

Enabling an IMS application as a web service provider with SAML signed assertion using IMS Enterprise Suite Version 2.2 SOAP Gateway (z/OS version)

Contents

Overview	1
Requirements	2
Contents of the sample ZIP file	2
General process	3
Part I. Creating web service artifacts for your IMS application	4
1.1 Pre-generated sample artifacts	5
1.2 Generating the web service artifacts with Rational Developer for System z	5
Part 2. Deploying the generated artifacts	17
2.1 Deploying the XML converter driver to IMS Connect	18
2.2 Deploying the web service artifacts to the SOAP Gateway server	19
Part 3. Setting up and enabling WS-Security for this web service	22
3.1 On the server side	22
3.2 On the client side	24
Part 4. Enabling client authentication over HTTPS communication	28
Part 5. Creating and running the Java client application	33
5.1 Setting the PATH and CLASSPATH variables	34
5.2 Compiling the Java client application	34
5.3 Running the Java application	34
Summary	36
Additional resources	37

Overview

With IBM® IMS™ Enterprise Suite SOAP Gateway, you can enable your IMS application as a web service. Different types of client applications can submit SOAP requests to IMS that drive the business logic of your IMS applications.

You can enable the web services security (WS-Security) feature to ensure that the security credentials of the client application is validated each time a message is submitted.

This sample guides you through the steps required to enable an IMS application as a web service. This guide uses the IMS Phonebook sample application (IVTNO) to demonstrate

how to enable WS-Security SAML 2.0 signed assertion and how to create a client application that sends messages via secure HTTPS communication (client authentication) to the IMS Phonebook web service that is deployed on the SOAP Gateway server.

The Apache Axis2 web services framework supports multiple XML data-binding approaches, such as XMLBeans, JiBX data binding, as well as the custom Axis Data Binding (ADB) approach developed specifically for Axis2. This sample demonstrates how to use the WSDL2Java tool that takes a WSDL document and generates fully annotated Java code from which to implement a service by using the XMLBeans approach.

Requirements

- IMS Enterprise Suite Version 2.2 SOAP Gateway
- IMS Version 11 or Version 12 with integrated IMS Connect
- The IMS Phonebook sample application files (included with this sample)
- IBM® Rational® Developer for System z™ Version 8.0.3.2 or later
 - Optional: If you don't have access to the tool, the generated artifacts are provided for you.
 - Required: The FEK.SFEKLOAD data set for Rational® Developer for System z must be added to the STEPLIB in the IMS Connect startup procedure for the XML converter function to work.
- Apache Ant for compiling your client application from <http://ant.apache.org/bindownload.cgi>. Store the downloaded ant.jar and ant-launcher.jar file in a convenient location.

Note: This sample is tested with V1.8.2.

Contents of the sample ZIP file

The sample ZIP file that you downloaded includes the COBOL copybook for the Phonebook sample application that will be enabled as a web service. The ZIP file also includes the generated files from Rational Developer for System z V8.0.3.2 for your reference in case you do not have access to the required version of Rational Developer for System z Version.

Filename	Description
Phonebook copybook and files that are generated by Rational Developer for System z With WS-Security enabled (with SAML 2.0 Signed Assertion security token) scenario	
IMSPHBK.cpy	IMS Phonebook application copybook
IMSPHBK.wsdl	WSDL file (generated by RDz V8.0.3.2)
IMSPHBK.xml	Correlator XML file (generated by RDz V8.0.3.2)
IMSPHBKD.cbl	XML converters file (generated by RDz V8.0.3.2)
IMSPHBK_migrated.xml	Migrated correlator file for the new correlator schema.
Sample JCL for compiling and linking the XML converter	

IMSPHBKD.jcl	Sample JCL for compiling and linking the IMSPHBKD.cbl XML converter
Sample Java application	
IMSPHBK_Security.java	Sample Java client application
The SAML SignedAssertion/ folder	
saml-provider.jceks	A sample keystore file
client/bindings.xml and policy.xml	Client binding and policy files (different SAML token types will have their own corresponding binding and policy files)
SAML/saml-creation/SAMLIssuerConfig.properties	File containing configuration properties to control how the SAML token is configured
The files 1347061119810/ folder	
Target/xxxxxx.java	Generated Phonebook service stub files Important: The IMSPHBKServiceStub.java file included here is a customized version to demonstrate the customization required.
z/OS shell script files	
wsdl2java_xmlbean.sh	Generate the client proxy code in xmlbean data bindings
antCompile.sh and ant.sh	Compile the source file
setpath.sh and setclasspath.sh	Set the Java PATH and CLASSPATH

General process

The following diagram shows the runtime process flow when the sample is completed. We will run a Java client application, IMSPHBK_security, to access the Phonebook application for information. We will create a stub file that will handle the request from the IMSPHBK_security application by generating it from the Phonebook web service WSDL. The IMSPHBK_security application calls this client stub, which will translate the requests into SOAP messages.

The web service WSDL is, in turn, generated from the IMS Phonebook application by using Rational Developer for System z. The artifact generation process also generates the XML converter to deploy into IMS Connect. This XML converter will handle the conversion between the XML messages and the binary data from the IMS Phonebook application.

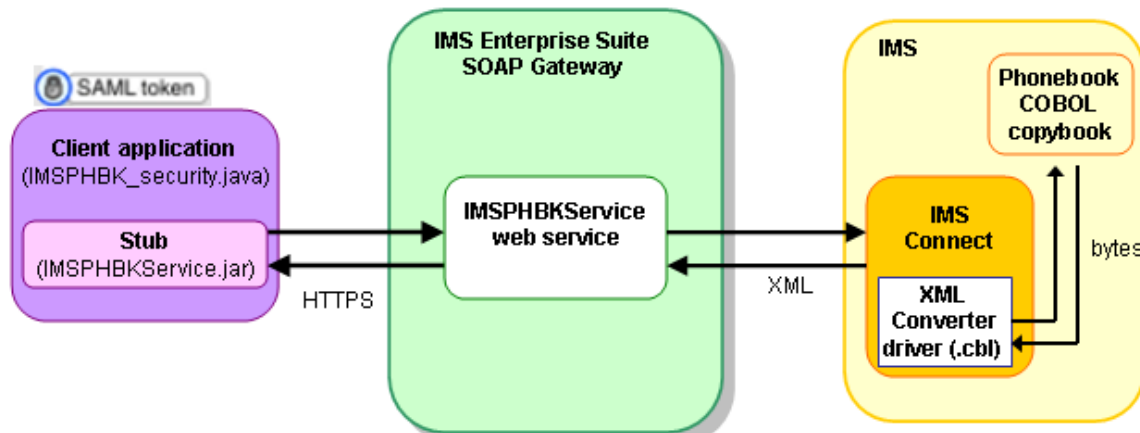


Figure 1. The runtime process for this sample

This sample demonstrates the steps in five parts:

- Part 1. Creating web service artifacts for your IMS application
- Part 2. Deploying the generated artifacts
- Part 3. Setting up and enabling WS-Security for this web service
- Part 4. Enabling client authentication over HTTPS communication
- Part 5. Creating and running the Java client application

Part 1. Creating web service artifacts for your IMS application

To enable an IMS application as a web service with IMS Enterprise Suite SOAP Gateway, you start by creating the following web service artifacts from the application.

1. A XML converter driver that helps convert between XML (the format web services understand) and binary (IMS message format)
2. A web service interface, which is a Web Services Description Language (WSDL) file, that describes where the web service is located, and what the input and output messages look like for invoking your IMS application.
3. A correlator XML file that specifies transaction and runtime properties such as the XML converter driver name, transaction code, and timeout values.

The source to generate these files is the IMS application that describes the input and output messages.

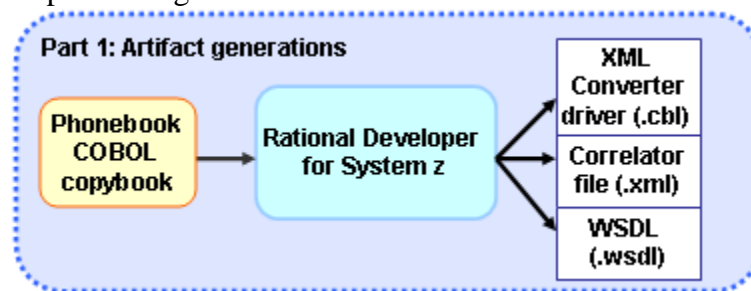


Figure 2. Generating the artifacts using Rational Developer for System z

These generated files need to be deployed to IMS Connect or the SOAP Gateway server before you can deploy a web service.

The following steps will generate these artifacts from the COBOL copybook of the IMS Phonebook sample application by using the Rational Developer for System z.

1.1 Pre-generated sample artifacts

In case you do not have access to Rational Developer for System z, the files that are generated by Rational Developer for System z for this sample application are provided in the ZIP file. You can skip this step and proceed to the next step: deploying the XML converters to IMS Connect.

1.2 Generating the web service artifacts with Rational Developer for System z

Rational Developer for System z provides the XML Services for the Enterprise (XSE) feature that generates the web service artifacts for your IMS application. The Enterprise Service Tools (EST) analyzes the COBOL copybook file that describes the input and output message format for your IMS application and automatically generates the XML converter driver, the web service WSDL file, and the correlator file.

To generate the web service artifacts:

1. Start Rational Developer for System z™ from your desktop by clicking **Start > All Programs > IBM Software Delivery Platform > IBM Rational Developer for System z with java V8.0.x > IBM Rational Developer for System z with java**.

You might be prompted to select a workspace. A workspace is a directory that stores all of the files for the projects. You can select your own directory or use the default directory.

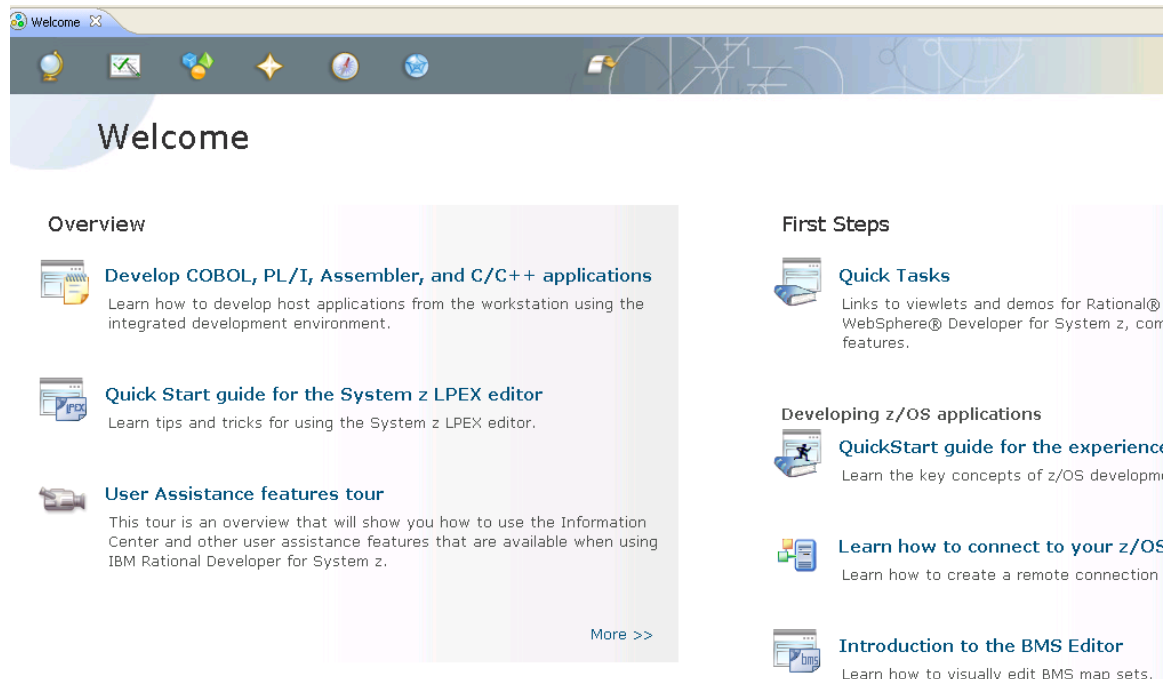


Figure 3. The Welcome panel in Rational Developer for System z

When Rational Developer for System z starts, the Welcome panel displays.

2. Click the **Workbench** icon. The workbench environment displays.
3. From the **Windows** menu, select **Open Perspective > Other**. The Open Perspective window displays.

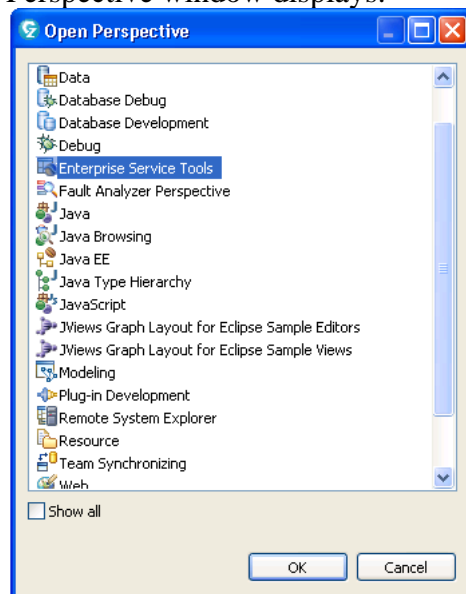


Figure 4. The Open Perspective window

4. Select **Enterprise Service Tools** from the list and click **OK**.

5. Right-click in the EST Project Explorer window and click **File > New > IMS Enterprise Suite SOAP Gateway Project**.

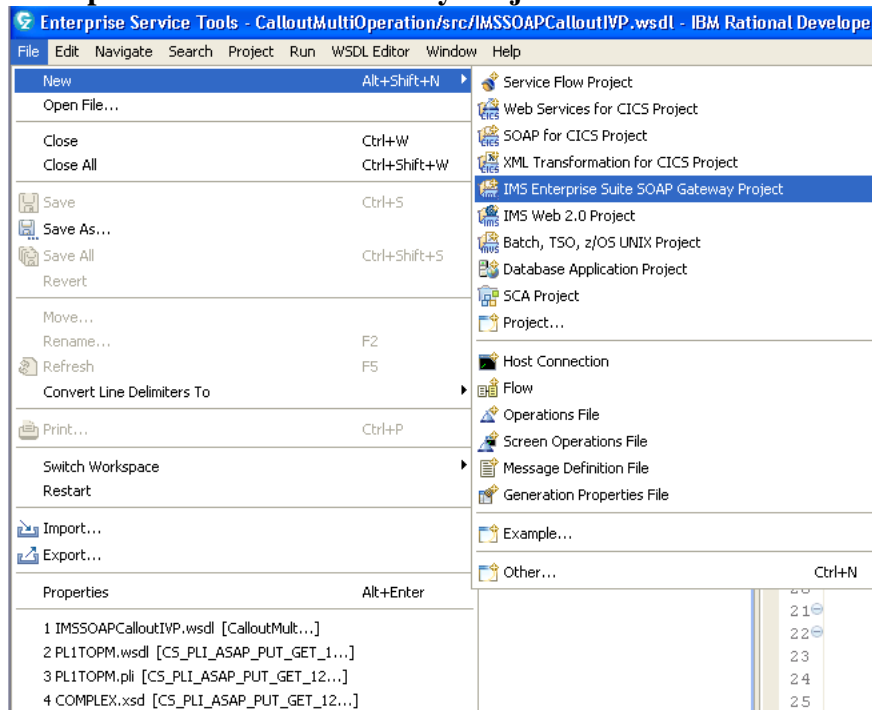


Figure 5. Creating a new IMS Enterprise Suite SOAP Gateway project

6. In the New IMS Enterprise Suite SOAP Gateway Project window, specify the following options:

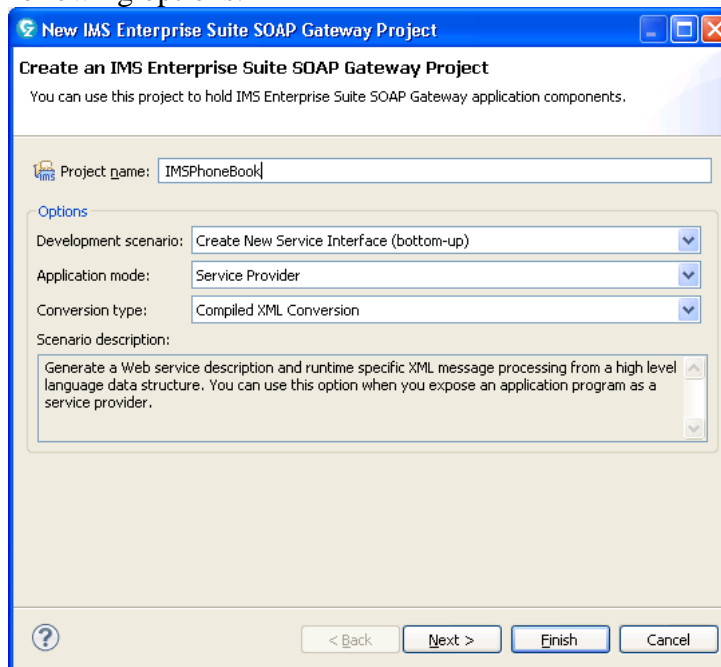


Figure 6. Creating an IMS Enterprise Suite SOAP Gateway project

- a. In the **Project name** field, type the name of your project: IMSPhoneBook.

- b. Select the following options:
 - Development scenario: Create New Service Interface (bottom-up) (default)
 - Application mode: Service Provider (default)
 - Conversion type: Compiled XML Conversion
- c. Click **Next**.

7. Import the source file:

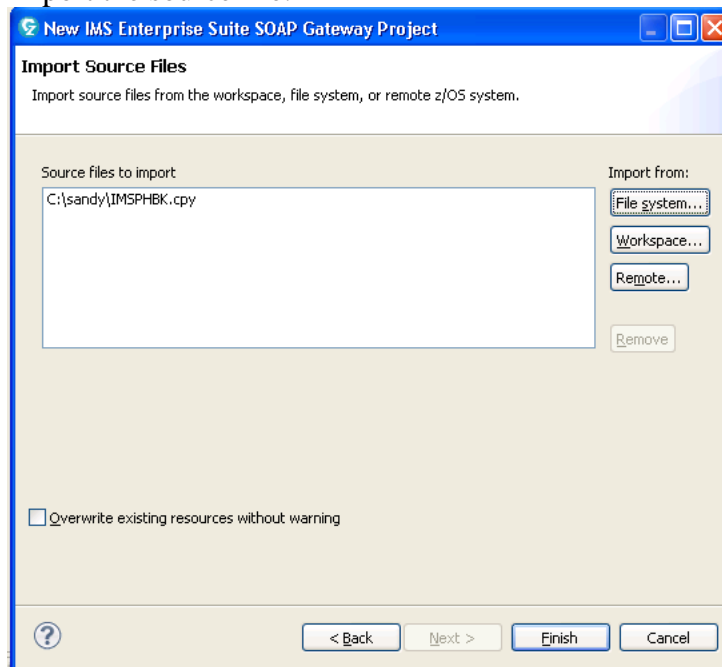


Figure 7. Importing Phonebook copybook source file

- a. Click **Import from File System**.
 - b. In the window that opens, navigate to where the COBOL copybook that describes the format of the input and output messages of your IMS application. In this example, use the **Browse** button to navigate to the IMS Phonebook copybook (IMSPHBK.cpy) and click **Open**.
 - c. Click **Finish**. A new project called **IMSPhoneBook** is now available in your EST Project Explorer, and the Create New Service Interface (bottom-up) wizard opens.
8. In the Create New Service Interface (bottom-up) wizard, specify the request and response language structures and set your COBOL preferences.

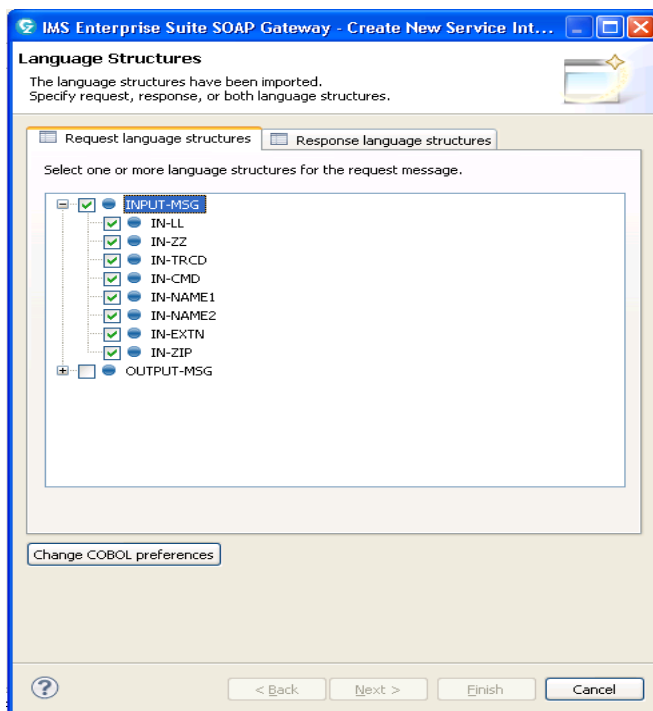


Figure 8. Request data structures selection

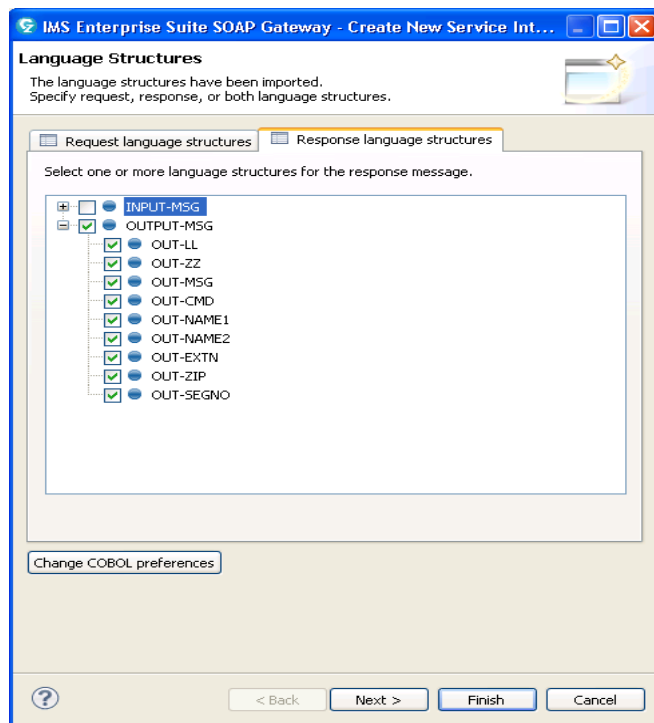


Figure 9. Response data structures selection

- a. In the **Request Language Structure** tab, select the COBOL data structure that corresponds to the input message of the IMS application: INPUT-MSG.

- b. In the **Outbound data structure** tab, select the COBOL data structure that corresponds to the output message of the IMS application: OUTPUT-MSG.
- c. Click **Change COBOL Preferences** and ensure the target platform is set to **z/OS**.

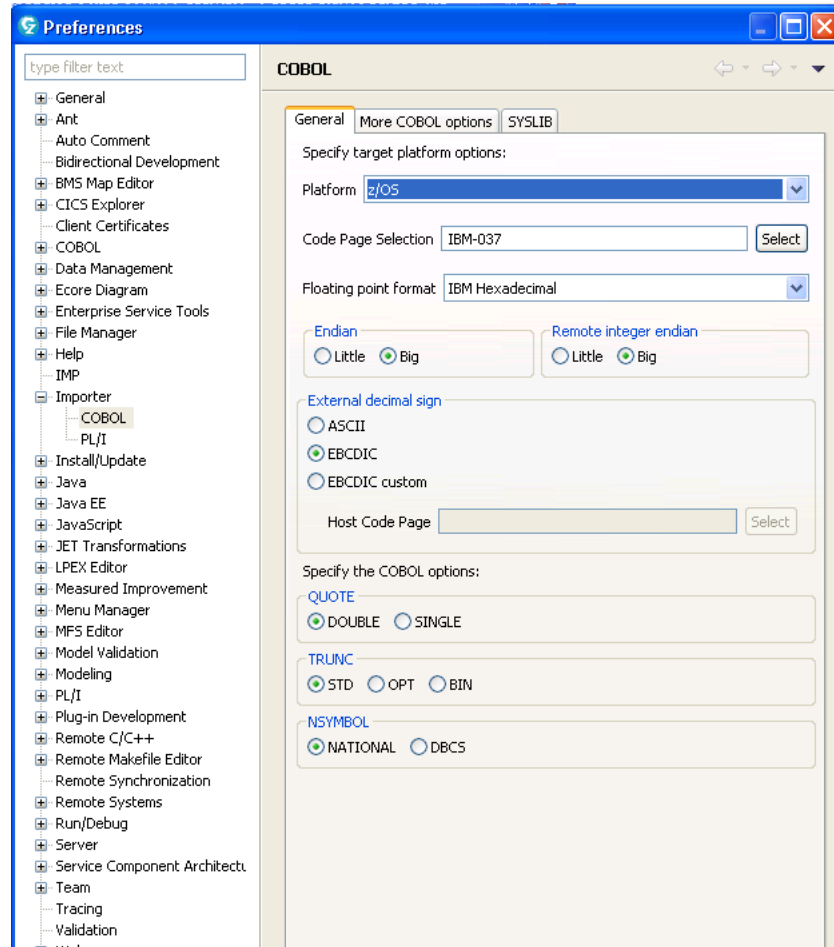


Figure 10. Specify the target platform options to z/OS

- d. Click **OK** to go back to the Create New Service Interface (bottom-up) wizard
 - e. Click **Next**
9. The IMS Message Layouts page is for specifying the minimum and maximum number for the INPUT-MSG that corresponds to the input message of the IMS application, and the minimum and maximum number for the OUTPUT-MSG that

[illegible]

For this sample, we can leave things the way they are. Click **Next**.

10. Specify the generation options:

IMS Enterprise Suite SOAP Gateway - Create New Service Int...

Generation Options

Specify generation options for the Web service enablement artifacts.

XML converters | WSDL and XSD | Advanced options

Specify identification attributes

Converter program name prefix: IMSPHBK

Author name: RD4Z

Service program name: IMSPHBK

Specify Enterprise COBOL for z/OS properties

Compiler level: 4.2

XMLPARSE option: COMPAT Enterprise COBOL for z/OS XML parser

☐ Optimization

Specify character encodings

Request code page: 1208 Unicode, UTF-8

Host code page: 1140 USA, Canada, etc. EBCDIC with Euro

Response code page: 1208 Unicode, UTF-8

? < Back Next > Finish Cancel

Figure 12. Generation Options

- a. In the **Host code page** field, select the code page that the host uses. SOAP Gateway supports only UTF-8 encoding for the inbound and outbound code pages. Therefore, you cannot change these settings.

Note : If the service for LE COBOL PM00230 is not installed, then, change the **XMLPARSE option** to XMLSS and do not forget to specify the RDz load library hlq.SFEKLOAD in the STEPLIB concatenation of the IMS Connect startup procedure.

Generation Options
Specify generation options for the Web service enablement artifacts.

☒ XML converters ☒ WSDL and XSD ☐ Advanced options

Specify WSDL properties

Service location:

Service name:

Operation name:

WSDL namespace:

Specify request XML schema properties

Target namespace:

Root element name:

Whitespace option:

Specify response XML schema properties

Target namespace:

Root element name:

Whitespace option:

Figure 13. WSDL and XSD generation options (The service location of https is for WS-Security enabled web services)

- b. In the **WSDL and XSD** tab, in the **Service location** field, change the hostname and port number to the location of SOAP Gateway. This field specifies the address of the web service. If SOAP Gateway is running on the same machine as your client, you can enter this value:
<https://localhost:8443/imsssoap/services/IMSPHBKService>
(WS-Security is enabled)
- c. Click **Next**.

11. Specify the IMS Enterprise Suite SOAP Gateway correlator properties.

IMS Enterprise Suite SOAP Gateway Web Service Provider

Specify properties for defining the Web service to IMS Enterprise Suite SOAP Gateway. The IMS correlator file is used with the IMS Enterprise Suite SOAP

IMS Enterprise Suite SOAP Gateway correlator file

Specify service identification properties

Generate to: ☒ Same project ☐ Remote location

SOAPAction:

File container:

File name: .xml

☐ Update ☒ Overwrite

Specify IMS Enterprise Suite SOAP Gateway and IMS Connect interaction properties

Transaction code:

Inbound connection bundle:

Socket timeout: (in milliseconds)

Execution timeout: (in milliseconds)

LTERM name:

WS-Security: ▼

Figure 14. Correlator Properties

- In the **Transaction code** field, enter IVTNO.
- In the **Inbound connection bundle** field, enter IMSPHBK.
- In the **WS-Security** field, select **Enabled** to ensure that you have service location set to:

<https://localhost:yourport/imssoap/services/IMSPHBKService>. In this sample, we will set it to 8993 for demonstration purposes.

When you deploy this phone book web service WS-Security enabled, you need to specify the security token type.

Tip: For message-level web services security (WS-Security), you can either use UserNameToken, SAML11Token, SAML20Token, SAML11SignedTokenTrustOne or SAML11SignedTokenTrustAny SAML20SignedTokenTrustOne or SAML20SignedTokenTrustAny sender-voucher tokens. For example:

```
iogmgmt -deploy -w IMSPHBK.wsdl -r IMSPHBK.xml -t SAML20SignedTokenTrustAny
```

See SOAP Gateway management utility reference section of the SOAP Gateway documentation in the information center for details.

Important: WS-Security field is a deprecated field. This value is ignored by SOAP

Gateway. The token type will be defined later when you deploy the web service. Rational Developer for System z V8.0.3.x and V8.5 generates an older version of the correlator schema. IMS Enterprise Suite V2.2 requires a newer version (V2.0) of the correlator schema. This new version is supported in Rational Developer for System z V8.5.1 or later. Therefore, correlator files generated by Rational Developer for System z V8.0.3.x or V8.5 will need to be migrated to the new schema. We will do that later by using the SOAP Gateway management utility `iogmgmt -migrate correlator` command.

- d. Accept the remaining default values and click **Next**.

12. Specify the location and names of the web service artifacts.

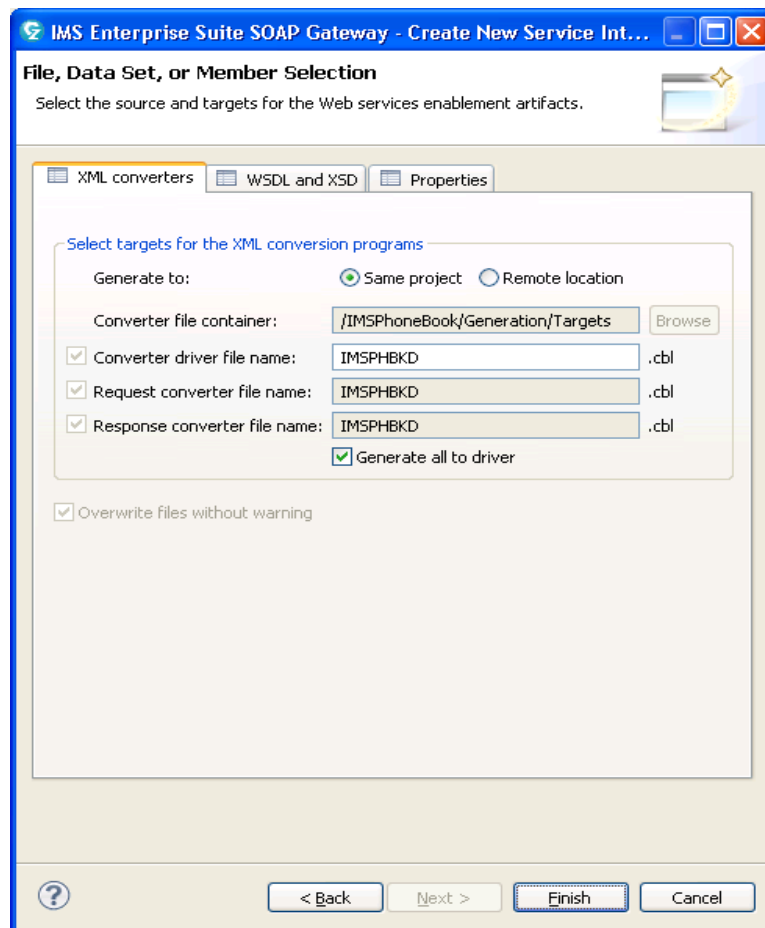


Figure 15. XML Converters tab with Generate all to driver check box selected

- a. Accept the default values for the location and names of the COBOL converters and driver.
- b. Ensure that **Generate all to driver** is selected. This option specifies that the generated web service artifacts (driver, inbound converter, and outbound converter) are all placed in the same file.

- c. In the **WSDL and XSD** tab:

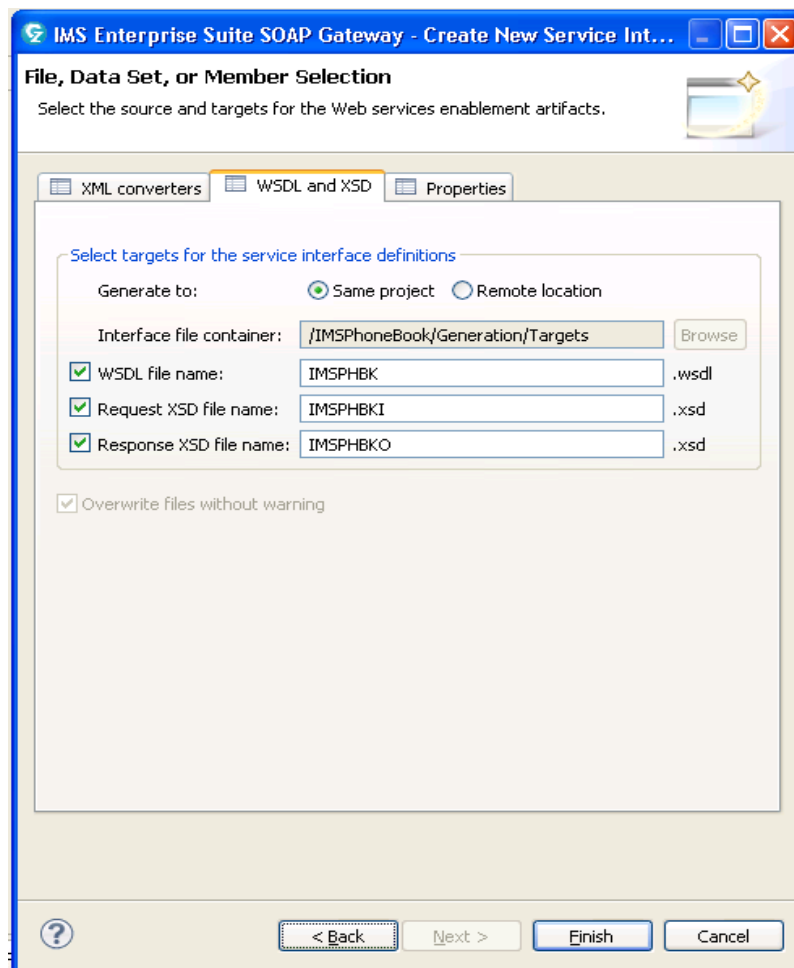


Figure 16. XSD files to be generated

- i. Accept the default location and name for the WSDL file.
- ii. Ensure that the **WSDL file name** check box is selected.
- iii. Optionally, enter names the inbound and outbound XSD files to be generated. These files are not required by SOAP Gateway.
- iv. Click **Finish**.

13. The following files are generated:

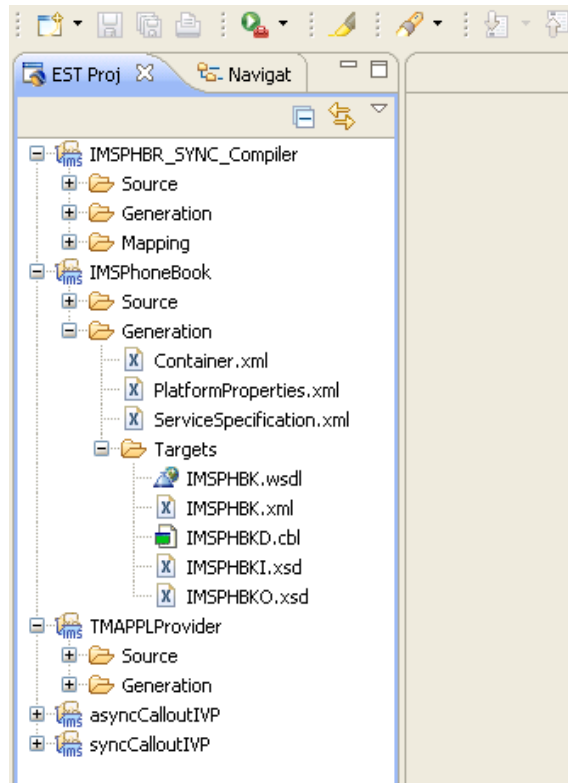


Figure 17. Generated WSDL and Correlator files

- IMSPHBKD.cbl: COBOL converter driver file
- IMSPHBK.xml: correlator XML file
- IMSPHBK.wsdl: WSDL file
- IMSPHBKI.xsd and IMSPHBKO.xsd: Inbound and outbound XSD files (optional; these files are not necessary for SOAP Gateway)

The next step is to deploy the converter driver file to IMS Connect, recycle the IMS Connect instance, and then deploy the IMS Phonebook application web service with the SOAP Gateway management utility.

Part 2. Deploying the generated artifacts

The following diagram demonstrates the where the generated XML converter driver, the correlator file and the WSDL file need to be deployed.

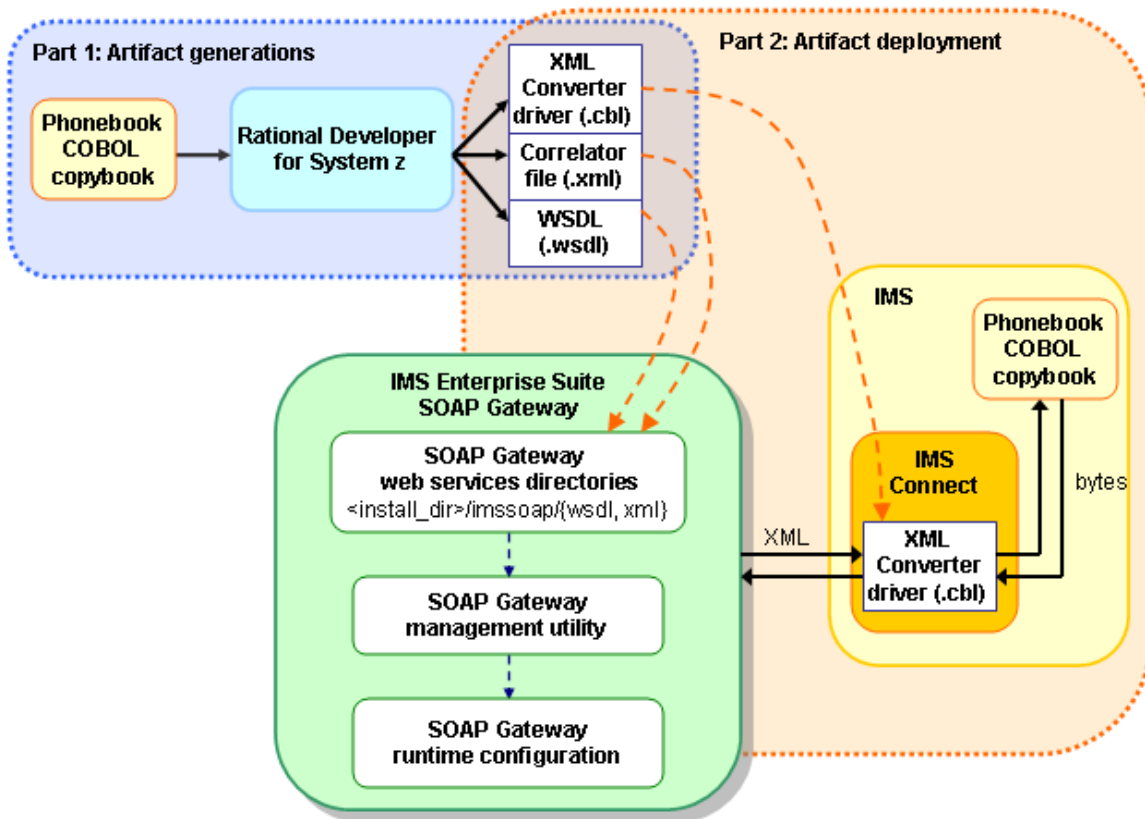


Figure 18. Deploying the generated artifacts

2.1 Deploying the XML converter driver to IMS Connect

To deploy the XML converter driver:

1. Transfer the XML converter driver (IMSPHBKD.cbl) from your workstation to the IMS Connect instance by using FTP. Transfer the file in ASCII mode. Do not use a binary mode FTP client or the COBOL source file may be corrupted. You may have to use this FTP option in some cases.
`quote site sbdataconn=(IBM-037,IBM-1252)`
2. Modify the provided IMSPHBKD.jcl sample JCL job to compile and bind the XML converter. The Rational Developer for System z datasets must be catalogued ahead of time. The highlighted values must be replaced with values specific to your environment. Consult your IMS system programmer for details.

```

//IMSPHBKD JOB LINK,MSGLEVEL=1,REGION=640K,CLASS=G,MSGCLASS=H,
// USER=USRT004,PASSWORD=ALL1SDUN,NOTIFY=&SYSUID
//ORDER JCLLIB ORDER=IGYV4R20.SIGYPROC
//COMPILE EXEC IGYWCL,LNGPRFX=IGYV4R20,PARM.COBOBOL=LIST,
// PARM.LKED='LET,LIST,MAP,AMODE(31)'
//COBOL.SYSLIB DD DSN=CEE.SCEESAMP,DISP=SHR
//COBOL.SYSIN DD DISP=SHR,UNIT=SYSDA,VOL=SER=IMSDQE,
// DSN=SANDY.XMLCNV.SOURCE(IMSPHBKD)
//LKED.SYSLIB DD DSN=TEODORO.RDZ8032.SFEKLOAD,DISP=SHR

```

```
// DD DSN=CEE.SCEELKED,DISP=SHR
//LKED.SYSLMOD DD DSN=IMSTESTL.TNUC0,DISP=SHR
//LKED.SYSIN DD *
ENTRY IMSPHBKD
ALIAS IMSPHBKX
NAME IMSPHBKD(R)
/*
```

3. Add the Rational Developer for System z datasets to your IMS Connect STEPLIB.
4. If you have not done so already, obtain APF authorization to access the Rational Developer for System z SFEKLOAD dataset, for example, with the MVS START command:
S APF,F=ADD,D='TEODORO.RDZ8032.SFEKLOAD',V=column_name
This command adds the data set to the APF authorization list.
5. Restart IMS Connect.

For more information:

- For IMS Version 12, see the “[IMS Connect XML message conversion](#)” topic in *IMS Version 12: Communications and Connections*.
- For IMS Version 11, see the “[IMS Connect XML message conversion](#)” topic in *IMS Version 11: Communications and Connections*.

2.2 Deploying the web service artifacts to the SOAP Gateway server

The following steps show you how to use the SOAP Gateway management utility to deploy your IMS application as a web service to a SOAP Gateway server on the z/OS platform.

1. Store the WSDL file and the correlator XML file in the SOAP Gateway server file system:
 - a. Store the WSDL file (IMSPHBK.wsdl) in the SOAP Gateway server at the same location *installation_directory/imssoap/wsdl*. For example:
/ES22/clone/essg3/imssoap/wsdl/
 - b. Store the correlator file (IMSPHBK.xml) in the IMS Enterprise Suite SOAP Gateway XML directory: *installation_directory/imssoap/xml*. For example: /ES22/clone/essg3/imssoap/xml/
Recommendation: Store the WSDL and XML files in a temporary directory as a backup. When you use the management utility to undeploy this web service, for example, `iogmgmt -undeploy -r myCorrelator.xml`, the correlator file and service files will be deleted from the IMS Enterprise Suite SOAP Gateway XML and services directories respectively and you will have to restore them.
 - c. Migrate the generated correlator file to the new schema required for IMS Enterprise Suite V2.2. This step is needed only if you are using Rational

Developer for System z V8.0.3.x or V8.5. V8.5.1 or later generates the new correlator schema and no migration is needed.

- i. Go to the management utility directory at

`<soap_install_dir>/imsserver/deploy`

For example:

`cd /ES22/clone/essg3/imsserver/deploy`

- ii. Migrate the correlator by using the following command:

`iogmgmt -migrate correlator`

2. Start the IMS Enterprise Suite SOAP Gateway server.

Start the SOAP Gateway server by the job name. The default job name is AEWIOGPR.

`/START AEWIOGPR`

The message "IOG3001I: The SOAP Gateway server is now up and running" appears in the console.

3. Create a connection bundle using the SOAP Gateway management utility.

- a. Change directory to `<soap_install_dir>/imsserver/deploy` if you are not there already. For example:

`cd /ES22/clone/essg3/imsserver/deploy`

- b. Use the Management Utility to create a connection bundle entry named IMSPHBK. A connection bundle is a file that contains connection information for IMS Connect.

Issue the following command:

`iogmgmt -conn -c -n IMSPHBK -d datastore_name -h host_name
-p port_number`

For example:

`iogmgmt -conn -c -n IMSPHBK -d IMS1 -h
csdmec06.vmec.svl.ibm.com -p 9999`

```
$ iogmgmt -conn -c -n IMSPHBK -d IMS1 -h  
csdmec06.vmec.svl.ibm.com -p 9999
```

IOGD0113I: The create connection bundle entry (IMSPHBK) command successfully changed the SOAP Gateway master configuration. The parameters submitted with the command were:

`-conn`

`-c`

`-n IMSPHBK`

`-d IMS1`

`-h csdmec06.vmec.svl.ibm.com`

`-p 9999. The SOAP Gateway server file system was`

updated. The changes will be reflected in the runtime configuration of the server after the next time that the SOAP Gateway starts. No action is required.

In the command shown:

- conn specifies connection bundle tasks;
- c specifies the create task;
- n specifies the connection bundle entry name;
- d specifies the datastore name of the target IMS Connect (case sensitive);
- h specifies the TCP/IP host name of the target IMS Connect;
- p specifies the listening port number of the target IMS Connect.

4. **Optional:** Use the management utility command:

```
iogmgmt -view -cf IMSPHBK.xml
```

to view the correlator XML file to verify the contents:

```
$ iogmgmt -view -cf IMSPHBK.xml
```

```
IOGD0301I: List of correlator entries from correlator file  
(IMSPHBK.xml) in the runtime configuration:
```

```
Correlator Type: Provider  
Service name: IMSPHBKService  
Operation name: IMSPHBKOperation  
XML adapter type: IBM XML Adapter  
Converter name: IMSPHBKD  
Transaction code: IVTNO  
Connection bundle: connbundle2  
Socket timeout: 0  
Execution timeout: 0  
Lterm name:  
WS-Security Type: SAML20SignedTokenTrustAny
```

6. **Optional:** You can use the command:

```
iogmgmt -corr -u -r correlator_name -i service_name -p  
operation_name -s 4000
```

to update or add information to the correlator XML file.

```
$ iogmgmt -corr -u -r IMSPHBK.xml -i IMSPHBKService -p IMSPHBKOperation -s  
4000
```

```
IOGD0503I: The update correlator command successfully updated IMSPHBK.xml  
in the master and runtime configuration. The correlator and parameters  
submitted with the command were:
```

```
-s 4000
```

```
. The SOAP Gateway server file system was updated with the listed  
properties. No action is required.
```

In the command shown, the following parameter specifies the correlator property to update and the new value for the property:

-s specifies the socket timeout value in milliseconds

The steps so far have all required web service artifacts generated, stored, and configured in the appropriate location. The next step is to deploy the Phonebook application and specify the WS-Security SAML token type during the deployment.

Part 3. Setting up and enabling WS-Security for this web service

The steps below describe how to enable WS-Security for this web service. If you do not want to enable WS-Security, skip to the next section, “Part 4. Enabling client authentication over HTTPS communication,” which shows you how to set up for client (mutual) authentication.

To enable web services security, you need to set up the SOAP Gateway server and prepare the client application.

3.1 On the server side

1. Deploy IMSPHBK web service with WS-Security enabled (SOAP Gateway server is still running).

To enable the WS-Security for the web service, you need to ensure that the end point of the WSDL file (IMSPHBK.wsdl) points to the secure port. The default secure port number is 8443. We are using 8993 in this phonebook sample.

- a. Edit the WSDL file (in `<soap_install_dir>/imsssoap/wsdl`). Go to the bottom lines and edit the port number:

```
<wsdl:service name="IMSPHBKService">
  <wsdl:port
    binding="tns:IMSPHBKBinding" name="IMSPHBKPort">
    <soap:address location=
      "https://localhost:8993/imsssoap/services/IMSPHBKService" />
    </wsdl:port>
  </wsdl:service>
```

- b. Use the `iogmgmt -deploy` command to deploy the web service:
`iogmgmt -deploy -w IMSPHBK.wsdl -r IMSPHBK.xml -t SAML20SignedTokenTrustAny`

```
$ iogmgmt -deploy -w IMSPHBK.wsdl -r IMSPHBK.xml -t
SAML20SignedTokenTrustAny
IOGD0104I: The deploy command successfully deployed the IMSPHBKService
web service to the runtime and master configurations: Web ser
vice definition and associated schema XML files:
/ES22/clone/essg3/imsssoap/wsdl/IMSPHBK.wsdl. Correlator XML
file:/ES22/clone/essg3/
imsssoap/xml/IMSPHBK.xml.
```

In the command:

- deploy specifies the task
- w specifies the complete path to the WSDL file
- r specifies the complete path to the correlator XML
- t specifies the WS-Security token type (SAML2SignedTokenTrustAny)

2. If the deployment is successful, go to the next step for client side setup. If unsuccessful, undeploy the IMSPHBK web service (while the SOAP Gateway server is running) and redeploy. As recommended earlier, make sure you have a backup copy of the WSDL file and correlator XML file before you undeploy a web service because these files will be removed from the wsdl and xml directories when a web service is undeployed.

To undeploy, use the following command:

```
$ iogmgmt -undeploy -r IMSPHBK.xml
```

```
IOGD0750I: The undeploy command successfully undeployed the service
with correlator (IMSPHBK.xml and /ES22/clone/essg3/imsssoap/WEB-I
NF/services//IMSPHBKService.aar) from the SOAP Gateway master and
runtime configurations.
```

In the command shown:

- undeploy specifies the task
- r specifies the correlator file name of the web service you want to undeploy

The associated correlator XML and the service file are deleted (see the previous recommendations for storing the XML and WSDL files.)

3. Verify that the deployment has completed successfully. Your IMS application is enabled as a web service. To verify this, do the following:

- a. Open an Internet browser and start the SOAP Gateway Administrative Console entering <http://hostname:port/imsssoap>

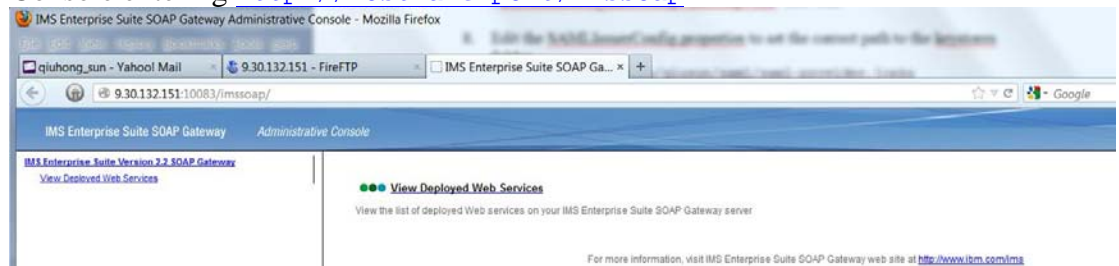


Figure 19. The SOAP Gateway administrative console

- a. Click View Deployed Web Services.

- b. Click **Services** in the right panel. You will see “**IMSPHBKService**” in the list of deployed services.

3.2 On the client side

We need to create a client application that can process the signed SAML tokens. The general steps are:

1. Use `wsdl2java_xmlbean.sh` to generate the proxy Java code. This shell script uses the Apache Axis WSDL2Java utility to generate the client stub code.
2. Update the generated stub file (`IMSPHBKServiceStub.java`) to import the required web services security related classes, set the token type and SAML attributes, and issue the token.
3. Issue the `antCompile.sh build.xml` command to compile the source file.
4. Make sure client side binding and policy files are present.
5. Rename `IMSPHBKService.aar` generated in the output directory (`output/build/lib/`) to `IMSPHBKService.jar`.
6. Edit `SAMLIssuerConfig.properties` to set the correct path to the keystores folder.

The following diagram shows the general steps demonstrated in this part of the task.

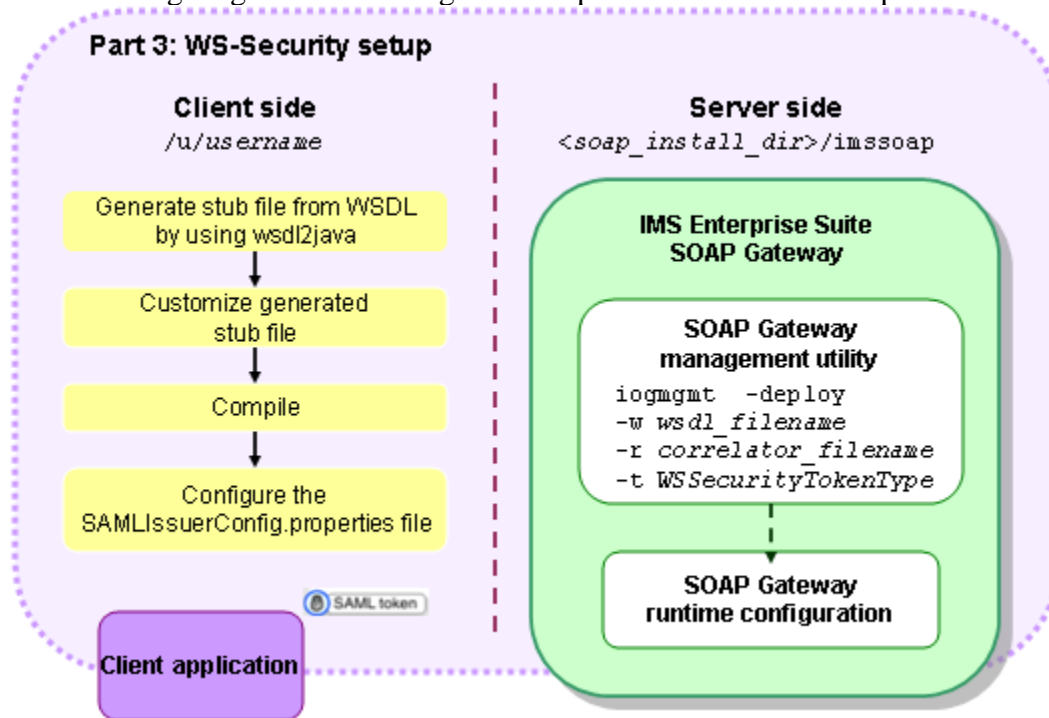


Figure 20. Running `wsdl2java` to generate the client application

To clearly distinguish the directory where you work on your client application from where the SOAP Gateway server runs, we will be using a user directory for all client-side work.

- When you see a path such as `<soap_install_dir>/imsssoap`, the task is for the server side.
- When you see a path such as `/u/username`, the task is for the client side.

We will be creating the client stub, an application generated from a WSDL file for handling the SOAP messages.

Take the following steps to create a Java client stub:

1. Use the **wsdl2java_xmlbean.sh** to generate a proxy java code.
 - a. Create a directory. In this example, we have a user called “qiusun” and we create a directory path as follows:
`/u/qiusun/saml/xmlbean_ES22`
 Use the `mkdir` command to create the directory path.
 - b. Copy the `wsdl2java_xmlbean.sh` file provided with this package into this directory. This file generates a Java client based on the WSDL file by using the XMLbeans approach.
 - c. Copy the `IMSPHBK.wsdl` file into the same directory.
 - d. Go to the directory that stores the `wsdl2java_xmlbean.sh` and `IMSPHBK.wsdl` files.
 - e. Edit `wsdl2java_xmlbean.sh` with the correct value for `IMSSOAP_DIR`.
 - f. Issue the following command:
`wsdl2java_xmlbean.sh IMSPHBK.wsdl output`

```
$ wsdl2java_xmlbean.sh IMSPHBK.wsdl output
Retrieving document at 'IMSPHBK.wsdl'.
log4j:WARN No appenders could be found for logger
(org.apache.axis2.description.WSDL11ToAllAxisServicesBuilder).
log4j:WARN Please initialize the log4j system properly.
(Location of error unknown)Duplicate variable declaration for:
'isUnwrapParameters'
(Location of error unknown)Duplicate variable declaration for:
'operationName'
(Location of error unknown)Duplicate variable declaration for:
'inputcount'
(Location of error unknown)Duplicate variable declaration for:
'inputcount'
```

In the command shown, the first argument should be the complete path to the WSDL file. In this case, we are assuming that the `IMSPHBK.wsdl` file is already copied into the same directory.

The second argument is the generated output directory

Note: You can ignore those warning messages. There is no functional problem, nothing failed, and the messages can be safely ignored.

2. You need to update the generated client stub file (IMSPHBKServiceStub.java) to set SAML context to the message context. This stub file has been generated in the previous step in the following directory:
`/u/qiusun/saml/xmlbean_ES22/output/src/files1347061119810/target/`

Compare the stub file with the one provided with this sample. The provided sample file contains comments that highlight places where changes or additions that are required. Search for

```
//@start SAMLSignedAssertionSupport  
and  
//@end SAMLSignedAssertionSupport  
for the required changes.
```

Note: Due to Axis 2 version update, for the stub file that was generated by Enterprise Suite 2.2, we need to manually call the following line:
`_serviceClient.engageModule("wss");`

Without this line, the SAML Token object will not be generated and inserted to the SOAP envelope.

3. Compile the source file using the antCompile.sh.
 - a. If you have not yet done so, download Ant from <http://ant.apache.org/bindownload.cgi>. Store the downloaded ant.jar and ant-launcher.jar file in a convenient location.
Note: In this example, we suggest that you store them under
`/u/qiusun/saml/xmlbean_ES22/ant/binary`
so you can use the antCompile.sh script without too many changes.
 - b. Modify the antCompile.sh script to set the IMSSOAP_DIR and JAVA_HOME variables based on your environment. If you store them in a different location, modify antCompile.sh accordingly.
 - c. Ensure that the SOAP Gateway installation directory, and the ant.jar and ant-launcher.jar file location are specified in you classpath. A **setclasspath.sh** file is provided with this sample.
 - i. Modify the script for your environment settings.
 - Ensure that the SOAP Gateway installation directory is updated based on your environment.
 - Ensure that the ant.jar and ant-launcher.jar files are pointed to in the classpath.
 - ii. Execute the shell script:
`./setclasspath.sh`
- b. Copy the antCompile.sh and ant.sh file provided with this package into the temporary directory `/u/qiusun/saml/xmlbean_ES22/output`. Use the

ASCII mode if you are using an FTP tool.

c. Issue the command:

```
. ./antCompile.sh build.xml
```

```
$ . ./antCompile.sh build.xml
```

The output looks as follows:

```
on <javac encoding="UTF-8" debug="on" memoryMaximumSize="512m"
memoryInitialSize="512m" fork="true" destdir="${classes}" srcdir
="${src}">
<javac encoding="UTF-8" debug="on" memoryMaximumSize="512m"
memoryInitialSize="512m" fork="true" destdir="${classes}">
build.xml already has proper encoding
./u/qiusun/:/ES22/clone/essg3/imsserver/server/lib/iogaxis/*/ES22/
clone/essg3/imsserver/server/lib/iogwss/*/ES22/clone/essg3/imsser
ver/server/lib/iogsoap/*/ES22/clone/essg3/imssoap/WEB-INF/lib/*/
ES22/clone/essg3/imsserver/deploy/*/javaroot/jdk170/J7.0/lib/ibmc
fw.jar:/javaroot/jdk170/J7.0/lib/ibmjgssprovider.jar:/javaroot/jdk170
/J7.0/lib/ibmpkcs.jar:/javaroot/jdk170/J7.0/lib/ext/ibmjceprovi
der.jar:/javaroot/jdk170/J7.0/lib/ext/ibmpkcs11impl.jar:/ES22/clone/
essg3/imsserver/server/lib/servlet-api.jar:/ES22/clone/essg3/ims
soap/WEB-INF/lib/IMSPHBKService.jar:/ES22/clone/essg3/imssoap/WEB-
INF/lib/XBeans-packaged.jar::/ES22/clone/essg3/imsserver/server/li
b/iogaxis/*/ES22/clone/essg3/imsserver/server/lib/iogwss/*/ES22/
clone/essg3/imsserver/server/lib/iogsoap/*.jar::/ES22/clone/essg3/
imssoap/WEB-INF/lib/*/ES22/clone/essg3/imssoap/imsserver/deploy/*:
Buildfile: /u/qiusun/saml/xmlbean_ES22/output/build.xml
init:
jar.xbeans:
pre.compile.test:
    [echo] XmlBeans Availability = true
    [echo] Stax Availability= true
    [echo] Axis2 Availability= true

compile.src:
    [javac] /u/qiusun/saml/xmlbean_ES22/output/build.xml:49:
        warning: 'includeantruntime' was not set, defaulting to
build.sysclasspath
h=last; set to false for repeatable builds
echo.classpath.problem:
jar.server:
BUILD SUCCESSFUL
Total time: 4 seconds
```

4. Rename the IMSPHBKService.aar to IMSPHBKService.jar. Renaming the file is needed because this file because it is not intended as a web service, but is needed as a JAR file later during the compilation of the client application in Part 5 of the sample.

5. Edit the SAMLIssuerConfig.properties file provided in the SAMLSignedAssertion\SAML\saml-creation directory to set the correct path to the keystores folder. The KeyStorePath value must be edited with your path.

```
IssuerURI=http://www.websphere.ibm.com/SAML/SelfIssuer
KeyStorePath=/u/qiusun/saml/saml-provider.jceks
KeyStoreType=jceks
KeyStorePassword=storepass
KeyAlias=samlissuer
KeyPassword=samlissuer
KeyName=CN=SAMLIssuer,O=IBM,C=US
```

This properties file defines the default keystore, the password to open the keystore, and the private key to sign SAML tokens.

Part 4. Enabling client authentication over HTTPS communication

Secure HTTPS communication is required for web services that use WS-Security with SAML tokens and the SAML 2.0 sender-vouches confirmation method. SAML tokens contain security information in the message header that is not protected unless the message is encrypted with SSL or transport-layer security. You can configure SOAP Gateway to provide this security in addition to WS-Security.

The client must make an HTTPS connection to IMS Enterprise Suite SOAP Gateway server. The client uses a local truststore to verify that the public key from IMS Enterprise Suite SOAP Gateway is trusted and SOAP Gateway verifies the client public key with the server truststore before continuing communication.

The following diagram shows the steps to create keystores and truststores on both the client and the server in order to set up HTTPS communication from the client to the SOAP Gateway server.

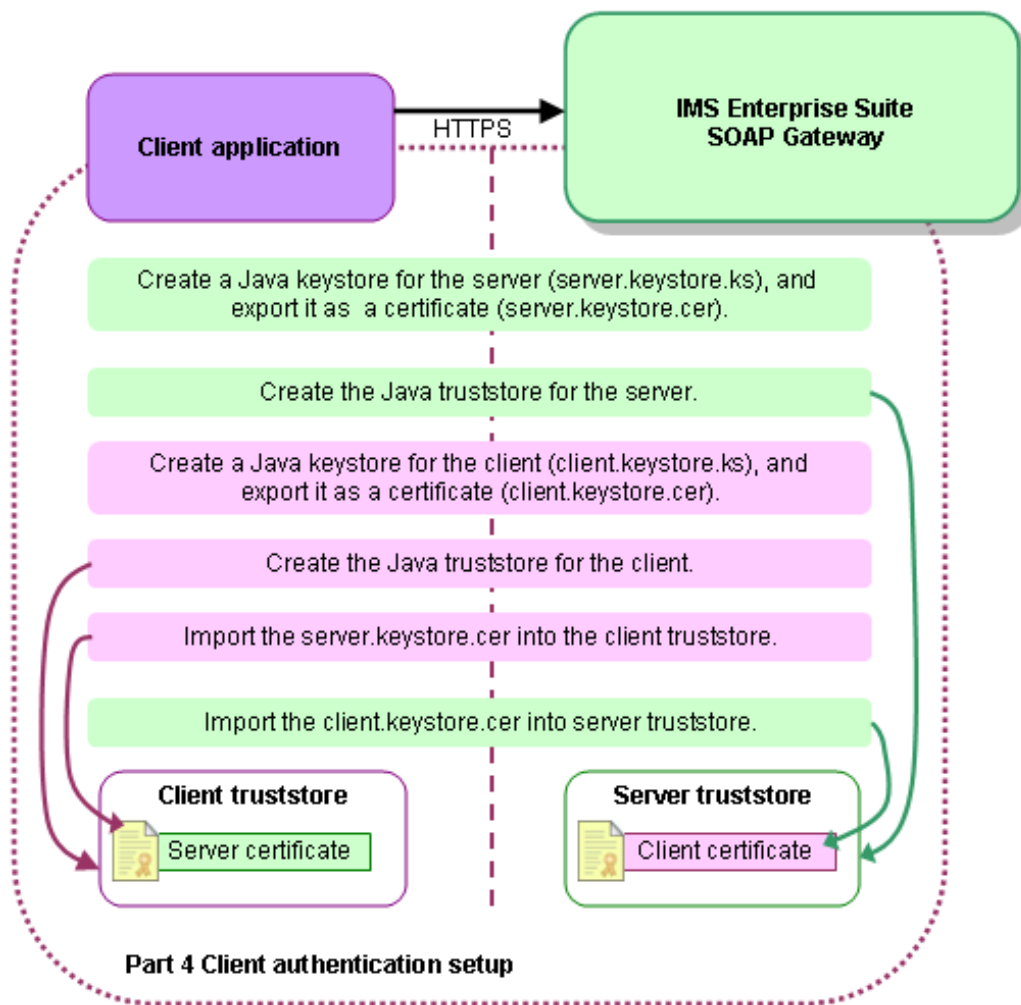


Figure 21. Setting up for client authentication

The following steps demonstrate the commands used with the IBM Java tools to create and configure the Java security stores on the SOAP Gateway server. Replace “/javaroot/jdk170/J7.0” with your java installation path.

```
cd /javaroot/jdk170/J7.0/bin
```

Create a new `ssl` directory. For example:

```
mkdir /u/qiusun/ssl
```

1. Create the Java Keystore for IMS Enterprise Suite SOAP Gateway (server.keystore.js) containing an RSA key pair.

```
keytool -genkey -alias server.keystore -dname "CN=Server
Keystore OU=IBM SWG, O=IBM, C=US" -keyalg RSA -keypass imsssoap
-storepass imsssoap -keystore /u/qiusun/ssl/server.keystore.js
```

2. Export the public key from server.keystore.ks as a certificate (server.keystore.cer)

```
keytool -export -alias server.keystore
-storepass imsssoap -file /u/qiusun/ssl/server.keystore.cer
-keystore /u/qiusun/ssl/server.keystore.ks
```

You can ignore the message JVMJ9VM082E Unable to switch to IFA processor - issue "extattr +a libj9ifa24.so"

3. Create the Java Truststore for IMS Enterprise Suite SOAP Gateway (server.truststore.ks).

```
keytool -genkey -alias server.truststore
-dname "CN=Server Truststore, OU=IBM SWG, O=IBM, C=US"
-keyalg RSA -keypass imsssoap -storepass imsssoap
-keystore /u/qiusun/ssl/server.truststore.ks
```

Directory "/u/qiusun/ssl" should now contain "server.keystore.cer", "server.truststore.ks" and "server.keystore.ks"

4. Create client side keystore (client.keystore.ks) containing an RSA key pair.

```
keytool -genkey -alias client.keystore
-dname "CN=Client Keystore, OU=IBM SWG, O=IBM, C=US"
-keyalg RSA -keypass imsssoap -storepass imsssoap
-keystore /u/qiusun/ssl/client.keystore.ks
```

5. Export the public key from client.keystore.ks as a certificate (client.keystore.cer)

```
keytool -export -alias client.keystore -storepass imsssoap -file
/u/qiusun/ssl/client.keystore.cer -keystore
/u/qiusun/ssl/client.keystore.ks
```

6. Create client side truststore (client.truststore.ks):

```
keytool -genkey -alias client.truststore -dname "CN=Client
Truststore OU=IBM SWG, O=IBM, C=US" -keyalg RSA -keypass imsssoap
-storepass imsssoap -keystore /u/qiusun/ssl/client.truststore.ks
```

7. Transfer the server certificate (server.keystore.cer) to the client side with FTP. Then, import the server.keystore.cer certificate into the client trust store

```
keytool -import -v -trustcacerts -alias server -file
/u/qiusun/ssl/server.keystore.cer -keystore
/u/qiusun/ssl/client.truststore.ks -keypass imsssoap -storepass
imsssoap
```

The output looks as follows:

```

Owner: CN="Server Keystore OU=IBM SWG", O=IBM, C=US
Issuer: CN="Server Keystore OU=IBM SWG", O=IBM, C=US
Serial number: 6a9807a9
Valid from: 11/8/12 2:58 PM until: 2/6/13 2:58 PM
Certificate fingerprints:
    MD5: 6F:24:34:A8:54:1D:07:50:81:14:3D:18:A1:CF:EB:EA
    SHA1:
24:C4:4F:51:05:7C:D3:CD:18:E4:49:EE:15:5C:9C:B6:96:90:73:94
    SHA256:
86:53:E3:BC:C7:F3:BC:52:AE:35:4E:05:E0:5D:E8:9F:08:A4:FA:40:93:B5
:00:18:EB:6C:D1:4F:AA:92:8A:05
    Signature algorithm name: SHA256withRSA
    Version: 3

```

Extensions:

```

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [

```

```

0000: 2b 1a 47 ca 0a 85 df da f9 ec 9e 75 54 57 4f 8c
..G.....uTWO.

```

```

0010: 52 07 74 43                                     R.tC

```

```

]
]

```

```

Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing /u/qiusun/ssl/client.truststore.ks]

```

8. Transfer the client certificate (client.keystore.cer) to the SOAP Gateway server side by using FTP if the server is running on a different system.
9. Import the client.keystore.cer into SOAP Gateway server truststore.

```

keytool -import -v -trustcacerts -alias client
-file /u/qiusun/ssl/client.keystore.cer
-keystore /u/qiusun/ssl/server.truststore.ks -keypass imsssoap
-storepass imsssoap

```

The output looks as follows:

```

$ keytool -import -v -trustcacerts -alias client -file /u/qiusun/ssl/
client.keystore.cer -keystore /u/qiusun/ssl/server.truststore.ks -
keypass imsssoap -storepass imsssoap
Owner: CN=Client Keystore, OU=IBM SWG, O=IBM, C=US
Issuer: CN=Client Keystore, OU=IBM SWG, O=IBM, C=US
Serial number: 60a2853f
Valid from: 11/8/12 3:00 PM until: 2/6/13 3:00 PM
Certificate fingerprints:
    MD5: 4B:5F:F0:5F:44:DE:08:A8:C4:14:D5:D8:53:B9:F5:17
    SHA1: 0B:2A:D5:AA:71:63:EC:45:8C:52:F8:12:15:8D:9E:B1:AB:5D:C9:18
    SHA256:

```

```

A1:C6:D2:38:8E:5D:47:32:95:32:FF:2C:31:DA:25:53:D5:54:9B:B1:6B:C8:34:F6:B2
:
61:1E:EC:E6:86:C1:F0
    Signature algorithm name: SHA256withRSA
    Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 83 cc a8 38 da 8e 54 b3  b7 f6 ae 4f 75 ce 8b 35  ...8..T....Ou..5
0010: 4f c8 6d 1a                                O.m.
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing /u/qiusun/ssl/server.truststore.ks]

```

10. Configure the IMS Enterprise Suite SOAP Gateway for client authentication with the management utility. Use the command

```

iogmgmt -prop -u -clientauth true -s 8993
-k server_keystore -w keystore_password
-t server_truststore -a truststore_password

```

```

$ iogmgmt -prop -u -clientauth true -s 8993 -k
/u/qiusun/ssl/server.keystore.ks \-w imssoap -t
/u/qiusun/ssl/server.truststore.ks -a ims
soap
IOGD0095I: Client Authentication was successfully enabled in the
SOAP Gateway server master configuration. The changes will take
effect the next time that the SOAP Gateway server starts.

```

In the command shown:

-prop -u	specifies the update properties task
-clientauth true	enables client authentication
-s	specifies the secured port number
-k	specifies the fully qualified path to the server side keystore
-w	specifies the password for the keystore
-t	specifies the fully qualified path to the server side truststore
-a	specifies the password for the truststore

11. To disable client authentication, you can issue `iogmgmt -prop -u -clientauth false`

```

$ iogmgmt -prop -u -clientauth false

```


IOGD0085I: The client authentication property value was set to (false). The indicated property was successfully updated. You must restart the SOAP Gateway server for the change to take effect.

Part 5. Creating and running the Java client application

To test the scenario, use a Java client application to invoke the web service. A sample Java application is included to invoke the IMS Phonebook application web service.

The provided IMSPHBK_Security.java sample client application includes the security and SSL-related import statements, the security SAML token that is passed in through the SOAP header, and the SSL keystore and truststore information.

This client application calls the IMS Phonebook web service to obtain the following information:

```
System.out.println("Command: " + output.getOutCmd() );
System.out.println("Last Name: " + output.getOutName1() );
System.out.println("First Name: " + output.getOutName2() );
System.out.println("Extension: " + output.getOutExtn() );
System.out.println("Zip Code: " + output.getOutZip() );
```

The SOAP envelope and message are hardcoded for demonstration purposes.

Important: Before using this sample, you must edit it and replace the hard-coded path to the keystore and truststore in the following statements:

```
System.setProperty("javax.net.ssl.trustStore",
    "/u/qiusun/ssl/client.truststore.ks");
System.setProperty("javax.net.ssl.trustStorePassword","imssoap");
System.setProperty("javax.net.ssl.trustStoreType","JKS");
System.setProperty("javax.net.ssl.keyStore",
    "/u/qiusun/ssl/client.keystore.ks");
System.setProperty("javax.net.ssl.keyStorePassword", "imssoap");
System.setProperty("javax.net.ssl.keyStoreType","JKS");
```

Your SOAP Gateway server address and port must also be updated accordingly:

```
IMSPHBKServiceStub stub = new IMSPHBKServiceStub(null,
    "https://9.30.132.151:8993/imssoap/services/IMSPHBKService");

System.out.println("Message: " + output.getOutMsg() );
```

5.1 Setting the PATH and CLASSPATH variables

- a. Set the Java PATH:

A **setpath.sh** file is provided with this package to set the java path.

```
$ cat setpath.sh
#!/bin/sh
export IMSSOAP_DIR=/ES22/clone/essg3
export JAVA_HOME=/javaroot/jdk170/J7.0
```

Modify the script to your SOAP home and JAVA home directory location, and execute it.

```
. ./setpath.sh
```

- b. Make sure that the Java CLASSPATH is updated to point to where the generated IMSPHBKService.jar and XBeans-packaged.jar (generated in Part 3 of the sample) are stored.

A **setclasspath.sh** file is provided with this package to set the Java classpath. Modify the script to your environment settings and execute it.

```
. ./setclasspath.sh
```

5.2 Compiling the Java client application

Issue this command:

```
javac IMSPHBK_Security.java
```

5.3 Running the Java application

To run the Java application, issue:

```
java IMSPHBK_Security
```

The output looks as follows:

```
$ java IMSPHBK_Security
log4j:WARN No appenders could be found for logger
(org.apache.axis2.description.AxisOperation).
log4j:WARN Please initialize the log4j system properly.
configured client side policy set
SAML=<saml2:Assertion
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
ID="_93986F9A19C044EE6B1352411733983" IssueI
nstant="2012-11-08T21:55:33.982Z"><saml2:Issuer>IBM IMS SOAP
Gateway</saml2:Issuer><ds:Signature
xmlns:ds="http://www.w3.org/2000/09
```

```

/xmlldsig#"><ds:SignedInfo><ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/><ds:SignatureMethod Algor
ithm="http://www.w3.org/2000/09/xmlldsig#rsa-sha1" /><ds:Reference
URI="#_93986F9A19C044EE6B1352411733983"><ds:Transforms><ds:Transfo
rm Algorithm="http://www.w3.org/2000/09/xmlldsig#enveloped-
signature" /><ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c1
4n#" /></ds:Transforms><ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmlldsig#sha1"
/><ds:DigestValue>tt5aOYR8BsWUyUquM13YReD
A4YM=</ds:DigestValue></ds:Reference></ds:SignedInfo><ds:Signature
Value>NvC3EeaC50+cifsB8L61yNoTP1g4CEbix97hgF8tvVKasn6YKZULembhEupC
dw0LIHWqNeyookMIcCbUGiajN0ogVSXybQmtex4/Dp79H4p4kigO1ZaYsaTsVEL0mE
FqP1FpCeRaDaa/meoEjZ1i2ygLBoOPo3exwdgmOxOnjOgw9gzTWBzcsM1Ntpz+A+V9
ryolxugQKAzjlehlRwzWBdVWYjLXXUiLdlM2lRyX0QpWagC9ADWT5dyEjhVYSwPHsh
Gpt6AMxjESh5fo0DMhSRFySugS0H9rXYZWJQ886bgnLMZ72M0+FdO2/mQKFGxiCv+E
BEJX+gQZNMwtS+0aSg==</ds:SignatureValue><ds:KeyInfo><ds:X509Data><
ds:X509Certificate>MIIC/zCCAeegAwIBAgIENfm2PDANBgkqhkiG9w0BAQsFADA
wMQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMRMwEQYDVQQDEwPTQU1MSXNzdWVyM
B4XDTEyMTAxNTA4Mjk0NFoXDTI2MDYyNDA4Mjk0NFowMDELMakGA1UEBhMCVVMxMDELM
KBgNVBAoTA0lCTTETMBEGA1UEAxMKU0FNTelzc3VlcjCCASIwDQYJKoZIhvcNAQEBB
QADggEPADCCAQoCggEBANLqEvrRkohcEo7U2WsYaD5KGcxIx6xfngfWIwRlo1HNo/a
oStpSJ6uXXsM8uzMwCxHyEnFYCYeiac0cB7UdXOXrL+D8/4UJuIPjUn/70LNzVA/NE
0Kgs8cWQJ4npYW7c3hfZKFniloEyGDNR718Ozu9mztDmqVf+MH1zypTsABNT2QgPf7
345EA/Fwogz2vg/h9qGIhF84YZm6aFmaG+zEuV4tzrOsUKzbDp1zmwPfcDxWsSOqK
QfGxQYdtRV3gaG/G+yDH4tQn/2ptgA9ceKh6VVz1NOn8HXgJLfsLr9CxljRMw2wTpT
oBohNUi6lud/nbO4+aZ/7Xbr7JmHYJM8CAwEAAMhMB8wHQYDVR0OBBYEFMc2pMDol
wQjOmsWtW58sK8xo3LMA0GCSqGSIB3DQEBcwUAA4IBAQBrfRznQB3y31LALJv59oU
jilX3lVR3S4WIXOxIkIrOuKucioz9/wHhJ5gvNYWJaDrbk5I/463ZcQIQ2bCiJKQtG
8y6KETrsx21gK/mjwgB0+5d2954wRqEJwUgIdztXkuZNhMljz5k+P+9y8uS4dKqsdo
odOSiePlddglnVnTCP4evndFChiHghXu7cUz2j2IhH0rMAoJFstNNvkvYucVluXGaY
R5rnPGgQoHNh/Plu5azOiL1NMWe6c6aIk4hMH3ByVauV2aXevJuTF5/FDP7PxEOw56
mdO2jjWJnXTgjsJPIkqjfulR1TeyHrHNE3zCGClhx801KI5ci6JjmssyH</ds:X509
Certificate></ds:X509Data></ds:KeyInfo></ds:Signature><saml2:Subje
ct><saml2:NameID>Alice</saml2:NameID><saml2:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches"
/></saml2:Sub
ject><saml2:Conditions NotBefore="2012-11-08T21:55:34.103Z"
NotOnOrAfter="2012-11-08T22:55:34.103Z"
/><saml2:AttributeStatement><sam
l2:Attribute Name="Address" AttributeNamespace="IBM WebSphere
namespace"><saml2:AttributeValue>123 SAML street,
Austin</saml2:Attrib
uteValue></saml2:Attribute><saml2:Attribute
Name="Groups"><saml2:AttributeValue>admin
users</saml2:AttributeValue><saml2:AttributeVa
lue>Building
ABC</saml2:AttributeValue><saml2:AttributeValue>Reporting to
Joe</saml2:AttributeValue></saml2:Attribute></saml2:Attrib
uteStatement></saml2:Assertion>
before execute, envelope = <?xml version='1.0' encoding='utf-
8'?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/en

```

```

velope/"><soapenv:Body><ims:INPUTMSG
xmlns:ims="http://www.IMSPHBKI.com/schemas/IMSPHBKIInterface"><ims
:in_ll>32</ims:in_ll><ims:in_
zz>0</ims:in_zz><ims:in_trcd>IVTNO</ims:in_trcd><ims:in_cmd>displa
y</ims:in_cmd><ims:in_name1>LAST1</ims:in_name1></ims:INPUTMSG></s
oapenv:Body></soapenv:Envelope>
  after execute, envelope = <?xml version='1.0' encoding='utf-
8'?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/env
elope/"><soapenv:Header><s:Security xmlns:s="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns
s:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
soapenv:mustUnderstand="1"><u:Timestamp><u:
Created>2012-11-
08T21:55:37.956Z</u:Created></u:Timestamp></s:Security></soapenv:H
eader><soapenv:Body><OUTPUTMSG xmlns="http://www.I
MSPHBKO.com/schemas/IMSPHBKOInterface"><out_ll>93</out_ll><out_zz>
768</out_zz><out_msg>ENTRY WAS DISPLAYED</out_msg><out_cmd>DISPLAY
</out_cmd><out_name1>LAST1</out_name1><out_name2>FIRST1</out_name2
><out_extn>8-111-1111</out_extn><out_zip>33333</out_zip><out_segno
>0001</out_segno></OUTPUTMSG></soapenv:Body></soapenv:Envelope>
Command: DISPLAY
Last Name: LAST1
First Name: FIRST1
Extension: 8-111-1111
Zip Code: 33333
Message: ENTRY WAS DISPLAYED

```

The last six lines are the response to the request.

Summary

The following diagram shows the overall task flow demonstrated in this sample.

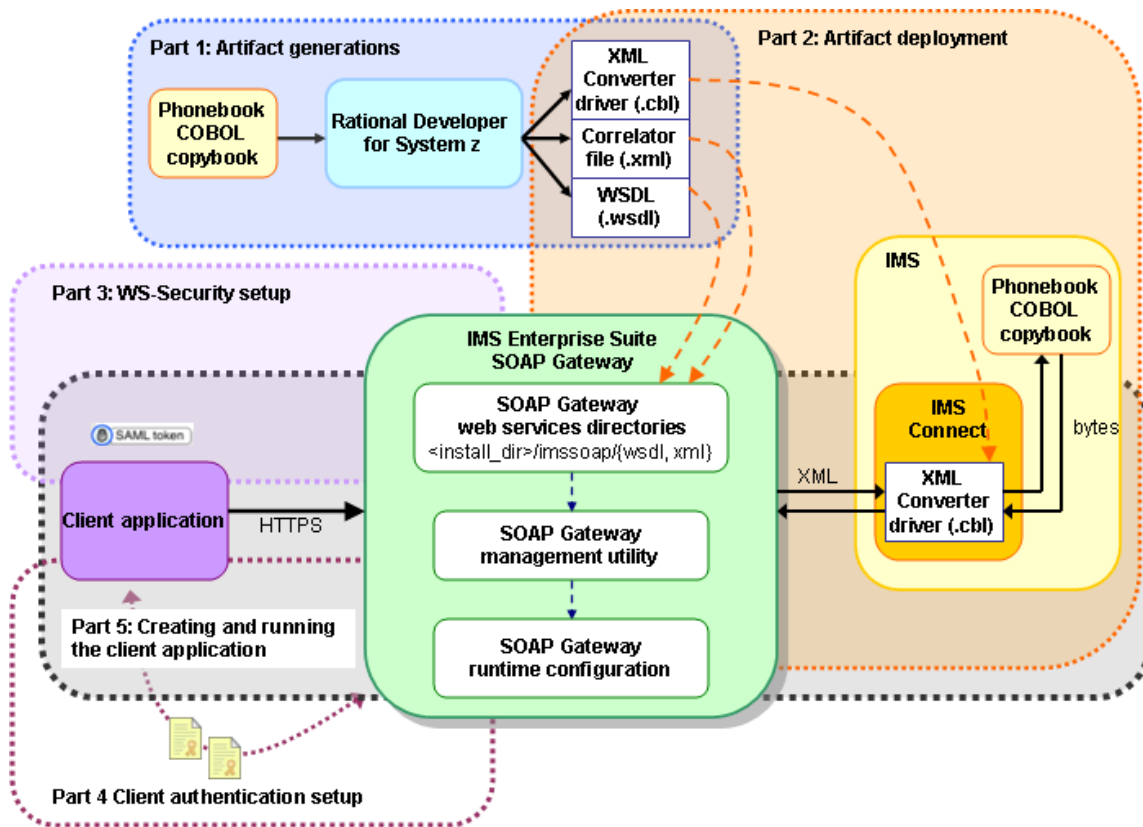


Figure 22. Overall task flow

Additional resources

For more information about the web service provider scenario support in SOAP Gateway, see:

- [Web service provider scenario](#) for an overview of how IMS applications can be enabled as a web service and related support for security.
- [Enabling an IMS application as a web service provider](#) for information about how to create the required web service artifacts, deploy the web service, and write a client application.

For more information about creating client applications, see

<http://axis.apache.org/axis2/java/core/docs/userguide-creatingclients.html>.