer
IBM

**Vivek Gadiyar** (gadiyar@us.ibm.com)
Architect
IBM

18 Nov 2005

Use IBM WebSphere® Application Server V6.0 Service Integration technologies to secure SOAP over HTTP requests to an example Bank Application Web service. The example provides authentication of SOAP/HTTP user requests and authorization of SOAP over HTTP user requests for specific operations. It shows you how to provide secure internet accessibility to the Bank Application so that customers can retrieve their account information.

Section 1. Before you start

About this series

IBM WebSphere® Application Server V6.0 Service Integration technologies facilitate the enablement and deployment of Web services accessible through Simple Object Access Protocol (SOAP) over HyperText Transport Protocol (HTTP) and Java Messaging Service (JMS). These two transports enable Web services clients and servers to communicate using JMS Queues/Topics and HTTP.

The prerequisite to this tutorial, “Transform protocols and route messages through
an ESB," demonstrated how to deploy a SOAP/JMS Stateless Session Bean Web
service implementation in WebSphere V6.0 and provide SOAP/JMS access using
configured Service Integration Bus (SIB) messaging resources. It also demonstrated
how to configure SIBWS inbound and outbound services to provide routing and
protocol transformation of SOAP/HTTP service requests to the same SOAP/JMS
target service.

This tutorial extends the example banking application to demonstrate how to
address a new business requirement to provide secure internet accessibility to the
banking application so that customers can retrieve their account information. The
scope is as follows:

- Demonstrates secure communications, in accordance with the Web
  Services Security (WS-Security) 1.0 specification, between the requestor
  and the SIB.
- Enables user authentication and operation-level authorization for requests
to target service.

About this tutorial

IBM WebSphere Application Server V6.0 Service Integration technologies facilitate
secure Web services communication in accordance with the Web Services Security
(WS-Security) 1.0 specification (see Resources). The secured Web service
communications can be configured between the requestor and the Service
Integration Bus as well as between the Service Integration Bus and the provider.

In the tutorial, "Transform protocols and route messages through an ESB" (and we
recommend that you look over this tutorial before continuing with this one -- see
Prerequisites or Resources for a link), we demonstrated the protocol transformation
and routing capabilities of Service Integration technologies in WebSphere
Application Server V6.0. In this tutorial, we modify the solution that was included in
that tutorial. We do this by securing communications between the SOAP/HTTP
requestor and the inbound service.

Prerequisites

This tutorial assumes that you are familiar with the tutorial "Transform protocols and
route messages through an ESB."

Section 2. Background: Enterprise Service Bus overview

An Enterprise Service Bus (ESB) is an architectural construct and middleware
infrastructure component that supports Service-Oriented Architecture (SOA). Within the WebSphere Business Integration Reference Architecture, the ESB is positioned to provide interconnectivity services. Run-time implementations of an ESB are realized by using the following IBM products (see Resources for available trial versions):

- WebSphere MQ
- WebSphere Message Broker/Event Broker
- Web Services Gateway
- WebSphere Application Server V6

When realizing an ESB implementation with WebSphere Application Server V6.0, the Service Integration technologies play a primary role in facilitating application connectivity. In particular, the WebSphere Application Server V6.0 Service Integration Bus (SIB) for messaging and the Service Integration Bus Web services enablement (SIBWS) for routing, protocol transformation, and security services of the ESB. A detailed discussion on the WebSphere Business Integration Reference Architecture, ESB implementations, and the WebSphere Application Server V6.0 SIB is beyond the scope of this tutorial (see Resources for more details).

In this tutorial, the secured Web services communications from the SOAP/HTTP requestor are intercepted and handled by the SIBWS, which in turn utilizes the SIBWS routing and protocol transformation capabilities (inbound and outbound service configurations) to direct authenticated and authorized service requests to the target Bank Application Web service.

Section 3. About the Bank Application Web services implementation

The example used in this tutorial is a simplified banking J2EE application (BankApp.ear) consisting of an Enterprise JavaBean (EJB) Module (BankAppEJB.jar) with a Stateless Session Bean Web services implementation. The Stateless Session Bean serves as a session facade to the accounting application. The J2EE application also consists of a second EJB Module (BankAppEJBJMSRouter.jar) for routing the SOAP/JMS request to the Bank Application Web services implementation.

The BankAppEJB module consists of four main classes:

- Account.java: JavaBean representation of an Account record.
- Accounts.java: Interface for access to and manipulation of Account records.
• **AccountsImpl.java**: Implementation of Accounts interface allowing the creation and updating and retrieval of static Account objects stored in a static hashtable called `existingAccounts`. The `existingAccounts` variable is pre-populated with four default accounts.

• **AccountsSessionFacadeBean**: Stateless Session Bean facade to the Accounts Interface.

The **BankAppEJBJMSRouter** module consists of a single Message Driven Bean (MDB), **WebServicesJMSRouter**. The implementation class of the bean, `com.ibm.ws.webservices.engine.transport.jms.JMSListenerMDB`, is provided by WebSphere Application Server V6.0. The MDB processes incoming messages, invokes the Web service, and sends a reply message where necessary.

The banking application was previously deployed for use by bank representatives and for internal accessibility only. Changing business requirements have resulted in the need to provide internet accessibility to the Bank Application so that customers can retrieve their account information. Thus, the need to secure SOAP/HTTP requests to the target Bank Application Web service and provide:

1. Authentication of SOAP/HTTP user requests.
2. Authorization of SOAP/HTTP user requests for specific operations.

**Authentication security requirements**

**User Groups:**

bacustomers

bareps

**Authorization security requirements**

**User group to role mappings:**

bacustomers > Customer

bareps > Bank representative

**Operation to role mappings:**

Create Account (Bank representative)

Update Account (Bank representative)

Review Balance (Bank representative, Customer)

**Users and groups**
User definitions

<table>
<thead>
<tr>
<th>User name</th>
<th>Password</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>testadmin</td>
<td>WebSphere administrator ID</td>
</tr>
<tr>
<td>barep</td>
<td>testrep</td>
<td>Bank representative user ID</td>
</tr>
<tr>
<td>bacustomer1</td>
<td>testcustomer1</td>
<td>Bank customer user ID</td>
</tr>
<tr>
<td>bacustomer2</td>
<td>testcustomer2</td>
<td>Bank customer user ID</td>
</tr>
</tbody>
</table>

Group definitions

<table>
<thead>
<tr>
<th>Group name</th>
<th>Members</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admins</td>
<td>admin</td>
<td>WebSphere administrative group</td>
</tr>
<tr>
<td>bareps</td>
<td>barep</td>
<td>Bank representatives group</td>
</tr>
<tr>
<td>bacustomers</td>
<td>bacustomer1, bacustomer2</td>
<td>Bank customers group</td>
</tr>
</tbody>
</table>

Section 4. The proposed solution

The proposed solution is to install the SIBWS components and expose a SOAP/HTTP endpoint for client access. The SIBWS will then handle the security requirements for SOAP/HTTP requests prior to routing authenticated and authorized requests to the target service. See Figure 1.

Figure 1. Proposed solution
Section 5. Implement the solution

In this tutorial, you can follow the tasks outlined below:
1. Enable global security
2. Configure the SIBWS for secure communications with global security enabled
3. Create the WS-Security configuration resource
4. Create the WS-Security binding resource
5. Associate inbound port with WS-Security binding and configuration resources
6. Create operation-level security and role-based authorization EAR file
7. Import authorization EAR files for assembly
8. Enable J2EE role-based authorization
9. Complete assembly of the role-based authorization EAR file
10. Enable operation-level security
11. Deploy authorization EAR file
12. Test the Web services client authentication and authorization

Enable global security

In this section you enable global security. This tutorial uses the sample file custom registry provided with WebSphere Application Server V6.0.

1. Download the users.props and groups.props file provided with this tutorial.
2. Start the WebSphere Application Server V6.0 SimpleBankAppServer. From the command line enter the following:
   `<wasv60_home>/profiles/SimpleBankAppServer/bin/startServer.bat | sh server1`
   
   Note: If the server starts with errors they need to be resolved prior to continuing. Check the server logs for more details.
   `<wasv60_home>/profiles/SimpleBankAppServer/logs/server1/SystemOut.log`
   `<wasv60_home>/profiles/SimpleBankAppServer/logs/server1/SystemErr.log`

   If the cause of the failure is not obvious from the information in the logs, check the WebSphere Application Server documentation for more
information on troubleshooting failed server startups.

3. Access the Administrative Console for the server
   • From a browser, access the Administrative Console URL: http://localhost:9060/ibm/console.
   • Enter a User ID and click Login.

4. From the Administrative Console navigation pane, expand Security and click Global security.

5. In the User registries section click Custom.

6. Enter admin as the Server user ID and testadmin as the password.

7. Enter com.ibm.websphere.security.FileRegistrySample as the Custom registry class name and click Apply.

8. In the Additional properties section click Custom properties.

9. Click New to add a new property and enter groupsFile as the Name and <download_location>/groups.props as the Value and then click OK.

10. Click New to add a new property and enter usersFile as the Name and <download_location>/users.props as the Value and then click OK.

11. From the Administrative Console navigation pane, expand Security and click Global security.

12. In the Authentication section expand Authentication mechanisms and click LTPA.

13. Enter a password of your choice for the LTPA tokens and click OK.


15. Set the Active authentication mechanism to Light Weight Third Party Authentication (LTPA).

16. Set the Active user registry to Custom user registry and click OK.

17. Save configuration changes and logout from the Administrative console.

18. Stop and restart the server.

19. Access the Administrative Console and log in using the admin user ID and password from Step 6.

This completes the configuration tasks required to enable global security.
Configure the SIBWS for secure communications with global security enabled

In this section you configure the SIBWS with the required authentication aliases to allow for secure communications with the SIB when global security is enabled.

1. From the Administrative Console navigation pane, expand **Service integration** and click **Buses**.
2. Click **SimpleBankAppSIBus**.
3. In the **Related Items** section click **J2EE Connector Architecture (J2C) authentication data entries**.
4. Click **New** to create a J2C authentication alias.
5. Enter **sibwsalias** as the Alias.
6. Enter **admin** as the User ID and **testadmin** as the password and click **OK**.
7. Save configuration changes.
8. From the Administrative Console navigation pane, expand **Resources** and click **Resource adapters**.
9. Click **SIB_RA** and, in the **Additional properties** section, click **J2C activation specification** and then **SIBWS_OUTBOUND_MDB**.
10. In the Authentication alias pull down list, select **<your_nodename>/sibwsalias** and click **OK**.
11. From the Administrative Console navigation pane, expand **Resources** and click **Resource adapters**.
12. Click **SIB JMS Resource Adapter** and, in the **Additional properties** section, click **J2C activation specification** and then **SimpleBankAppActivationSpec**.
13. In the Authentication alias pull down list, select the **<your_nodename>/sibwsalias** and click **OK**.
14. Save configuration changes.
15. From the Administrative Console navigation pane, expand **Servers** and click **Application servers**.
16. Click **server1** and, in the **Additional properties** section, click **Endpoint listeners**.
17. Click SOAPHTTPChannel1 and, in the Additional Properties section, click Connection Properties.

18. Click SimpleBankAppSIBus.

19. Click New to create a new custom property.

20. Enter com.ibm.websphere.sib.webservices.EPLAuthAlias as the Name and sibwsalias as the Value and click OK.

21. From the Administrative Console navigation pane, expand Service Integration and click Buses.

22. Click SimpleBankAppSIBus.

23. In the Security section, uncheck the Secure check box and click OK. NOTE: In WebSphere Application Server Version 6, when you use the default messaging provider and global security is enabled for the server or cell, by default the service integration bus queue destination inherits the security characteristics of the server or cell and requires authentication of all clients. The SOAP/JMS clients provided with this tutorial is not configured for basic authentication; thus, the client requests will fail without disabling security on the SIB.

24. Save configuration changes.

This completes the configuration tasks required to allow the SIBWS to communicate with the SIB when global security is enabled.

Create the WS-Security configuration resource

In this section you create the WS-Security configuration resources used to secure the Web services requests between the service requesters (clients) and inbound services.

1. From the Administrative Console navigation pane, expand Service integration > Web services and click WS-Security configurations.

2. Click New.

3. Select 1.0 as the security version as shown in Figure 2 and click Next. Figure 2. WS-Security configuration security version
4. Select **Inbound** as shown in Figure 3 and click **Next**.

**Figure 3. Specify WS-Security configuration type**

5. Enter **BankAppWS-SecurityConfiguration** as the name as shown in Figure 4.

**Figure 4. WS-Security configuration name**
6. Click **Next** and then **Finish**.

7. Click **BankAppWS-SecurityConfiguration**.

8. In the **Request consumer** section click **Required Security Token** and then click **New**.

9. Enter **BankAppUsernameToken** as the **Name** and 
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken as the **Local Name** as shown in Figure 5.

**Figure 5. WS-Security configuration required security token**

10. Click **OK**.

12. In the Request consumer section click Caller and then click New.

13. Enter BankAppUsernameTokenCaller as the Name and http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken as the Local Name as shown in Figure 6.

   Figure 6. WS-Security configuration caller

14. Click OK.

15. Save configuration changes.

This completes the WS-Security configuration creation and configuration tasks.

Create the WS-Security binding resource

In this section you create the WS-Security bindings used to secure the Web services requests between the service requesters (clients) and inbound services. The WS-Security binding contains the implementation information for WS-Security configuration resources.

1. From the Administrative Console navigation pane, expand Service integration > Web services and click WS-Security bindings.

2. Click New.

3. Select 1.0 as the security version as shown in Figure 7 and click Next.

   Figure 7. WS-Security binding security version

Configure secure Web services communications through an ESB
© Copyright IBM Corporation 1994, 2007. All rights reserved.
4. Select **Request Consumer** as shown in Figure 8 and click **Next**.

   **Figure 8. WS-Security binding type**

   Use this wizard to configure a new WS-Security binding.

   - **Step 1: Select security version**
   - **Step 2: Specify binding type**
   - **Step 3: Specify WS-Security binding.**
   - **Step 4: Summary**

5. Enter **BankAppWS-SecurityBinding** as the **Name** as shown in Figure 9.

   **Figure 9. WS-Security binding name**
6. Click **Next** and then **Finish**.

7. Click **BankAppWS-SecurityBinding**.

8. In the **Additional Properties** section click **Token Consumers** and then **New**.

9. Enter the following as shown in **Figure 10**:
   General Properties > Token consumer name:
   
   **BankAppUsernameTokenConsumer**

   General Properties > Token consumer class:
   
   **com.ibm.wsspi.wssecurity.token.UsernameTokenConsumer**

   General Properties > Part reference name: (reference to WS-Security Configuration Required Token):

   **BankAppUsernameToken**

   Value Type > Local name:

   **http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken**

   **Figure 10. WS-Security binding token consumer configuration**
10. Click Apply.

11. In the Additional Properties section click JAAS Configuration.

12. In the JAAS Configuration name drop down window select system.wssecurity.UsernameToken and click OK.

13. Save configuration changes.

This completes the WS-Security binding creation and configuration tasks.

Associate inbound port with WS-Security binding and configuration resources

In this section you associate the SOAP/HTTP inbound port with the WS-Security configuration and bindings created in the previous sections.
1. From the Administrative Console navigation pane, expand **Service integration** and click **Buses**.

2. Click **SimpleBankAppSIBus**.

3. In the **Services** section click **Inbound Services** and then **BankAppInboundService**.

4. In the **Additional properties** section click **Inbound Ports** and then **SOAPHTTPChannel1InboundPort**.

5. In the **Security request binding** pull down, select **BankAppWS-SecurityBinding (1.0)**.

6. In the **Security configuration** pull down, select **BankAppWS-SecurityConfiguration (1.0)**. The resulting inbound port to WS-Security configuration and binding associations are shown in Figure 11.

**Figure 11. Inbound port to WS-Security configuration and binding association**

7. Click **OK**.

8. Save configuration changes.
This completes the tasks required to associate the inbound port with the WS-Security configuration binding elements.

Create operation-level security and role-based authorization EAR file

In this section you create the EAR file that will be used for operation-level security and role-based authorization. The EAR file will contain an EJB module with a single Stateless Session EJB that has bean methods corresponding to operations defined in the WSDL for the inbound service.

1. Copy the file `<was_home>/installableApps/sibwsauthbean.ear` to `<temp_location>`.

2. Create the authorization EAR file for the Web service. From the command line enter the following:

   ```
   cd `<temp_location>`
   `\was_home\util\sibwsAuthGen` `<wsdl_location_url>` `<serviceName>`
   ```

   where:

   ```
   `<wsdl_location_url>` = http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService
   `<serviceName>` = BankAppInboundService
   ```

3. Ensure the following messages shown in Listing 1 are displayed:

   **Listing 1. Message output from creating Authorization EAR file**

   ```
   Retrieving document at
   'http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService'.
   Retrieving document at
   'http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService?wsdl=bindings', relative to
   'http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService'.
   Retrieving document at
   'http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService?wsdl=porttypes', relative to
   'http://localhost:9080/sibws/wsd1/SimpleBankAppSIBus
   /BankAppInboundService?wsdl=bindings'.
   CWSWS0092I: Retrieving Service
   : [http://www.ibm.com/websphere/sib/webservices/tendainoteNode02Cell
   /SimpleBankAppSIBus/Service]BankAppInboundService
   CWSWS0093I: Retrieving Port Type :SOAPHTTPChannel1InboundPort
   CWSWS0094I: Retrieving Methods :getAccountBalance
   CWSWS0094I: Retrieving Methods :updateAccountBalance
   CWSWS0094I: Retrieving Methods :createAccount
   CWSWS0095I: Making Directory
   : ejb\com\ibm\wsgw\beans\security true
   ```
4. Review and delete the contents of the <temp_location>\ejb folder. This is a temporary directory used when creating the EJB JAR file.

This completes the tasks required to create the EAR file that will be used for operation-level security and role-based authorization.

Import authorization EAR files for assembly

In this section you import the role-based and SIBWS authorization EAR files into IBM Rational® Application Developer V6.0 (Application Developer) for assembly and modification of the deployment descriptors. The Application Server Toolkit (AST) could also be used for this purpose.

1. Start up Application Developer and create a temporary workspace.
2. From the J2EE Perspective select File > Import > EAR file and click Next.
3. Click Browse to locate and select <temp_location>BankAppInboundService.ear.
4. Set the Project Name to BankAppInboundService (default) and click Next.
5. Click Next to accept defaults.
6. In the EAR Modules and Utility JAR Projects screen enter
BankAppInboundServiceEJB as the new project name as shown in Figure 12.

Figure 12. BankAppInboundService EAR modules and utility JAR projects

7. Click Finish.

8. From the J2EE Perspective select File > Import > EAR file and click Next.

9. Click Browse to locate and select <temp_location>\sibwsauthbean.ear.

10. Click Next to accept defaults.

11. Click Next to accept defaults and then click Finish.

This completes the tasks required to import the necessary EAR files into Application Developer for assembly and modification of the deployment descriptors.
Enable J2EE role-based authorization

In this section you enable J2EE role-based authorization. The operation-level security is enforced by applying J2EE role-based authorization on the EJB. Authorized calls for the EJB methods are passed on to the target Web service while unauthorized calls are denied.

1. From the J2EE Perspective double click EJB Projects > BankAppInboundServiceEJB > Deployment Descriptor: BankAppInboundService to open the EJB Deployment Descriptor editor.

2. Click on the Assembly tab.

3. In the Security Roles section click Add.

4. Enter barep as the Name and Bank Representative Role as the Description and click Finish.

5. In the Security Roles section click Add.

6. Enter bacustomer as the Name and Bank Customer Role as the Description and click Finish.

7. In the Method Permissions section click Add.

8. Select the check box for barep and click Next.

9. Select the check box for BankAppInboundServiceService EJB and click Next.

10. Using the scroll bar, scroll to the bottom of the method list and select the star (*) for all methods as shown in Figure 13.

Figure 13. Bank representative method permissions
11. Click **Finish**.

12. In the **Method Permissions** section click **Add**.

13. Select the check box for **bacustomer** and click **Next**.

14. Select the check box for **BankAppInboundServiceService** EJB and click **Next**.

15. Select the check box with the star (*) for all Home methods elements.

16. Select the check boxes for all Remote method elements except the createAccount() and updateAccount() methods as shown in Figure 14. **Figure 14. Bank customer method permissions**
17. Click **Finish**.

18. Save changes to the EJB deployment descriptor and close the editor.

This completes the tasks required to enable J2EE role-based authorization.

**Complete assembly of the role-based authorization EAR file**

In this section you complete the assembly of the role-based authorization EAR file by adding the updated BankAppInboundServiceEJB module to the authorization EAR file.

1. From the J2EE Perspective double click Enterprise Applications > sibwsauthbean > Deployment Descriptor: SIBWebservices.
2. Click on the Module tab.
3. In the Modules section click Add.
4. Select the BankAppInboundServiceEJB module and click Finish.
5. Click on the Security tab.
6. Click Gather to pull in all roles defined in the EJB deployment descriptors.
7. Save changes to the application deployment descriptor and close the editor.

Now add an EJB reference for the BankAppInboundServiceService EJB in the authorization EJB module.

9. Click on the References tab.
10. Select the authorization bean and click Add.
11. Select EJB Reference and click Next.
12. Select the Enterprise beans in the workspace radio button.
13. Expand and select SIBWebservices > BankAppInboundServiceEJB > BankAppInboundServiceService as the EJB and click Next.
14. Click Finish. (Ignore the classpath errors for the authorization EJB project. These classes will be in the classpath once deployment to the Application Server is complete.)
15. Save the EJB deployment descriptor changes and close the editor.
16. From the menu bar select File > Export > EAR file and click Next.
17. Select sibwsauthbean from the EAR Project pull down and browse to select destination of your choice.
   NOTE: You may choose to change the name of the EAR file to a more descriptive one, for example, BankAppInboundServiceAuth.ear.
18. Click Finish.

This completes the assembly of the role-based authorization EAR file that will be used to enforce operation-level security for the target Web service.
Enable operation-level security

In this section you enable operation-level security for the inbound service

1. From the Administrative Console navigation pane, expand Service integration and click Buses.

2. Click SimpleBankAppSIBus.

3. In the Services section click Inbound Services and then BankAppInboundService.

4. Select the Enable operation level security check box as shown in Figure 15 and click OK.

Figure 15. Inbound service - enable operation-level security

5. Save configuration changes.

This completes the tasks required to enable operation-level security on the inbound service.

Deploy authorization EAR file
In this section you deploy the EAR file that was created in the previous sections.

1. From the Administrative Console navigation pane, expand **Applications** and click **Install New Application**.

2. Click **Browse** and select the authorization EAR file exported from Application Developer in the previous section. Click **Next**.

3. Click **Next** to accept defaults and then **Continue** to accept defaults.

4. Ensure the **Deploy enterprise beans** option is checked and click **Next**.

5. Click **Next** on steps 2-5 to accept defaults.

6. Select **barep** and click **Look up Groups**.

7. Click **Search** to pull in all groups from the registry.

8. Select **bareps** and then click **>>**. The resulting selection is shown in **Figure 16**.

**Figure 16. Bank representative group mappings**

9. Click **OK**.
10. Repeat steps 6 to 8 as above for the **bacustomer** role, this time selecting the **bacustomers** group rather than the **bareps** group and click **OK**. The resulting role and group mappings are shown in Figure 17.

**Figure 17. Roles to groups mappings**

![Roles to groups mappings](image)

11. Click **Next** to continue on to Step 7.

12. Click **Step 8** and then **Finish** to deploy the application.

13. Verify that there are no error messages from the install.

14. Click **Save Master configuration** and save configuration changes.

15. Start the Application.
   - Expand **Applications** and click **Enterprise Applications**.
   - Select the **SIBWebservices** and click **Start**.
   - Check the server logs to ensure the application started without errors.

16. From the Administrative Console navigation pane, expand **Servers** and click **Application servers**.

17. Click **server1**.

18. In the **Classloader policy** pull down select **Single** and click **OK**. NOTE: Setting the Server Classloader policy to single results in a single classloader for all applications deployed on the server. With operation-level security enabled, the SIBWS application (**sibws.-your_nodename.-server1**) receives the incoming requests and makes a remote EJB call to the authorization EJB. Thus, the RMI classes required for client invocation of the authorization EJB need to be visible to the SIBWS application. These classes are packaged in the authorization EAR file. With a server classloader policy setting of **Multiple**, each
application is loaded in a separate classloader and not visible to other applications on the same server. Thus, the need for Single as the Server Classloader policy.

This completes the tasks required to deploy the authorization EAR file required to support operation-level security through role-based authorization.

19. Verify if configuration changes were successful.

   • Log out from the console and close the Admin console browser.
   • Stop the server. From the command line enter:
     `<wasv60_home>/profiles/SimpleBankAppServer/bin/stopServer.bat | sh server1 -username admin -password testadmin`
   • Restart the Server. From the command line enter:
     `<wasv60_home>/profiles/SimpleBankAppServer/bin/startServer.bat | sh server1`
   • Check the server startup messages for errors.

If the server starts with errors they need to be resolved prior to continuing. Check the server logs for more details and verify your configuration is according to the steps outlined above.

   `<wasv60_home>/profiles/SimpleBankAppServer/logs/server1/SystemOut.log`
   `<wasv60_home>/profiles/SimpleBankAppServer/logs/server1/SystemErr.log`

If the cause of the failure is not obvious from the information in the logs, check the WebSphere documentation for more information on troubleshooting failed server startups.

If the status of `<your_nodename>.server1-SimpleBankAppSIBus` is started without errors, then you have successfully configured the WebSphere Application Server V6.0 for SIB and SIBWS security features for this tutorial.

Test the Web service client authentication and authorization

In this section you run the J2EE application client, BankAppJMSHTTPTestClient.java, packaged in BankAppWSClientSec.ear, to test the deployed solution.

Test client overview:
• Developed using Application Developer after generating the necessary proxy classes from the corresponding WSDL files described below:

**SOAP/JMS (deployed target service):** AccountSessionFacade.wsdl, the WSDL file corresponding to the target Bank Application Web services implementation.

**SOAP/HTTP (configured inbound service - endpoint listener and inbound port):**

• BankAppInboundServiceService.wsdl
• BankAppInboundServicePortTypes.wsdl
• BankAppInboundServiceBindings.wsdl

The steps necessary to export the WSDL files from the Administrative Console are listed below:

• Expand **Service Integration** and select **Buses**.
• Select **SimpleBankAppSIBus**.
• Under **Additional Properties** click **Inbound Services**.
• Click **BankAppInboundService** and then **Publish WSDL files to zip file**.
• Click **BankAppInboundService.zip** to download WSDL.

• WS-Security configuration performed using Application Developer to modify the WS-Security binding and configuration deployment information in ibm-webservicesclient-bnd.xmi and ibm-webservicesclient-bnd.xmi.

• Accepts either of the following command-line arguments:

  JMS: Executes the method BankAppJMSHTTPTestClient.testJMS().

  HTTP [ROLE]: Executes the method BankAppJMSHTTPTestClient.testHTTP(String role)

  BOTH [ROLE]: Executes both of the above methods.

Where [ROLE] = REP or CUST. This argument is used to represent the intended role for SOAP/HTTP requests and is optional. The default value when not entered is REP for Bank Representative and results in an update account operation. Entering CUST results in a retrieve account operation only.

Source code for **BankAppJMSHTTPTestClient.java** is available in the **Download** section.

To run the provided **BankAppJMSHTTPTestClient** code, follow the steps below:
1. Download the BankAppWSClientSec.ear file provided with this tutorial from the Download section.

2. Run the client to verify SOAP/JMS or SOAP/HTTP access to the Bank Application Web services implementation.
   - To test SOAP/JMS enter the following at the command line. Enter an appropriate user ID and password.
     ```
     <was_home>\bin\launchClient <download_location>\BankAppWSClientSec.ear JMS
     ```
   - To test SOAP/HTTP enter the following at the command line. Enter the appropriate user ID and password for each role you would like to test.
     ```
     <was_home>\bin\launchClient <download_location>\BankAppWSClientSec.ear HTTP [ROLE]
     ```
   - To test both SOAP/JMS and SOAP/HTTP enter the following at the command line. Enter the appropriate user IDs and passwords for each role you would like to test.
     ```
     <was_home>\bin\launchClient <download_location>\BankAppWSClientSec.ear BOTH [ROLE]
     ```

Sample output from a successful run using the BOTH REP command line option and signing it as user barep and password testrep is shown in Listing 2.

**Listing 2. BankAppJMSHTTPTestClient output bank representative authorized**

```
C:\...\<was_home>\launchClient BankAppWSClientSec.ear BOTH REP
IBM WebSphere Application Server, Release 6.0
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class
com.ibm.simple.bankapp.j2ee.client.BankAppJMSHTTPTestClient

Attempting to invoke Bank App using SOAP/JMS Endpoint
JMS Endpoint:: jms:/queue?destination=jms/SimpleBankAppQueue&connectionFactory=
jms/SimpleBankAppIncomingQCF&
targetService=AccountSessionFacadeJMS&jndiProviderURL=corbaloc:iiop:localhost:2809/NameServiceServerRootBinding Stub::
com.ibm.simple.bankapp.ejb.AccountSessionFacadeJMSSoapBindingStub
Invoking createAccount
Account created new account number is 5
```
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class
com.ibm.simple.bankapp.j2ee.client.BankAppJMSHTTPTestClient

Attempting to invoke Bank App using SOAP/HTTP Endpoint
HTTP Endpoint::
http://localhost:9080/wsgwsoaphttp1/soaphttpengine
/SimpleBankAppSIBus/BankAppInboundService/SOAPHTTPChannel1InboundPort
Binding Stub :: com.ibm.www.SOAPHTTPChannel1InboundPortBindingStub
Invoking updateAccount where acctnum = 4
Account Updated:: New Account Balance = 600.0

Invoking getAccountBalance where acctnum = 4
New Balance = 600.0

Sample output from a successful run with authorized access using the HTTP CUST command line option and signing it as user bacustomer and password testcustomer1 is shown in Listing 3.

Listing 3. BankAppJMSHTTPTestClient output bank customer authorized

C:\...\<was_home>\launchClient BankAppWSClientSec.ear HTTP CUST
IBM WebSphere Application Server, Release 6.0
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class
com.ibm.simple.bankapp.j2ee.client.BankAppJMSHTTPTestClient
Trying to use context
Done

Attempting to invoke Bank App using SOAP/HTTP Endpoint
as Bank Customer
HTTP Endpoint::
http://localhost:9080/wsgwsoaphttp1/soaphttpengine/SimpleBankAppSIBus
/BankAppInboundService/SOAPHTTPChannel1InboundPort
Binding Stub :: com.ibm.www.SOAPHTTPChannel1InboundPortBindingStub
Invoking getAccountBalance where acctnum = 4
Account Balance = 1100.0

Sample output from a successful run with denied access using the HTTP REP command line option and signing it as user bacustomer and password testcustomer1 is shown in Listing 4.

Listing 4. BankAppJMSHTTPTestClient output bank customer not authorized

C:\...\<was_home>\launchClient BankAppWSClientSec.ear HTTP REP
IBM WebSphere Application Server, Release 6.0
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class
com.ibm.simple.bankapp.j2ee.client.BankAppJMSHTTPTestClient
Trying to use context
Attempting to invoke Bank App using SOAP/HTTP Endpoint as Bank Rep

HTTP Endpoint:
http://localhost:9080/wsgwsoaphttp1/soaphttpengine
/SimpleBankAppSIBus/BankAppInboundService/SOAPHTTPChannel1InboundPort

Binding Stub :: com.ibm.www.SOAPHTTPChannel1InboundPortBindingStub

Invoking getAccountBalance where acctnum = 4
Current Account Balance = 1100.0

Invoking updateAccount where acctnum = 4

WebServicesFault
faultCode: {http://www.ibm.com/sib}sibws
faultString: OperationLevelSecurityCheck
faultActor: null
faultDetail:

OperationLevelSecurityCheck
at
com.ibm.ws.webservices.engine.xmlsoap.builders.WebServicesFaultProcessor.createFault
(SoapFault.java:411)
at
com.ibm.ws.webservices.engine.xmlsoap.SOAPFault.getFault
(SoapFault.java:486)
at
com.ibm.ws.webservices.engine.SOAPPart.getFault
(SoapPart.java:1090)
at
com.ibm.ws.webservices.engine.SOAPPart.getFault
(SoapPart.java:747)
at
com.ibm.ws.webservices.engine.Message.getFault
(Message.java:883)
at
com.ibm.ws.webservices.engine.Message.ifFaultThrowSelf
(Message.java:863)
at
com.ibm.ws.webservices.engine.PivotHandlerWrapper.invoke
(PivotHandlerWrapper.java:278)
at
com.ibm.ws.webservices.engine.handlers.jaxrpc.JAXRPCHandler.invoke
(JAXRPCHandler.java:151)

In the server log files the following exception is logged when operation-level authorization is denied for a customer. See <wasv60_home>/profiles/SimpleBankAppServer/logs/server1/SystemOut.log.

Listing 5. Server log exception

SECJ0053E: Authorization failed for customRealm/bacustomer1 while invoking (Bean)websphere/WSGW/Security/BankAppInboundService/updateAccountBalance:1 securityName: customRealm/bacustomer1;accessID: user:customRealm/222 is not granted any of the required roles: barep

[9/21/05 16:52:24:674 EDT] 00000051 OperationLevelE CWSWS1040E: The following exception occurred while performing operation level security check on method updateAccountBalance.
java.lang.reflect.InvocationTargetException
at sun.reflect.NativeMethodAccessorImpl.invoke0 (Compiled Code)
at sun.reflect.NativeMethodAccessorImpl.invoke (Compiled Code)
at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java (Compiled Code))
at java.lang.reflect.Method.invoke (Compiled Code)
You have successfully tested your Bank Application Web services implementation and the ability to invoke the Web service using SOAP/JMS and secured SOAP/HTTP through an ESB.

---

**Section 6. Troubleshooting**

**Problem 1**

During deployment of the application on Windows you receive the following error:

**Listing 6. Problem 1 error**

```
java.io.IOException: URI length is greater than Windows limit of 259 characters.
```

**Solution:**

Change the location used by WebSphere Application Server for temp files during application deployment.

- From the Administrative Console navigation pane, expand **Servers** and click **Application Servers**.
- Click **server1**.
- Under **Server Infrastructure**, expand **Java and Process Management** and click **Process Definition**.
- Under **Additional Properties** click **Java Virtual Machine**.
- Enter `-Dworkspace.user.root=<new_location>` in the Generic JVM arguments field.
- Save Server Configuration.
Problem 2

When attempting to run the J2EE application test client you receive the following errors:

Listing 7. Problem 2 error

```java
C:\...\>\launchClient ... \BankAppWSClientSec.ear ...
IBM WebSphere Application Server, Release 6.0
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class
com.ibm.simple.bankapp.j2ee.client.BankAppJMSHTTPTestClient

Attempting to invoke Bank App using SOAP/JMS Endpoint
JMS Endpoint::
jms://queue?destination=jms/SimpleBankAppQueue&connectionFactory=
jms/SimpleBankAppIncomingQCF&targetService=
AccountSessionFacadeJMS&jndiProviderURL=
corbaloc:iiop:localhost:2809/NameServiceServerRoot
Binding Stub::
com.ibm.simple.bankapp.ejb.AccountSessionFacadeJMSSoapBindingStub

Invoking createAccount
[7/11/05 15:28:26:240 EDT] 0000000d E UOW=3-7c64e7aa-5298439:blueberry
source=com.ibm.ws.webservices.engine.transport.jms.JMSSender org=IBM
prod=WebSphere component=Application Server thread=[P=97377:O=0:CT]
WSWS3016E: Caught JMSException: javax.jms.JMSException: CWSIA0241E:
An exception was received during the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E:
It is not possible to contact a messaging engine in bus
SimpleBankAppSIBus..
WSWS3017E: Linked Exception:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It
is not possible to contact a messaging engine in bus SimpleBankAppSIBus.
WebServicesFault
faultCode: JMS
faultString: WSWS3016E: Caught JMSException: javax.jms.JMSException:
CWSIA0241E: An exception was received during the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException:
... ...
```

Solution:

Ensure that the SIB ENDPOINT_ADDRESS for the server is set to the default 7276:

- From the Administrative Console navigation pane, expand **Servers** and click **Application Servers**.
- Click **server1**
• Under **Communications** expand **Ports**.
• Verify port is set to 7276.

If the value is not set to 7276, there are two options to resolve this problem.

1. Change the SIB_ENDPOINT_ADDRESS port to the default of 7276.
   • Under **Communications**, click **Ports**.
   • Click **SIB_ENPOINT_ADDRESS** and enter the default value of 7276 for the **Port**.
   • Save configuration changes.
   • Restart server.

2. Add a provider endpoint for the JMS queue connection factory **SimpleBankAppIncomingQCF**.
   • Expand **Resources > JMS Providers** and click **Default Messaging**.
   • In the Default Messaging Provider frame under **Connection Factories**, click **JMS queue connection factory**.
   • Click **SimpleBankAppIncomingQCF**.
   • In the Provider endpoints section enter `<your_host>:<SIB_ENDPOINT_ADDRESS>:BootstrapBasicMessaging`.
   Example: "localhost:7277:BootstrapBasicMessaging". You must make sure that this entry is correct.

   **NOTE:** If you set up the server to use the default SIB_ENDPOINT_ADDRESS of 7276, then you do not have to enter a provider endpoint. In the case where the defaults were not used when creating your profile, the provider endpoint entry is required (and should be as above) in order for the application client to have access to resources in the SIBus.

   See **Configuring connection to a non-default bootstrap server** for more information.

---

**Section 7. Conclusion**

This tutorial demonstrated how to secure communications, in accordance with the Web Services Security (WS-Security) 1.0 specification, between the requestor and the WebSphere Application Server V6.0 SIB. The authors demonstrated how to configure and enable authentication and operation-level authorization of requests between the requestor and inbound service. The overall solution demonstrates
securing Web services through an ESB using WebSphere Application Server V6.0 Service Integration technologies.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
<th>Download method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property files, EAR file, and Java code</td>
<td>ws-seccommcode.zip</td>
<td>47 KB</td>
<td>HTTP</td>
</tr>
</tbody>
</table>

Information about download methods
Resources

Learn

- If you are new to the Enterprise Service Bus, the following resources are good starting points:
  - WebSphere Software - The Enterprise Service Bus
  - Patterns: Implementing an SOA using an Enterprise Service Bus

- The following Redbooks are available online:
  - Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6
  - WebSphere Version 6 Web Services Handbook Development and Deployment

- Read previous articles and tutorials by the author:
  - Transform protocols and route messages through an ESB
  - Develop a multi-transport, accessible Web service in Rational Application Developer V6.0
  - Accessing an EJB Web service using SOAP over JMS or SOAP over HTTP with WebSphere Studio

- The following InfoCenters provide comprehensive product documentation:
  - WebSphere Application Server Infocenter - Introduction to Service Integration
  - WebSphere Application Server Infocenter

- Read the Web Services Security (WS-Security) 1.0 specification.

Get products and technologies

- Download trial versions of the products used in this tutorial:
  - WebSphere Application Server
  - Rational Application Developer
  - WebSphere MQ

About the authors

Tendai Chinoda
Tendai Chinoda is an Advisory Software Engineer at the IBM Business Partner
Technical Enablement - WebSphere Competency Center. He provides enablement services to premier ISVs and Business Partners for the IBM WebSphere family of products. Tendai is an IBM Certified for On Demand Business - Solution Designer, an IBM Certified Solution Developer - Web Services Development with WebSphere Studio V5, a Sun Certified Programmer for the Java 2 Platform 1.2, and holds other WebSphere product certifications. He received his Bachelors degree in Electrical Engineering at Tennessee State University, Nashville TN and completed his Masters in Computer Science and Engineering from the Pennsylvania State University, University Park, PA. You can contact him at tendai@us.ibm.com.

Vivek Gadiyar
Vivek Gadiyar is an Architect with IBM Business Partner Technical Enablement. He works with ISVs and System Integrators to enable them on the WebSphere platform. Vivek is an IBM Certified IT Specialist and his current focus in on WebSphere SAP Integration. Previously, Vivek worked on several development teams for WebSphere Portal, WebSphere Application Server, Encina++ (OTS). and Encina.