IBM InfoSphere Information Services Director
Version 8 Release 7

Guide to Publishing Secure Services

SC19-3484-00

IBM InfoSphere Information Services Director
Version 8 Release 7

Guide to Publishing Secure Services

SC19-3484-00

Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 49.

Contents

Publishing secure services

After you have designed and deployed a service by using IBM® InfoSphere® Information Services Director, use the IBM InfoSphere Information Server console or an assembly tool to secure that service.

Before you begin

- These topics assume that you are using IBM InfoSphere Information Server and InfoSphere Information Services Director 8.5 with IBM WebSphere® Application Server, Versions 6.1 or 7.0, at the supported corresponding fix pack levels. To find the supported fix pack level for your installed version of WebSphere Application Server, refer to the IBM InfoSphere Information Server system requirements page: http://www.ibm.com/software/data/infosphere/info-server/overview/requirements.html

About this task

- These tasks assume that you are familiar with Web services, SOAP Over HTTP or SOAP Over JMS, and other service-oriented architecture (SOA) concepts, and have experience with InfoSphere Information Services Director and the design, deployment, and testing of services by using InfoSphere Information Server.
- The examples of securing a deployed services in these tasks use the SOAP over HTTP binding.
- Many of the tasks consist of modifying deployment descriptor files of services. You can use an assembly tool, such as IBM Rational® Application Developer or IBM WebSphere Application Server Toolkit, to manipulate these various descriptor files. The tasks in these topics use Rational Application Developer 7.0 and IBM WebSphere Application Server Toolkit 6.0. The assembly tool that is used for most of these procedures is Rational Application Developer, Version 7.5.4.
  - Rational Application Developer and Rational Software Architect share the same code base so Rational Application Developer can be interchanged with Rational Software Architect hereafter. Rational Software Architect is a superset of Rational Application Developer and other IBM Rational offerings (Rational Web Developer and Rational Software Modeler, for example).
  - Among the various IBM assembly tools available, Rational Software Architect or Rational Application Developer provide better support for Web services security, in particular by automating most of the tasks and abstracting most of the security details found in the deployment descriptors through a series of wizards. Aside from the wizards, the user interfaces and editors in Rational Application Developer 7.0 and IBM WebSphere Application Server Toolkit 6.0 are nearly identical.

Web services security overview

Web services security is an industry standard that describes how payload specific security can be applied to standard SOAP-based Web services.

Note: Although the term "Web services" typically refers to SOAP-formatted messages transported over HTTP, this topic talks about a series of standards called WS-Security or Web Services Security. For example, the WS-Security username token authentication is one of the many security technologies that are part of the

WS-Security standards. Some security mechanisms are not part of WS-Security, for example, HTTP basic authentication, J2EE authorization, SSL encryption.

Web services security relies on other accepted standards such as XML digital signature and XML encryption, and takes advantage of mathematical algorithms, techniques, and existing software assets that have long been used for communication and data security on the Internet and such older secure exchanges such as electronic data interchange (EDI) transactions. Web services security is necessary because HTTPS with SSL is insufficient for complex applications, especially those that require lengthy process flows with different handlers (applications) that take part in an entire Web service "conversation" or activity.

A commonly used illustration is a service that encompasses an entire purchase request. It might contain credit card details intended for the billing department, account update information for customer service, and simple product number information for completing an order and having it delivered. In a distributed service environment, different applications in the enterprise work together to provide the complete service. It might be critical that the inventory application have access to product ordering details determine if the desired product is in stock, but equally critical that the inventory application not have access to detailed account information or credit card numbers. It is important that the parties engaging in Web services interactions are able to validate the other party and ensure that no one altered the contents of the message while it was in transit.

Authentication, integrity, and confidentiality are the core security capabilities of Web services security. Other aspects of security, such as non-repudiation, can be derived from the core functions in the Web services security framework.

Security, and Web services security in particular, is increasingly coming up in discussions about service-oriented architecture (SOA) and Information as a Service. Adoption of the Web services security standard has been slow by vendors and customers, but it has been several years since the first level of Web services security standards were confirmed, and interest is on the rise. Reasons for slow adoption include complexity, concerns about performance, lack of vendor supported tooling, standards immaturity (or fear of it), and the simple fact that for many Web services implementations, Web services security is not considered necessary. HTTPS with SSL has been acceptable for most point-to-point Web services implementations, and complex multi-hopping, where Web services security becomes imperative, is still bleeding edge for most enterprises. Additionally, many production Web services implementations exist behind firewalls or do not expose sensitive information, which eliminates the need for Web services security.

Now that SOA is more common (at least in corporate architecture and planning discussions) and Web services is being prescribed more frequently for complex multi-hop solutions where many parties "touch" the payload of a message, Web services security is getting increased attention. Consequently, Web services security, and the ability for IBM InfoSphere Information Services Director to support it, is on the minds of architects and decision makers both inside of IBM and out.

Web services security is not easy. Implementing it requires a large degree of planning and discipline, and cooperation between client and server deployment teams. This involves a whole lot of careful thinking about things like:

- Which kinds of Web services security do I need and want? (authentication, signing, encryption, and so on)

- Which directions? (Client to host? Host back to client? Both directions for all kinds of Web services security? Forwarding and intermediaries?)
- Which parts and subparts of a message? Which technologies (X.509, for example)?

From a technology perspective, most of the runtime technology defined by Web services security already exists. Industry proven techniques for digital signing, private key encryption algorithms, and integration with various protocols are utilized in Web services security. What is new is the syntax and rules for the definition of Web services security for requestor and provider engines, and the specific file names, extensions, and custom properties represented by each vendor implementation.

Adding security by using the IBM InfoSphere Information Server console

By using IBM InfoSphere Information Services Director in the IBM InfoSphere Information Server console, you can add HTTP basic authentication, WS-Security username token authentication, or data confidentiality that uses SSL encryption.

About this task

You can add security to your services in two ways: by using the IBM InfoSphere Information Server console or by using an assembly tool. Using IBM InfoSphere Information Server console is the preferred method because the IBM InfoSphere Information Server console automates some tasks for you.

Adding HTTP basic authentication by using the IBM InfoSphere Information Server console

Complete this task to add HTTP basic authentication to your service.

About this task

This task assumes that the binding attached to the service is SOAP over HTTP. The configuration is similar for other bindings such as REST, REST 2.0, RSS, or EJB.

Procedure

1. In the Information Services Application workspace, select an application and click **Open**.
2. Click **Edit**.
3. Double-click the service that you want to secure with HTTP basic authentication.
4. In the service Overview pane, select **Requires Authentication** in the Security panel.
5. Click the service Bindings pane.
6. From the **Authentication Support** menu, select **HTTP Basic**.
7. Click **Save Application**.
8. Click **Close Application**.

What to do next

You must redeploy the service for these changes to take effect.

When you add HTTP authentication to a service by using the IBM InfoSphere Information Server console, IBM InfoSphere Information Services Director adds a default authorization rule that allows only users with the InfoSphere Information Services Director Consumer, Administrator, or Catalog Manager role to invoke the operations within those services.

Adding user name token authentication by using the IBM InfoSphere Information Server console

Complete this task to add user name token authentication to your service.

About this task

This task assumes that the binding attached to the service is SOAP over HTTP.

Procedure

1. In the Information Services Application workspace, select an application and click **Open**.
2. Click **Edit**.
3. Double-click the service that you want to secure with user name token authentication.
4. In the service Overview pane, select **Requires Authentication** in the Security panel.
5. Click the service Bindings pane.
6. From the **Authentication Support** menu, select **WS-Security Username Token**.
7. Click **Save Application**.
8. Click **Close Application**.

What to do next

You must redeploy the service for these changes to take effect.

When you add authentication to a service by using the IBM InfoSphere Information Server console, IBM InfoSphere Information Services Director adds a default authorization rule that allows only users with the InfoSphere Information Services Director Consumer, Administrator, or Catalog Manager role to invoke the operations within those services.

Adding SSL encryption data confidentiality by using the IBM InfoSphere Information Server console

Complete this task to add SSL encryption data confidentiality to your service.

About this task

This task assumes that the binding attached to the service is SOAP over HTTP. The configuration is similar for other bindings such as Text Over HTTP, REST, REST 2.0, or RSS.

Procedure

1. In the Information Services Application workspace, select an application and click **Open**.
2. Click **Edit**.

3. Double-click the service that you want to secure with SSL encryption data confidentiality.
4. In the service Overview pane, select **Requires Confidentiality** in the Security panel. If you click the service Bindings pane, you will see that HTTPS is required to invoke this service.
5. Click **Save Application**.
6. Click **Close Application**.

What to do next

Your application server must be configured for HTTPS and SSL to use an SSL encrypted service.

Adding security by using an assembly tool

By using an assembly tool, you can add authentication, authorization, data confidentiality and data integrity.

About this task

You can add security to your services in two ways: by using the IBM InfoSphere Information Server console or by using an assembly tool. Using the IBM InfoSphere Information Server console automates some tasks that you must do manually with an assembly tool. However, using an assembly tool allows you to secure a service in ways that are not possible by using the IBM InfoSphere Information Server console.

Binding structure overview

To enable security features without using an application such as an assembly tool, you need to understand the basic internal structure of bindings to modify their settings.

This section provides an overview of the internal structure of two popular bindings: SOAP over HTTP and SOAP over JMS.

SOAP over JMS

SOAP over JMS is one type of binding that you can attach to a service. Read this topic to understand the basic structure of a SOAP over JMS binding, which is helpful to know if you intend to enable security settings without using an application such as an assembly tool.

The SOAP over JMS binding is responsible for unmarshalling SOAP requests that are sent over a JMS queue or topic to the service, redirecting them to the IBM InfoSphere Information Server service implementation as EJB calls and marshalling the response in a SOAP message sent back to the JMS queue or topic.

A SOAP over JMS binding consists of the following artifacts:

- A WSDL file that describes the service.
- An application server-specific router Message-Driven Bean (MDB) that listens for incoming SOAP requests over JMS.
- A facade stateless session EJB that redirects requests to the actual InfoSphere Information Server service implementation, which contains the business logic a client is seeking to invoke.

- A set of serializer and deserializer classes that map SOAP messages into Java objects (WSDL types into Java type, for example) according to the JAX-RPC specifications.

These artifacts are bundled into:

- An MDB jar (soapjrouter.jar) that contains the router MDB and its deployment descriptors.
- An EJB jar (soapjbinding.jar) that contains the facade EJB, its deployment descriptors, the WSDL file and the serializer and deserializer classes.

After IBM InfoSphere Information Services Director creates an enterprise archive (EAR) file of the service before it is deployed, InfoSphere Information Server creates and packages the MDB and EJB modules into the service implementation EAR file when the service is deployed, as shown in the following diagram.

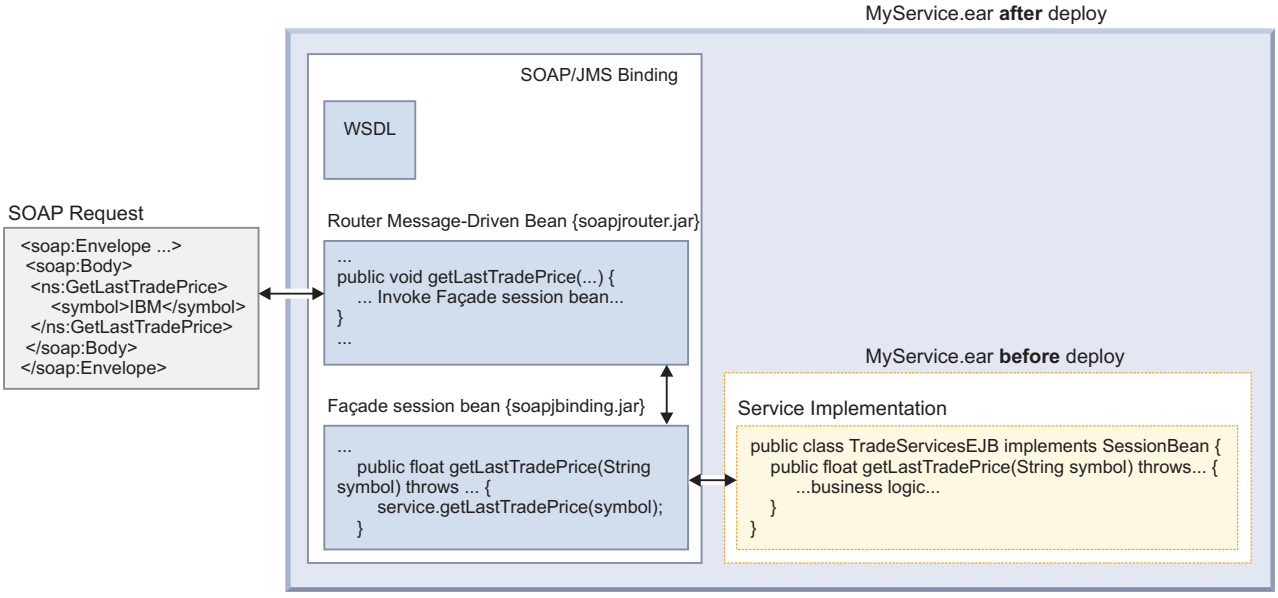

Figure 1. A deployed service with SOAP over JMS binding

SOAP over HTTP

SOAP over HTTP is one type of binding that you can attach to a service. Read this topic to understand the basic structure of a SOAP over HTTP binding, which is helpful to know if you intend to enable security settings without using an application such as an assembly tool.

The SOAP over HTTP binding can be seen as a server-side proxy component that sits in between clients and services. It is responsible for unmarshalling SOAP requests that are sent over HTTP to the service, redirecting them to the IBM InfoSphere Information Server service implementation as EJB calls and marshalling the response in a SOAP message sent over HTTP back to the client.

A SOAP over HTTP binding effectively consists of the following artifacts:

- A WSDL file that describes the service.
- An application server-specific router servlet that listens for incoming SOAP requests over HTTP.
- A facade stateless session EJB that redirects requests to the actual service implementation, which contains the business logic a client is seeking to invoke.

- A set of serializer and deserializer classes that map SOAP messages into Java objects (WSDL types into Java type, for example) according to the JAX-RPC specifications.

These artifacts are bundled into:

- A Web module (soaprouter.war) that contains the router servlet and its deployment descriptors.
- An EJB jar (soapbinding.jar) that contains the facade EJB, its deployment descriptors, the WSDL file and the serializer and deserializer classes.

After IBM InfoSphere Information Services Director creates an enterprise archive (EAR) file of the service before it is deployed, InfoSphere Information Server creates and packages the Web and EJB modules into the service implementation EAR file when the service is deployed, as shown in the following diagram.

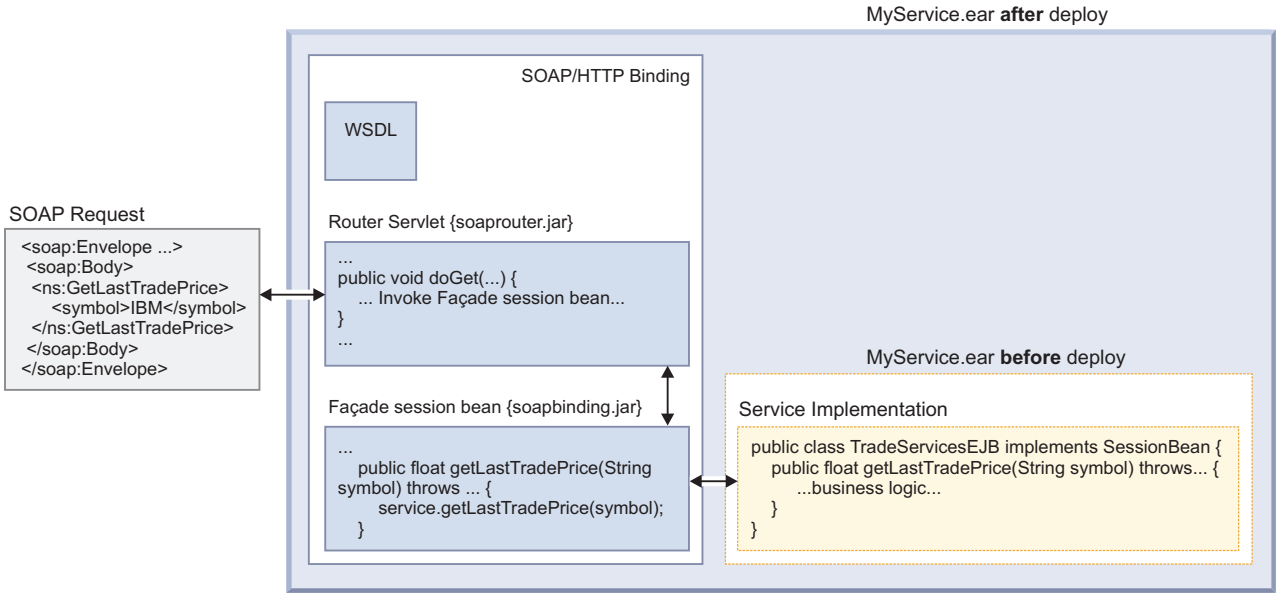

Figure 2. A deployed service with SOAP over HTTP binding

Configuring an assembly tool with IBM InfoSphere Information Server

To publish secure services, configure an assembly tool to assist you in modifying the descriptor files for service bindings.

About this task

IBM InfoSphere Information Server Version 8.5 includes IBM WebSphere Application Server 6.1 or 7.0 and this needs to be reflected in the assembly tool configuration. (For supported WebSphere Application Server fix pack levels, refer to the IBM InfoSphere Information Server system requirements page: http://www.ibm.com/software/data/infosphere/info-server/overview/requirements.html.) More precisely, theWebSphere Application Server 6.0 or 7.0 runtime libraries must be made available to the assembly tool. This will spare you various errors and warnings while importing your InfoSphere Information Server application enterprise archive (EAR) file into the assembly tool, and also avoids runtime errors if you plan to test your service by using a proxy test client generated by your assembly tool.

The following procedure describes how to configure IBM Rational Application Developer with IBM WebSphere Application Server, Version 6.1 or Version 7.0. The steps to configure IBM WebSphere Application Server Toolkit are identical.

Procedure

1. In Rational Application Developer, click **Window** > **Preferences**.
2. From the list of preferences, select **Server** > **Installed Runtimes**.
3. In the Installed Server Runtime Environments window, click **Add** to add the InfoSphere Information Server runtime libraries.

   The window lists IBM WebSphere Application Server runtimes, from version 6.1 to version 7.0. Some runtime libraries might be stubs; others might contain the complete implementation code, depending on what you selected during the Rational Application Developer installation process.
4. In the New Server Runtime window, select **WebSphere Application Server v6.1** or **WebSphere Application Server v7.0** from the **IBM** folder and click **Next**.
5. In the **Name** field, type `Information Server WAS v6.1` or `Information Server WAS v7.0`, and the installation directory (for example, `C:\IBM\WebSphere\AppServer`) for this run time and click **Finish**.
6. Click **OK** to exit the Preferences window.

Disabling an unsecured service by using the IBM InfoSphere Information Server console

Before stopping and exporting a service by using the IBM WebSphere Application Server console, you must disable the unit of business logic (an IBM InfoSphere DataStage® job, for example) that the service contains. To disable the service, you use the IBM InfoSphere Information Server console.

Before you begin

- Create a service

Procedure

1. Log in to the IBM InfoSphere Information Server console and open the project that contains the service that you want to disable.
2. Select **OPERATE** > **Deployed Information Services Applications**.
3. From the View pane, select the application that contains the service that you want to disable.
4. Click **Edit**.
5. In the bottom of the View pane, click **Disable** and select **Disable**.
6. Click **Save** and **Close**.

Exporting a service by using the WebSphere Application Server console

Before you can enable security features for a service, export the application that contains the service to create an enterprise archive (EAR) file. The EAR file contains the descriptor files that you must modify to enable security features such as authentication.

Before you begin

- Create a service

- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8

Procedure

1. In WebSphere Application Server, Version 6.1, select **Applications** > **Enterprise Applications**.

   In WebSphere Application Server, Version 7.0, select **Applications** > **Application Types** > **WebSphere enterprise applications**.
2. Select the application that contains the service you want to export and click **Stop**.
3. Select the application again and click **Export**.
4. In the Export Application EAR files window, click the application name.
5. Save the file to a directory on your system.
6. In the Export Application EAR files window, click **Back** to go back to the Enterprise Applications window.

Importing an unsecured service EAR file into an assembly tool

After you have disabled the unsecured service and exported it as an enterprise archive (EAR) file, you can use an assembly tool to import the enterprise archive (EAR) file and enable security features such as authentication, authorization, or confidentiality to a service.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8

About this task

This procedure shows you how to import a service EAR file into IBM Rational Application Developer. The steps are similar if you use IBM WebSphere Application Server Toolkit to import the service EAR file.

Procedure

1. In the **Project Explorer** tab, right-click and select **Import** > **EAR file**.
2. In the Import window, click **Browse** to select the location of the EAR file.
3. Select a project name.
4. Select the target runtime environment that you specified when you configured the assembly tool. If you followed the task to configure Rational Application Developer with InfoSphere Information Server, the correct runtime environment is **Information Server WAS v6.1** or **Information Server WAS v7.0**.

   Important: If you select a runtime environment other than the one you selected when you configured the assembly tool, the EAR file might not be imported correctly.
5. Click **Next** to move to the next Import window.

6. Click **Next** to move to the EAR Module and Utility JAR Projects window.
7. Click **Finish**.

Results

The EAR file is imported into the assembly tool. You can now view the EAR file and the internal modules, such as soapbinding, in the **Project Explorer** tab.

Note: You can ignore the following warning if it is displayed in the **Problems** tab: **CHKJ2874W: Migrate this EJB module's default datasource binding to a default CMP Connection Factory binding**.

Securing a service

There are four ways that you can secure a service.

Adding authorization by using J2EE security constraints

Add authorization as one way to secure a service.

About this task

To add authorization, you define the EJB method permissions in the EJB deployment descriptor. You can edit the deployment descriptor in the XML file, or you can use an assembly tool such as IBM WebSphere Application Server Toolkit.

In the security model defined by the J2EE specifications, you define EJB method permissions in a declarative fashion in the deployment descriptor as opposed to defining them programmatically in the service implementation.

You can, for example, restrict access to a service only to users that have the Suite Administrator role in IBM InfoSphere Information Server. In this case, you edit the service session bean deployment descriptor `ejb-jar.xml` to define security roles and security method permissions.

Adding J2EE security constraints without using an assembly tool:

As defined by the J2EE specification security model, add authorization to a service by editing the EJB deployment descriptor. You can edit the deployment descriptor in the XML file as described in this task.

Before you begin

- Create a service
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8

Procedure

1. Unpack the content of the *applicationName*`.ear` file in a temporary working directory.
2. Edit the service session bean deployment descriptor *serviceName*`.jar#META-INF/ejb-jar.xml` and add the following security elements under the **`<assembly-descriptor>`** element.

   `security-role`
   This element is required and can appear multiple times. Declares

security roles that are allowed to call methods defined in this EJB. If you specify a role that the **<method-permission>** elements use, make sure that you declare it in the **<security-role>** element. The exact definitions of which roles can invoke which EJB method are in the **<method-permission>** element.

description
This element is a child element of the **<security-role>** element and is optional. A text description of this security role.

role-name
This element is a child element of the **<security-role>** element and is required. A role name that has been granted some permissions to access the EJB.

method-permission
This element is required and can appear multiple times. Defines which roles can invoke which EJB methods.

role-name
This element is a child element of the **<method-permission>** element. This element is optional and can appear multiple times. A role name can invoke the methods defined by the **<method>** elements. Cannot be specified along with an **<unchecked>** element.

unchecked
This element is a child element of the **<method-permission>** element and is optional. Specifies that no access permission is needed to invoke the methods defined by the **<method>** elements. Cannot be specified along with a **<role-name>** element.

method This element is a child element of the **<method-permission>** element. This element is required and can appear multiple times. Defines an EJB method that is subject to either a **<role-name>** or **<unchecked>** access permission.

ejb-name
This element is a child element of the **<method>** sub-element and is required. The name of the EJB where this method is defined.

method-name
This element is a child element of the **<method>** sub-element and is required. The name of the EJB method.

Note: Required elements are to be considered in the context of enabling authorization. A required element is not necessarily required by the `ejb-jar.xml` XML Schema, but it is required to enable authorization.

3. Pack the edited deployment descriptor along with the rest of the untouched files back into the *applicationName*`.ear` file.

Example

The following example shows SOAP over HTTP binding `web.xml` and `ejb-jar.xml` files where HTTP basic authentication (with no SSL encryption) has been enabled for all users that are part of the SuiteUser and SuiteAdmin roles:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.
//DTD Enterprise JavaBeans 2.0//EN"
"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar id="ejb-jar_ID">
<description/>
    <display-name>MyApp</display-name>
    <enterprise-beans>
      <session id="MyService">
         <description/>
         <display-name>MyService</display-name>
         <ejb-name>MyService</ejb-name>
         <home>com.ibm.isd.MyApp.MyService.server.MyServiceHome
         </home>

<remote>com.ibm.isd.MyApp.MyService.server.MyServiceRemote
</remote>
         <ejb-class>com.ibm.isd.MyApp.MyService.server.impl.
         MyServiceBean</ejb-class>
         <session-type>Stateless</session-type>
         <transaction-type>Container</transaction-type>
         <resource-ref id="ResRef_MyService_Ref">
           <res-ref-name>jca/AgentConnectionFactory
           </res-ref-name>
           <res-type>com.ascential.asb.agent.ra.ConnectionFactory
           </res-type>
           <res-auth>Application</res-auth>
           <res-sharing-scope>Unshareable</res-sharing-scope>
         </resource-ref>
      </session>
    </enterprise-beans>
    <assembly-descriptor>
      <security-role>
        <role-name>SuiteUser</role-name>
      </security-role>
      <method-permission>
        <role-name>SuiteUser</role-name>
        <method>
          <ejb-name>MyService</ejb-name>
          <method-name>doNothing</method-name>
          <method-params>
            <method-param>int</method-param>
          </method-params>
        </method>
      </method-permission>
    </assembly-descriptor>
</ejb-jar>
```

Adding J2EE security constraints by using an assembly tool:

As defined by the J2EE specification security model, add authorization to a service by editing the EJB deployment descriptor. You can edit the deployment descriptor in the XML file, or you can use an assembly tool. This task documents the assembly tool method by using IBM WebSphere Application Server Server Toolkit.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8

- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

Procedure

1. In the **Project Explorer** tab, expand **EJB Projects** > **svcs** and double-click the deployment descriptor file to edit it.
2. Click **Assembly**.
3. In the **Security Roles** section, type the name of the role (SuiteUser, for example) that you want to add in the **Name** field and click **Add**.
4. In the **Method Permissions** section, click **Add**.
5. In the Add Method Permission window, select the security role that you want to add and click **Finish**.
6. Click **Save** to save your changes.

Adding authentication mechanisms

Add authentication as one way to secure a service.

About this task

The methods of authentication that are covered in this task are:

- Unsigned security tokens
- Signed security tokens
- HTTP basic authentication

Adding unsigned security tokens by using the WS-Security wizard:

An unsigned security token that authenticates incoming service requests is one way to secure a service. Typically, an unsigned security token refers to a user name token.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

About this task

You can accomplish this task by using the WS-Security Wizard or the Web Services Editor. The Web Services Editor method is the more manual method. The following procedure describes how to add a user name token to your service by using the WS-Security wizard.

Procedure

1. Change to the **Java EE perspective**.
2. In the **Project Explorer** tab, expand the **soapbinding** of the service that you imported, and then expand **Services**.
3. Right-click your service and select **Secure Web Service** > **Add Stand Alone Security Token**.
4. In the WS-Security Wizard - Server Side Stand Alone Security Token window, select the **Username Token** token type.
5. Select the **system.wssecurity.UsernameToken** JAAS configuration name. This name indicates that the incoming login requests will be processed by the stack of login modules defined in the `system.wssecurity.UsernameToken` JAAS login configuration.
6. Click **Finish**.
7. To validate that the service deployment descriptors have been updated correctly, in the **Project Explorer** tab, expand **soapbinding** > **Services** and right-click your service. Select **Show** and then click **Deployment Descriptor**. The Web Services Editor opens.
8. Click the **Extensions** tab to verify that a user name token has been created in **Request Consumer Service Configuration Details** > **Required Security Token**.
9. Click the **Bindings** tab to verify that a user name token has been created in **Request Consumer Binding Configuration Details** > **Token Consumer**.

Adding unsigned security tokens by using the Web Services Editor:

An unsigned security token that authenticates incoming service requests is one way to secure a service. Typically, an unsigned security token refers to a user name token.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

About this task

You can accomplish this task by using the WS-Security Wizard or the Web Services Editor. The Web Services Editor method is the more manual method. This procedure describes how to add a user name token to your service by using the Web Services Editor.

Procedure

1. In the **Project Explorer** tab of Rational Application Developer, expand **soapbinding** > **ejbModule** > **META-INF** and double-click the **webservice.xml** descriptor file.

2. Click the **Extensions** tab in the Web Services Editor.
3. Expand **Web Service Description Extension** and select the description for your service.
4. Expand **Port Component Binding** and select the binding for your service.
5. Expand **Request Consumer Service Configuration Details** > **Required Security Token**.
6. Click **Add**.
7. In the Required Security Token window, select the **Username Token** token type and complete the remaining fields.
8. Click **OK**. The user name token is displayed under **Request Consumer Service Configuration Details** > **Required Security Token**.
9. Click the **Bindings Configurations** tab and expand **Token Consumer**.
10. Click **Add**.
11. In the **Token Consumer** window, complete the name, class, security token, value type, and URI fields. Make sure to select the newly created security token in the **Security token** drop-down list. Also make sure to select the **Use jaas.config** check box and specify `system.wssecurity.UsernameToken` in the **jaas.config name** field. For all other fields, you can use the default values.
12. Click **OK**. The token consumer is displayed under **Request Consumer Binding Configuration Details** > **Token Consumer**.

Adding signed security tokens by using the Web Services Editor:

A signed security token that authenticates incoming service requests is one way to secure a service.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

About this task

Examples of signed security tokens are Kerberos ticket tokens, X.509 certificate tokens or Lightweight Third Party Authentication (LTPA) tokens. In this procedure, IBM Rational Application Developer is used to illustrate the definition of an X.509 certificate token as the authentication mechanism for incoming service requests.

Procedure

1. In the **Project Explorer** tab of Rational Application Developer, expand **soapbinding** > **ejbModule** > **META-INF** and double-click the `webservice.xml` descriptor file.
2. Click the **Extensions** tab in the Web Services Editor.

3. Expand **Web Service Description Extension** and select the description for your service.
4. Expand **Port Component Binding** and select the binding for your service.
5. Expand **Request Consumer Service Configuration Details** > **Required Security Token**.
6. Click **Add**.
7. In the Required Security Token window, select the **X509 certificate token** token type and complete the remaining fields.
8. Click **OK**. The X509 certificate token is displayed under **Request Consumer Service Configuration Details** > **Required Security Token**.
9. Click the **Bindings Configurations** tab and expand **Token Consumer**.
10. Click **Add**.
11. In the **Token Consumer** window, complete the name, class, security token, value type, and URI fields. Other fields can be left at their default values.
12. Click **OK**. The token consumer appears under **Request Consumer Binding Configuration Details** > **Token Consumer**.

Adding HTTP basic authentication without using an assembly tool:

Use the HTTP basic authentication method to secure a service within a secured network such as HTTPS or an intranet. To enable HTTP basic authentication, edit J2EE deployment descriptors defined in the SOAP binding module.

Before you begin

- Create a service
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8

About this task

To add authentication to a service, you edit the EJB deployment descriptor. You can edit the deployment descriptor in the XML file, or you can use an assembly tool such as IBM Rational Application Developer.

Procedure

1. Unpack the contents of your EAR file in a temporary working directory.
2. Edit the router servlet Web deployment descriptor `soaprouter.war#WEB-INF/web.xml` and add the following security elements under the **<web-app>** root element.

   security-constraint
   This element is required. Defines the access privileges to a collection of resources defined by the **<web-resource-collection>** sub-element.

   web-resource-collection
   This element is a child element of the **<security-constraint>** element and is required. Defines the components of the Web application to which this security constraint is applied.

   web-resource-name
   This element is a child element of the

<web-resource-collection> sub-element and is required. Defines the name of this Web resource collection.

description
This element is a child element of the **<web-resource-collection>** sub-element and is optional. A text description of this Web resource collection.

url-pattern
This element is a child element of the **<web-resource-collection>** sub-element and is required. Defines which URL patterns this security constraint applies. In the case of the SOAP router servlet, the URL pattern /* should be used. The asterisk (*) alone is not valid.

http-method
This element is a child element of the **<web-resource-collection>** sub-element. This element is optional and can appear multiple times. Use one or more of the **<http-method>** elements to declare which HTTP methods (GET or POST, for example) are subject to the security constraint. By default, it is applied to all HTTP methods. In the case of the SOAP/HTTP router servlet, requests are made by using the POST method. If you specify the **<http-method>** element, make sure that the POST method is listed. To protect the WSDL file, make sure to specify the GET method as well.

auth-constraint
This element is a child element of the **<security-constraint>** element and is required. Defines which groups or principals have access to the collection of Web resources defined in this security constraint.

description
This is a child element of the **<auth-constraint>** sub-element and is optional. A text description of this security constraint.

role-name
This element is a child element of the **<auth-constraint>** sub-element. This element is required and can appear multiple times. Defines which security roles can access resources defined in this security constraint (SuiteUser or SuiteAdmin, for example). Replicates the roles that have been defined for the service implementation.

user-data-constraint
This element is a child element of the **<security-constraint>** element and is optional. Defines how the client communicates with the server. Typically used only for enabling SSL.

description
This element is a child element of the **<user-data-constraint>** sub-element and is optional. A text description of this user data constraint.

transport-guarantee
This element is a child element of the **<user-data-constraint>** sub-element and is optional. Specifies the type of security constraint to enforce on data transferred between client and server:

- NONE (no constraint)
- INTEGRAL (data integrity)
- CONFIDENTIAL (which is basically SSL)

login-config
This element is required. Defines how a user is authenticated.

auth-method
This element is a child element of the **<login-config>** element and is required. Specifies the method used to authenticate a user. In this case, set to BASIC only.

realm-name
This element is a child element of the **<login-config>** element and is optional. This element is optional. The name of the realm that is referenced to authenticate user credentials.

security-role
This element is required and can appear multiple times. Declares security roles. There should be one **<security-role>** element for each role listed under the **<auth-constraint>** element.

description
This element is a child element of the **<security-role>** element and is optional. A text description of this security role.

role-name
This element is a child element of the **<security-role>** element and is required. A role name that has been granted some permissions in this Web module.

Note: Required elements are to be considered in the context of enabling HTTP basic authentication. A required element is not necessarily required by the `web.xml` XML Schema, but rather it is required to enable HTTP basic authentication.

3. Edit the binding EJB deployment descriptor `soapbinding.jar#META-INF/ejb-jar.xml` and add the following security elements under the **<assembly-descriptor>** element.

security-role
This element is required and can appear multiple times. Declares security roles that are allowed to call methods defined in this EJB. If you specify a role that the **<method-permission>** elements use, make sure that you declare it in the **<security-role>** element. The exact definitions of which roles can invoke which EJB method are in the **<method-permission>** element.

description
This element is a child element of the **<security-role>** element and is optional. A text description of this security role.

role-name
This element is a child element of the **<security-role>** element and is required. A role name that has been granted some permissions to access the EJB.

method-permission
This element is required and can appear multiple times. Defines which roles can invoke which EJB methods.

role-name
This element is a child element of the **<method-permission>** element. This element is optional and can appear multiple times. A role name that can invoke the methods defined by the **<method>** elements. Cannot be specified along with an **<unchecked>** element.

unchecked
This element is a child element of the **<method-permission>** element and is optional. Specifies that no access permission is needed to invoke the methods defined by the **<method>** elements. Cannot be specified along with a **<role-name>** element.

method This element is a child element of the **<method-permission>** element. This element is required and can appear multiple times. Defines an EJB method that is subject to either a **<role-name>** or **<unchecked>** access permission.

ejb-name
This element is a child element of the **<method>** sub-element and is required. The name of the EJB where this method is defined.

method-name
This element is a child element of the **<method>** sub-element and is required. The name of the EJB method.

Note: Required elements are to be considered in the context of enabling HTTP basic authentication. A required element is not necessarily required by the `web.xml` XML Schema, but rather it is required to enable HTTP basic authentication.

4. Pack the edited deployment descriptors along with the rest of the untouched files back into the *`applicationName`*`.ear` file.

Example

The following example shows SOAP over HTTP binding `web.xml` and `ejb-jar.xml` files where HTTP basic authentication (with no SSL encryption) has been enabled for all users that are part of the SuiteUser and SuiteAdmin roles:

Example of the `web.xml` file

```
soaprouter.jar#WEB-INF/web.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" id="WebApp_ID" version="2.4">
  <display-name>MyAppSOAPBinding_HTTPRouter</display-name>
  <servlet>
```

```
    <display-name>Web services Router servlet for port-componentMyServiceSOAP
    </display-name>
    <servlet-name>MyServiceSOAP</servlet-name>
    <servlet-class>com.ibm.ws.webservices.engine.transport.http.WebServicesServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServiceSOAP</servlet-name>
    <url-pattern>MyService</url-pattern>
  </servlet-mapping>
  <security-constraint>
    <display-name>SoapRouterConstraints</display-name>
    <web-resource-collection>
      <web-resource-name>SecuredSoapRouterServlet</web-resource-name>
      <url-pattern>/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
      <description>Authorized roles</description>
      <role-name>SuiteUser</role-name>
      <role-name>SuiteAdmin</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
  <login-config>
    <auth-method>BASIC</auth-method>
  </login-config>
  <security-role>
    <role-name>SuiteUser</role-name>
  </security-role>
  <security-role>
    <role-name>SuiteAdmin</role-name>
  </security-role>
</web-app>
```

Example of the ejb-jar.xml file

```
soapbinding.jar#META-INF/ejb-jar.xml
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ejb-jar_ID"
version="2.1" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
  <enterprise-beans>
    <session id="MyServiceSOAP">
      <description></description>
      <ejb-name>MyServiceSOAP</ejb-name>
      <home>com.ibm.isd.MyApp.MyService.server.MyServiceHome
      </home>
      <remote>com.ibm.isd.MyApp.MyService.server.MyServiceRemote
      </remote>
      <service-endpoint>com.ibm.isd.MyApp.MyService.MyService
      </service-endpoint>
      <ejb-class>com.ibm.isd.MyApp.MyService.MyServiceSOAPBindingBean
      </ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <ejb-ref id="EjbRef_MyService_Ref">
        <ejb-ref-name>ejb/MyService</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.ibm.isd.MyApp.MyService.server.MyServiceHome</home>
        <remote>com.ibm.isd.MyApp.MyService.server.MyServiceRemote</remote>
      </ejb-ref>
    </session>
```

```
</enterprise-beans>
  <assembly-descriptor>
    <security-role>
      <role-name>SuiteUser</role-name>
    </security-role>
    <security-role>
      <role-name>SuiteAdmin</role-name>
    </security-role>
    <method-permission>
      <role-name>SuiteUser</role-name>
      <role-name>SuiteAdmin</role-name>
      <method>
        <ejb-name>MyServiceSOAP</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>doNothing</method-name>
        <method-params>
        <method-param>int</method-param>
        </method-params>
      </method>
    </method-permission>
    <method-permission>
      <unchecked/>
      <method>
        <ejb-name>MyServiceSOAP</ejb-name>
        <method-intf>Home</method-intf>
        <method-name>create</method-name>
        <method-params>
        </method-params>
      </method>
      <method>
        <ejb-name>MyServiceSOAP</ejb-name>
        <method-intf>Home</method-intf>
        <method-name>remove</method-name>
      </method>
    </method-permission></assembly-descriptor>
  </assembly-descriptor>
</ejb-jar>
```

Adding HTTP basic authentication by using an assembly tool:

Use the HTTP basic authentication method to secure a service within a secured network such as HTTPS or an intranet. To enable HTTP basic authentication, edit J2EE deployment descriptors defined in the SOAP binding module.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

About this task

To add authentication to a service, you edit the EJB deployment descriptor. You can edit the deployment descriptor in the XML file, Alternatively, you can use an

assembly tool such as IBM Rational Application Developer as described in this procedure.

Procedure

1. In the **Project Explorer** tab, expand the **soaprouter** module.
2. Double-click the Web deployment descriptor file to edit it.
3. Click the **Pages** tab.
4. In the **Login** section, select **Authentication method** > **BASIC** and save the change.
5. Click the **Security** tab.
6. In the **Security Roles** section, type the name of the role that you want to add in the **Name** field and click **Add**.
7. In the **Security Constraints** section, click **Add**.

   Make the security constraints for authentication the same as the security constraints that you define when you add authorization to the service.
8. Type a name for the constraint and click **Next**.
9. Type a Web resource name, select **POST** and **GET** for the **HTTP method**, and add `/*` to the **Pattern** field. In order to protect the service WSDL, you must protect the GET HTTP method.
10. Click **Finish**.
11. In the **Authorized Roles** section, click **Add**.
12. Select the **Role Name** that you added in the **Security Roles** section and click **Finish**.
13. In the **User Data Constraint** section, select **NONE** and save the changes. This constraint is used only for SSL connections.
14. In **EJB Projects** section of the **Project Explorer** tab, expand the **soapbinding** module.
15. Double-click the EJB deployment descriptor file to edit it.
16. Click the **Assembly** tab.
17. In the **Security Roles** section, click **Add** and type the same role name that you added to the Web deployment descriptor.
18. In the **Method Permissions** section, click **Add**.
19. In the Method Permission window, select the security role and click **Next**.
20. In the Enterprise Bean Selection window, select the enterprise bean for your service and click **Next**.
21. In the Method Elements window, select the methods that you want to secure. In a typical situation, select all the methods.
22. Click **Finish** and save the changes.

Adding data confidentiality by using SSL encryption

Add data confidentiality as one way to secure a service.

About this task

Add data confidentiality by using the Secure Sockets Layer (SSL) protocol. Complete this task to enable SSL on the server side of the service. You must then make the same changes on the client side configuration. For example, if you change the server side configuration to use the Transport Secure Layer (TSL) protocol instead of SSL, you must also configure the client to use TSL.

The SSL protocol is based on public key infrastructure (PKI). As such, private and public keys for both client and server must be generated and stored, respectively, in key stores and trust stores in order for the SSL connection to be completed. This task uses sample key stores and self-signed certificates. Use sample key stores only for testing purposes. In a real production environment, implement your own PKI.

This is an overview of the process to enable SSL:

- The client authenticates the server, after which a session is initiated. This authentication is called the SSL handshake protocol. Optionally, the client can also authenticate to the server.
- If the SSL handshake is successful, the data transfer is then initiated and encrypted. This is called the SSL report protocol.
- If there are any alarms at any point during the session, an alert packet is sent to the appropriate recipient. This alert packet is called the SSL alert protocol.

The following task describes how to add data confidentiality by using SSL encryption:

Configuring a service to use SSL encryption:

After you create a Secure Sockets Layer (SSL) repertoire within IBM WebSphere Application Server, complete this task to enable SSL on the server side of the service. You must then make the same changes on the client side configuration. For example, if you change the server side configuration to use the Transport Secure Layer (TSL) protocol instead of SSL, you must also configure the client to use TSL.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an EAR file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

Procedure

1. In the **Project Explorer** tab of IBM Rational Application Developer, expand **soaprouter** and double-click the deployment descriptor to edit it.
2. Select the **Security** tab.
3. In the **Security Constraints** section, click **Add**.
4. Enter a name for the constraint on the **Add Constraints** page and click **Next**.
5. Enter a name for the resource on the **Add Web Resource** page.
6. Select an HTTP method from the **HTTP Methods** list. Select at least the POST method to secure your service. If you do not want the security constraint to apply to the Web Services Description Language (WSDL) file, do not select the GET method.
7. Add `/*` to the **Pattern** field to specify the URL pattern that is applied to the security constraint.
8. Click **Finish**.

9. In the **User Data Constraint** section, select **CONFIDENTIAL** in the **Type** field.
10. Save the changes.

What to do next

Adding data integrity

Add data integrity by using an XML digital signature as one way to secure a service.

About this task

Overview of the digital signing process:

- The client (sender) signs parts of the request message by using a private key. The client private key is stored in the client keystore.
- The server (receiver) validates the client digital signatures in the request message by using the client public key. The client public key is stored in the server keystore.

Adding an XML digital signature:

To add data integrity to your service by using an XML digital signature, edit the deployment descriptor file by using an assembly tool such as IBM Rational Application Developer.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9

About this task

Restrictions

In this task, you use the Web Services Editor in IBM Rational Application Developer to edit the deployment descriptor file. Do not use the Rational Application Developer security wizard to edit the deployment descriptor file. The wizard does not allow you to control XML digital signature features such as signing and transforming algorithms.

This task shows how to sign and validate digital signatures for request messages, not response messages. However, the procedure to sign and validate digital signatures for responses messages is virtually identical and thus not documented here. To summarize that task, you configure the response consumer (client side) and the response generator (server side) the same way that the request generator (client side) and the request consumer (server side) are configured in this section.

Procedure

1. In the **Project Explorer** tab of Rational Application Developer, expand **soapbinding** > **Services** and right-click your service. Select **Show** and then click **Deployment Descriptor**. The Web Services Editor opens.
2. In the Web Services Editor, click the **Extensions** tab.
3. Expand **Web Service Description Extension** and select the description for your service.
4. Expand **Port Component Binding** and select the binding for your service.
5. Expand **Request Consumer Service Configuration Details** > **Required Integrity**.
6. In the Required Integrity window, click **Add**.
7. In the **Required integrity name** field of the Required Confidentiality window, type a name.
8. In the **Usage type** field, select **Required** if you want the integrity to be required or **Optional** if you want the integrity to be optional.
9. In the **Message Parts** field, select the message parts that you want to be signed. You can use either keywords or XPath expressions to sign the message parts. If you want to add message parts:
   a. Click **Add**.
   b. You can use keywords to identify generic parts of your message, select **http://www.ibm.com/websphere/webservices/wssecurity/dialect-was** from the Parts Dialect column and select the part of the message that you want to sign from the Parts Keyword column.
   c. You can use XPath expressions to identify more specific parts of your message, select **http://www.w3.org/TR/1999/REC-xpath-19991116** from the Parts Dialect column.
   d. Optional: In the **Message Parts** field, specify a nonce to avoid reply attacks.
   e. Optional: In the **Message Parts** field, specify a timestamp to assign a lifetime to the message.
10. In the Web Services Editor, click the **Bindings Configurations** tab.
11. Expand **Request Consumer Binding Configuration** > **Trust Anchor** and click **Add**.
12. In the Trust Anchor window:
    a. Type a trust anchor name.
    b. Type a password for the keystore.
    c. Type the full path to the server-side keystore, for example, `C:\sampleks\receiverkeys.jks`.
    d. Select the keystore type.
    e. Click **OK**.
13. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Token Consumer** and click **Add**.
14. In the Token Consumer window:
    a. Type a token consumer name.
    b. Select **com.ibm.wssip.wssecurity.token.X509TokenConsumer** as the token consumer class.
    c. Do not select a security token. For signing, it is not necessary to specify a security token in the service extensions.
    d. Select **Use value type** and select the **X509 certificate token v3** value type.

e. Select **Use jaas.config** and type `system.wssecurity.X509BST` in the **jaas.config name** field.

f. Select **Use certificate path settings** and **Certificate path reference** or **Trust any certificate**. If you select **Use certificate path settings**, select the trust anchor name that you created above.

g. Click **OK**.

15. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Key Locator** and click **Add**.
16. In the Key Locator window:
    a. Type a key locator name.
    b. Select a key locator class implementation. Use **com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator** if you are using the X.509 security token from the sender message for digital signature validation.
    c. Click **OK**.
17. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Key Information** and click **Add**.
18. In the Key Information window:
    a. Type a key information name.
    b. Select a key information type. Use **STRREF** as the key information type if the security token used for digital signature validation is directly referenced by using a URI in the sender message.
    c. Select a key information class. If you selected **STRREF** as the key information type, **com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentConsumer** will automatically be selected.
    d. Select **Use key locator** and select the name of the key locator that you created in the Key Locator section.
    e. Select **Use token** and select the name of the token consumer that you created in the Token Consumer section.
    f. Click **OK**.
19. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Signing Information** and click **Add**.
20. In the Signing Information window:
    a. Type a signing information name.
    b. Select a canonicalization method algorithm. The default value is **http://www.w3.org/2001/10/xml-exc-c14n#**.
    c. Select a signature method algorithm. The default value is **http://www.w3.org/2000/09/xmldsig#rsa-sha1**.
    d. Click **Add**, select the key information element that you created in the Key Information section, and type a key information name.
    e. Click **OK**.
21. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Part Reference** and click **Add**.
22. In the Part Reference window:
    a. Type a part reference name.
    b. Select a required integrity part that you created in the Required Integrity section.
    c. Select a digest method algorithm. The default value is **http://www.w3.org/2000/09/xmldsig#sha1**.

d. Click **OK**.

23. In the Web Services Editor, expand **Request Consumer Binding Configuration** > **Transforms** and click **Add**. The Transforms section becomes available after you create a part reference. Transforms elements hold information about the operations that were performed on data before it was signed.
24. In the Transform window:
    a. Type a transform name.
    b. Select an algorithm to perform the operation. The default value is **http://www.w3.org/2002/10/xml-exc-c14n#**.
    c. Click **OK**.
25. Save your changes.

Exporting a secured service EAR file from an assembly tool

After you enable security features such as authentication, authorization, or confidentiality to a service by using an assembly tool, you can export the secured service as an enterprise archive (EAR) file.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9
- Secure a service as described in "Securing a service" on page 10

About this task

This procedure shows you how to export a service EAR file from IBM Rational Application Developer. The steps are similar if you use IBM WebSphere Application Server Toolkit to export the service EAR file.

Procedure

1. In the **Project Explorer** tab, right-click the EAR file and select **Export** > **EAR file**.
2. In the Export window, click **Browse** to select the location of the EAR file.
3. Select the **Overwrite existing file** check box to overwrite the unsecured service EAR file.
4. Click **Finish**.

Redeploying a secured service with the IBM WebSphere Application Server console

After you enable security features such as authentication, authorization, or confidentiality to a service, and export the secured service as an enterprise archive (EAR) file, redeploy the secured service using the IBM WebSphere Application Server console.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9
- Secure a service as described in "Securing a service" on page 10
- Export the EAR file that contains the service as described in "Exporting a secured service EAR file from an assembly tool" on page 27

Procedure

1. Select **Applications** > **Enterprise Applications**.
2. If your application is not stopped (indicated with a red X in the status column), select the application and click **Stop**.
3. Select the application and click **Uninstall**.
4. In the Uninstall Application window, click **OK**.
5. Click **Save** to apply the changes to the master configuration and click **Save** in the window that is displayed.
6. In the Enterprise Applications window, select the application and click **Install**.
7. In the Preparing for the application installation window, click **Browse** to select the location of the secured service EAR file and click **Next**.
8. Click **Next** until you reach the application summary.
9. Click **Finish**. Messages indicating the progress of the installation are displayed.
10. Click **Save to Master Configuration** and click **Save** in the window that is displayed.
11. In the Enterprise Application window, select the newly installed application and click **Start**.

Enabling a secured service by using the IBM InfoSphere Information Server console

After the secured service is redeployed from the IBM WebSphere Application Server console, enable the service from the IBM InfoSphere Information Server console.

Before you begin

- Create a service

- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9
- Secure a service as described in "Securing a service" on page 10
- Export the EAR file that contains the service as described in "Exporting a secured service EAR file from an assembly tool" on page 27
- Redeploy the service EAR file as described in "Redeploying a secured service with the IBM WebSphere Application Server console" on page 28

Procedure

1. Log in to the IBM InfoSphere Information Server console and open a project that contains the service that you want to enable.
2. Select **OPERATE** > **Deployed Information Services Applications**.
3. From the View pane, select the application that contains the service that you want to enable.
4. Click **Edit**.
5. In the bottom of the View pane, click **Disable** and select **Enable**.
6. Click **Save** and **Close**.

Generating a test client

You can generate a test client to test your services. You can generate the test client with security enabled or disabled.

Generating a test client with security disabled

Use this task to generate a test client for a service with security that is disabled.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7

About this task

If you invoke a service secured by using user name token authentication with a non-secured test client, you will receive the following security exception:

```
exception: com.ibm.wsspi.wssecurity.SoapSecurityException:
WSEC5509E: A security token whose type is
http://myapp.wisd.ibm.com#UsernameTokenhttp:
//docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#UsernameToken] is required.
```

For an X.509 certificate token, you will receive the following security exception:

```
exception: com.ibm.wsspi.wssecurity.SoapSecurityException:
WSEC5509E: A security token whose type is
[X509Tokenhttp://docs.oasis-open.org/wss/2004
/01/oasis-200401-wss-x509-token-profile-1.0#X509] is required.
```

Procedure

1. In the **Servers** tab of IBM WebSphere Application Server, right-click anywhere inside the window and click **New** > **Server**.
2. In the **Define a New Server** page of the New Server window, enter a name for the server host, click **IBM** > **WebSphere v6.1 Server** or **WebSphere v7.0 Server** as the server type, and click **Information Server WAS v6.1** or **Information Server WAS v7.0** as the server runtime environment.
3. Click **Next** and ensure that the information on the following page is correct, including the profile name and authentication information. You can optionally click the **Test Connection** link to ensure that IBM Rational Application Developer can connect to WebSphere Application Server by using the authentication information that was just specified.
4. Click **Next**.
5. In the **Add and Remove Projects** page, click *MyApp*`.ear` file in the **Available projects** field and click **Add >** to add it to the list of configured projects.
6. Click **Next**, and then click **Finish**. With the server in place, whenever needed, you can now add a project to this server by right-clicking on the server in the **Servers** tab and selecting **Add and Remove Projects**.
7. In the **Servers** tab, select the server that you created and click the **Start the server** icon.
8. Select **File** > **New** > **Project** > **Dynamic Web Project**.
9. In the New Dynamic Web Project window, enter a project name (MyAppClient, for example), select the appropriate WebSphere Application Server version as the target runtime, select **Add project to an EAR**, and enter an enterprise archive (EAR) file project name (MyAppClient.ear, for example) for the EAR file that will contain the client code.
10. Click **Finish**.
11. Click **File** > **New** > **Other**.
12. In the **New** window, expand **Web Services**, select **Web Service Client**, and click **Next**.
13. In the **Web Services** page of the Web Service Client window, enter the Web Services Description Language (WSDL) URL (for example, http://localhost:9080/wisd/MyApp/MyService?wsdl) in the **Service definition** field.
14. Select **Test client** as the level of automation.
15. In the Configuration panel, verify these settings:

    Server Websphere v6.1 Server or Websphere v7.0 Server

    Web service runtime
    IBM WebSphere JAX-RPC

    Client project
    MyAppClient

    Client EAR project
    MyAppClient.ear
16. Click **Finish**.

17. In the **Web Service Client Test** page, select the **Web service sample JSPs** test facility and any method that you want to test. Use the default settings for the rest of this page and click **Finish**.
18. To see the test client in the **Project Explorer** under **JSR-109 Web Services** > **Clients**, restart IBM Rational Application Developer. You will need to see the test client if you want to use it to run the security wizards.
19. Optional: To test your service by using the test client, in the **Web Services Test Client** tab that opens, select a method, enter any input number, and click **Invoke**. The **Result** pane displays the result.
20. Optional: To invoke the test client without using the assembly tool, use a Web browser and enter the URL `https://`*`hostname`*`:`*`WASSSLChannelPort`*`/`*`applicationName`*`/sample`*`serviceName`*`PortTypeProxy/TestClient.jsp`

    hostname
    The name of the server hosting your test client (localhost, for example)

    WASSSLChannelPort
    The port number used by the SSL transport channel (9443 is the default)

    applicationName
    The name of your test client

    serviceName
    The name of the service that you want to invoke

Generating a test client with authentication enabled

Use this task to generate a test client for a service with authentication that is enabled.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9
- Add authentication to the service as described in "Adding authentication mechanisms" on page 13
- Export the EAR file that contains the service as described in "Exporting a secured service EAR file from an assembly tool" on page 27
- Redeploy the service EAR file as described in "Redeploying a secured service with the IBM WebSphere Application Server console" on page 28
- Enable the secured service as described in "Enabling a secured service by using the IBM InfoSphere Information Server console" on page 28

About this task

The IBM Rational Application Developer security wizard is used in this task to show how to enable WS-Security Username Token authentication on the test client. Alternatively, you can use the Rational Application Developer deployment descriptor editor to enable security on the client.

If you invoke a service that requires authentication but do not specify a user name and password, the following security exception or similar will be returned:

```
exception: com.ibm.wsspi.wssecurity.SoapSecurityException: WSEC5509E:
A security token whose type is [http://docs.oasis-open.org/wss/2004/
01/oasis-200401-wss-username-token-profile-1.0#UsernameToken] is required.
```

Similarly, if you invoke a service that requires Username Token authentication but specify an invalid user name or password, the following exception or similar will be returned:

```
exception: com.ibm.wsspi.wssecurity.SoapSecurityException: WSEC6521E:
Login failed. The exception is : javax.security.auth.login.LoginException:
WSEC6690E: Failed to check username [admin] and password in the
UserRegsitry: UserRegistryProcessor.checkRegistry()=false
```

Procedure

1. In the **Servers** tab of IBM WebSphere Application Server, right-click anywhere inside the window and select **New** > **Server**.
2. In the **Define a New Server** page of the New Server window, enter a name for the server host, select **IBM** > **WebSphere v6.1 Server** or **WebSphere v7.0 Server** as the server type, and the server runtime environment. If you followed the task to configure IBM Rational Application Developer with IBM InfoSphere Information Server, the correct runtime environment is **Information Server WAS v6.1** or **Information Server WAS v7.0**.
3. Click **Next** and ensure that the information on the following page is correct, including the profile name and authentication information.
4. Click **Next**.
5. In the **Add and Remove Projects** page, select your secured EAR file from the **Available projects** field and click **Add >** to add it to the list of configured projects.
6. Click **Next**, and then click **Finish**.
7. In the **Servers** tab, select the server that you created and click the **Start the server** icon.
8. Select **File** > **New** > **Project** > **Dynamic Web Project**.
9. In the New Dynamic Web Project window, enter a project name (MyAppClient, for example), select **Information Server WAS v6.1** or **Information Server WAS v7.0** as the target runtime, select **Add project to an EAR**, and enter an enterprise archive (EAR) project name (MyAppClient.ear, for example) for the EAR file that will contain the client code.
10. Click **Finish**.
11. Select **File** > **New** > **Other**.
12. In the **New** window, expand **Web Services**, select **Web Service Client**, and click **Next**.
13. In the **Web Services** page of the Web Service Client window, enter the Web Services Description Language (WSDL) URL (for example, `http://localhost:9080/wisd/MyApp/MyService?wsdl`) in the **Service definition** field.

14. Select **Test client** as the level of automation.
15. In the Configuration panel, verify these settings:

    Server Websphere v6.1 Server or Websphere v7.0 Server

    Web service runtime
    IBM WebSphere JAX-RPC

    Client project
    MyAppClient

    Client EAR project
    MyAppClient.ear
16. Click **Finish**.
17. In the **Web Service Client Test** page, select the **Web service sample JSPs** test facility and the method that you want to test. Use the default settings for the rest of this page and click **Finish**.
18. In the **Project Explorer** expand **JSR-109 Web Services** > **Clients**.
19. Right-click the test client that you just created and select **Secure Web Service Client** > **Add Stand Alone Security Token** (assuming that the security that has been enabled in the service is a user name security token for authentication purposes). If the test client is not in the **Clients** folder, restart Rational Application Developer.
20. In the WS-Security Wizard, select the **Username Token** token type and the **com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler** callback handler. You will specify the login credentials on the next page of the wizard.
21. Click **Next**.
22. Enter a valid User ID and Password for the service and click **Finish**.
23. Save the changes and restart the test client.
24. Optional: To test your service by using the test client, in the **Web Services Test Client** tab that opens, select a method, enter any input number, and click **Invoke**. The **Result** pane displays the result.
25. Optional: To invoke the test client without using the assembly tool, use a Web browser and enter the URL `https://`*`hostname`*`:`*`WASSSLChannelPort`*`/`*`applicationName`*`/sample`*`serviceName`*`PortTypeProxy/TestClient.jsp` in the address field.

    hostname
    The name of the server hosting your test client (localhost, for example).

    WASSSLChannelPort
    The port number used by the SSL transport channel (9443 is the default).

    applicationName
    The name of your test client.

    serviceName
    The name of the service that you want to invoke.

Generating sample key stores, keys, and certificates

Any type of encryption, including SSL, requires that you generate key stores on both the client and server sides. Key stores contain keys that sign and encrypt messages. Use the sample key stores in this topic to test your encryption or run the script by using the Java keytool utility to generate the sample key stores yourself.

Note: To keep this example simple, the sample key stores were generated by using self-signed certificates instead of certificates signed by a Certificate Authority (CA).

Use these sample key stores only for testing purposes. Implement your own Public Key Infrastructure (PKI) in a real production environment.

You can generate key stores by using different methods. The sample key stores in this topic were generated by using `keytool`, the Java command-line tool, and the following script. You can also use iKeyman, the IBM Key Management tool, which is a graphical user interface tool for managing key stores and keys. It is installed with IBM WebSphere Application Server in *install-dir*`/WebSphere/AppServer/bin/ikeyman.bat`.

SSL Encryption

- Client key store (clientsslkeys.jceks):
  - Client private key
- Client trust store (clientssltrusts.jceks):
  - Server self-signed certificate with only a public key
- Server key store (serversslkeys.jceks):
  - Server private key
- Server trust store (serverssltrusts.jceks):
  - Client self-signed certificate with only a public key

The sample key stores also have the following characteristics.

Table 1. Characteristics for client key store `clientsslkeys.jceks`

Field	Value
Filename	`C:\sampleks\clientsslkeys.jceks`
Storepass	password
Storetype	JCEK
Distinguished Name	CN=wasclient, O=IBM, C=US
Keypass	password
Alias	wasclient

Table 2. Characteristics for client trust store `clientssltrusts.jceks`

Field	Value
Filename	`C:\sampleks\clientssltrusts.jceks`
Storepass	password
Storetype	JCEK
Certificate file	wasserver.cert

Table 3. Characteristics for server key store `serversslkeys.jceks`

Field	Value
Filename	`C:\sampleks\serversslkeys.jceks`
Storepass	password
Storetype	JCEK
Distinguished Name	CN=wasserver, O=IBM, C=US

Table 3. Characteristics for server key store serversslkeys.jceks (continued)

Field	Value
Keypass	password
Alias	wasserver

Table 4. Characteristics for server trust store serverssltrusts.jceks

Field	Value
Filename	C:\sampleks\serverssltrusts.jceks
Storepass	password
Storetype	JCEK
Certificate file	wasclient.cert

Creating the client key store

By using Java keytool, you can generate the client key store with the following script.

```
REM  Generate the client key store and private key
keytool -genkey -v -alias wasclient -keypass password -keystore
clientsslkeys.jceks -storepass password -storetype jceks -dname
"CN=wasclient, O=IBM, C=US" -keyalg "RSA"
REM  Generate the client self-signed certificate
keytool -export -v -alias wasclient -file wasclient.cert -rfc
-keystore
clientsslkeys.jceks -storepass password -storetype jceks
```

Creating the server key store

By using Java keytool, you can generate the server key store with the following script.

```
REM  Generate the server key store and private key
keytool -genkey -v -alias wasserver -keypass password -keystore
serversslkeys.jceks -storepass password -storetype jceks -dname
"CN=wasserver, O=IBM, C=US" -keyalg "RSA"
REM  Generate the server self-signed certificate
keytool -export -v -alias wasserver -file wasserver.cert -rfc
-keystore
serversslkeys.jceks -storepass password -storetype jceks
```

Creating the client trust store

By using Java keytool, you can generate the client trust store with the following script.

```
REM  Import the server certificate in the client truststore
keytool -import -v -noprompt -alias wasserver -file wasserver.cert -
keystore clientssltrusts.jceks -storepass password -storetype jceks
```

Creating the server trust store

By using Java keytool, you can generate the server trust store with the following script.

```
REM  Import the client certificate in the server truststore
keytool -import -v -noprompt -alias wasclient -file wasclient.cert -
keystore serverssltrusts.jceks -storepass password -storetype jceks
```

Generating a test client with SSL enabled

Use this task to generate a test client for a service that enables data confidentiality by using Secure Sockets Layer (SSL) encryption.

Before you begin

- Create a service
- Configure an assembly tool as described in "Configuring an assembly tool with IBM InfoSphere Information Server" on page 7
- Disable the unsecured service as described in "Disabling an unsecured service by using the IBM InfoSphere Information Server console" on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in "Exporting a service by using the WebSphere Application Server console" on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in "Importing an unsecured service EAR file into an assembly tool" on page 9
- Add confidentiality to the service as described in "Adding data confidentiality by using SSL encryption" on page 22
- Export the EAR file that contains the service as described in "Exporting a secured service EAR file from an assembly tool" on page 27
- Redeploy the service EAR file as described in "Redeploying a secured service with the IBM WebSphere Application Server console" on page 28
- Enable the secured service as described in "Enabling a secured service by using the IBM InfoSphere Information Server console" on page 28

About this task

In one part of this task, you will create a new SSL Java Secure Socket Extension (JSSE) repertoire by using IBM WebSphere Application Server. To create this new repertoire, you will need to use the sample key stores from "Generating sample key stores, keys, and certificates" on page 33.

Procedure

1. Log in to the IBM WebSphere Application Server administrative console.
2. In the navigation pane, expand **Security** and click **SSL**.
3. In the SSL configuration repertoires window, click **New JSSE repertoire**.
4. In the **Configuration** tab:
   a. Type an alias name for the repertoire. For example, `WASClientSSL`.
   b. Do not select the **Client Authentication** check box. The SSL repertoire acts as an SSL client and not as a server, so there is not a client to authenticate.
   c. For the security level, select **HIGH**.
   d. Optional: Select ciphers from the **Cipher suites** list if you want to override the predefined ciphers that encrypt the SSL according to the security level that you select.
   e. Optional: Select the **Cryptographic token** check box.
   f. Optional: Select a predefined or custom JSSE provider.
   g. Select the **SSL** protocol.
   h. Enter the location, name, and password for the key file and select a format. Use the key file located in the sample key stores reference topic.

i. Enter the location, name, and password for the trust file and select a format. Use the trust file located in the sample key stores reference topic.
j. Click **OK**.

5. Save the changes and restart IBM WebSphere Application Server.
6. In the **Servers** tab of IBM Rational Application Developer, right-click anywhere inside the window and select **New** > **Server**.
7. In the **Define a New Server** page of the New Server window, enter a name for the server host, select **IBM** > **WebSphere v6.1 Server** or **WebSphere v7.0 Server** as the server type, and the server runtime environment. If you followed the task to configure IBM Rational Application Developer with IBM InfoSphere Information Server, the correct runtime environment is **Information Server WAS v6.1** or **Information Server WAS v7.0**.
8. Click **Next** and ensure that the information on the following page is correct, including the profile name and authentication information.
9. Click **Next**.
10. In the **Add and Remove Projects** page, select your secured EAR file from the **Available projects** field and click **Add >** to add it to the list of configured projects.
11. Click **Next**, and then click **Finish**.
12. In the **Servers** tab, select the server that you created and click the **Start the server** icon.
13. Select **File** > **New** > **Project** > **Dynamic Web Project**.
14. In the New Dynamic Web Project window, enter a project name (MyAppClient, for example), select **Information Server WAS v6.1** or **Information Server WAS v7.0** as the target runtime, select **Add project to an EAR**, and enter an enterprise archive (EAR) project name (MyAppClient.ear, for example) for the EAR file that will contain the client code.
15. Click **Finish**.
16. Select **File** > **New** > **Other**.
17. In the **New** window, expand **Web Services**, select **Web Service Client**, and click **Next**.
18. In the **Web Services** page of the Web Service Client window, enter the Web Services Description Language (WSDL) URL (http://localhost:9080/wisd/MyApp/MyService?wsdl, for example) in the **Service definition** field.
19. Select **Test client** as the level of automation.
20. In the Configuration panel, verify these settings:

    Server Websphere v6.1 Server or Websphere v7.0 Server

    Web service runtime
    IBM WebSphere JAX-RPC

    Client project
    MyAppClient

    Client EAR project
    MyAppClient.ear
21. Click **Finish**.
22. In the **Web Service Client Test** page, select the **Web service sample JSPs** test facility and the method that you want to test. Use the default settings for the rest of this page and click **Finish**.

23. In the **Project Explorer** of IBM Rational Application Developer, expand **JSR-109 Web Services** > **Clients** and double-click the generated test client to edit the deployment descriptor. Restart Rational Application Developer if the test client is not in the folder.
24. Click the **WS Binding** tab in the Web Deployment Descriptor window and expand the **Port Qualified Name Binding Details** section.
25. In the **HTTP SSL configurationName** field, enter the name of the client SSL repertoire. For example, *name2Node01/WASClientSSL*.
26. Save the changes and restart the test client.
27. Optional: To test your service by using the test client, in the **Web Services Test Client** tab that opens, select a method, enter any input number, and click **Invoke**. The **Result** pane displays the result.
28. Optional: To invoke the test client without using the assembly tool, use a Web browser and enter the URL `https://`*`hostname`*`:`*`WASSSLChannelPort`*`/`*`applicationName`*`/sample`*`serviceName`*`PortTypeProxy/TestClient.jsp` in the address field.

    hostname
    The name of the server hosting your test client (localhost, for example).

    WASSSLChannelPort
    The port number used by the SSL transport channel (9443 is the default).

    applicationName
    The name of your test client.

    serviceName
    The name of the service that you want to invoke.

Example

Avoid these errors when you enable SSL by using IBM WebSphere Application Server:

- If you invoke a secured SSL service by using HTTP instead of HTTPS, the following exception is returned:

  ```
  exception: WSWS3499W: Redirected new location:
  http://localhost:9080/wisd/MyApp/MyService
  ```

- If you invoke a secured SSL service by using a test client not configured for SSL, the following exception is returned:

  ```
  exception: WSWS3713E: Connection to the remote host
  localhost failed. Received the following error:
  Handshake terminated SSL engine: CLOSED
  ```

- If you invoke a secured SSL service by using a test client that is not properly configured for SSL (the client trust store is not properly configured), the following exception is returned:

  ```
  exception: com.ibm.wsspi.channel.framework.exception
  ChannelException:com.ibm.wsspi.channel.framework.exception.
  ChannelException: Invalid trust file name of null
  ```

Tracing SOAP messages

Trace SOAP messages exchanged between client and server to troubleshoot problems with a service. Use the TCP/IP Monitor tool in IBM Rational Application Developer to trace SOAP messages.

Before you begin

- Create a service
- Configure an assembly tool as described in “Configuring an assembly tool with IBM InfoSphere Information Server” on page 7
- Disable the unsecured service as described in “Disabling an unsecured service by using the IBM InfoSphere Information Server console” on page 8
- Export the service to create an enterprise archive (EAR) file of the service as described in “Exporting a service by using the WebSphere Application Server console” on page 8
- Import the service EAR file into an assembly tool if you want to use the tool to enable security features as described in “Importing an unsecured service EAR file into an assembly tool” on page 9
- Secure a service as described in “Securing a service” on page 10
- Export the EAR file that contains the service as described in “Exporting a secured service EAR file from an assembly tool” on page 27
- Redeploy the service EAR file as described in “Redeploying a secured service with the IBM WebSphere Application Server console” on page 28
- Enable the secured service as described in “Enabling a secured service by using the IBM InfoSphere Information Server console” on page 28

Procedure

1. Select **Window** > **Show View** > **Other**.
2. In the Show View window, expand **Debug** , select **TCP/IP Monitor** and click **OK**.
3. Right-click inside the TCP/IP Monitor window and select **Properties**.
4. Click **Add** and enter or select the following information in the New Monitor window:
   a. Enter an unused local monitoring port number. For example, *13557*. The client will send SOAP requests to this port.
   b. Enter the name of the host computer that is running the server. For example, if the client and server are on the same computer, the host name is `localhost`.
   c. Enter the port number to monitor, which is the port that the client would send requests if TCP/IP monitoring were not enabled. For example, the default port number is `9080` for IBM WebSphere Application Server.
   d. Select either **HTTP** or **TCP/IP** type of monitoring. **TCP/IP** shows the complete HTTP messages, which includes the HTTP headers and the SOAP messages. **HTTP** shows only the SOAP messages.
   e. Click **OK**.
5. In the TCP/IP Monitor Preferences window that opens, select the monitor and click **Start** to begin monitoring.
6. In the **Project Explorer** expand**JSR-109 Web Services** > **Clients**, right-click the test client, and select **Open**.
7. Select the **getEndpoint()** method and click **Invoke**. Copy the endpoint URL returned.
8. Select the **setEndpoint()** method, paste the endpoint URL previously returned by the **getEndpoint()** method and change the port number to point to the TCP/IP Monitor port number. Click **Invoke**. For example, if the host name is localhost and the local port number is 13557, the endpoint is `http://localhost:13557/wisd/MyApp/MyService`.

9. Invoke your service with the test client by using one of the following methods:
   - In the **Web Services Test Client**, select a method that is defined in the service, enter any input number, and click **Invoke**.
   - To invoke the test client without using the assembly tool, use a Web browser and enter the URL `https://`*`hostname`*`:`*`WASSSLChannelPort`*`/`*`applicationName`*`/sample`*`serviceName`*`PortTypeProxy/TestClient.jsp` in the address field.

     hostname
     The name of the server hosting your test client (localhost, for example).

     WASSSLChannelPort
     The port number used by the SSL transport channel (9443 is the default).

     applicationName
     The name of your test client.

     serviceName
     The name of the service that you want to invoke.
10. After you invoke the service, analyze the HTTP and SOAP messages in the TCP/IP Monitor window to troubleshoot problems that occur in the service.

Security technologies comparison

To choose a technology that fits your requirements, you can compare security technologies by using two parameters: degree of security provided and degree of difficulty to set up.

Typically a technology is more difficult to implement when it is based on public key cryptography, which requires the setup of a Public Key Infrastructure (PKI). This is the case of any type of digital signing and encryption. However, these technologies usually offer a better level of security than other technologies that might require less infrastructure work.

Table 5. Authentication methods and their security levels

Authentication method	Level	Degree of security provided	Degree of difficulty to set up	Notes
HTTP basic authentication	Transport	low	low	User name and password are sent in cleartext (base64) over the network. Used only in secure networks (HTTPS or intranet).
HTTP digest authentication (not supported by IBM WebSphere Application Server)	Transport	medium	medium	User name and password are encrypted.

Table 5. Authentication methods and their security levels (continued)

Authentication method	Level	Degree of security provided	Degree of difficulty to set up	Notes
User name token	Message	low	medium	User name and password are sent in cleartext over the network. Used only in secure networks (HTTPS, intranet or XML Encryption).
X.509 certificate token	Message	high	medium	Requires a PKI.
Kerberos ticket token	Message	high	high	Requires a Kerberos infrastructure.

Table 6. Authorization method and its security level

Authorization method	Level	Degree of security provided	Degree of difficulty to set up	Notes
J2EE declarative security constraints	Application	medium	medium	Does not require any security infrastructure.

J2EE declarative security constraints are easy to configure (only require changes to the J2EE deployment descriptors) and at the same time provide a good level of access control. They are often the starting point towards securing services.

Table 7. Confidentiality method and its security level

Confidentiality method	Level	Degree of security provided	Degree of difficulty to set up	Notes
SSL	Transport	medium	medium	Requires a PKI. Confidentiality is provided from network point to network point.

SSL and Transport Layer Security (TLS) provide additional security features on top of encryption, including authentication, data protection, and cryptographic token support for secure HTTP connections.

Encryption is a computation-intensive operation that can affect performance. It also increases management and administrative overhead. Therefore, they should be used after due consideration has been given to their applicability.

Table 8. Digital integrity method and its security level

Data integrity method	Level	Degree of security provided	Degree of difficulty to set up	Notes
XML Digital Signature	Message	high	medium	Requires a PKI

Digital signature is a computation-intensive operation that can lead to performance penalties. It also increases management and administrative overhead. Therefore, they should be used after due consideration has been given to their applicability.

Security technologies and bindings

Not all security technologies are available for all bindings.

For some bindings, only a subset of the complete list of existing security technologies might be available (or applicable). For example, none of the Web services security technologies is applicable for the EJB binding. The following table summarizes the security technologies available per binding.

Table 9. Security technologies and bindings

	Security technology	EJB	SOAP over HTTP	SOAP over JMS	Text over JMS
Authentication	JAAS/EJB	Available			
Authentication	HTTP Basic		Available		
Authentication	HTTP Digest		Not supported by IBM WebSphere Application Server 6.0 and later		
Authentication	Web services security username token		Available	Available	
Authentication	Other Web services security (Kerberos, X509, LTPA, SAML)		Available	Available	
Authorization	J2EE	Available	Available	Available	Available
Data confidentiality	HTTPS and SSL	Available	Available	Available	Available

Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible. The installation program installs the following product modules and components:

- IBM InfoSphere Business Glossary
- IBM InfoSphere Business Glossary Anywhere
- IBM InfoSphere DataStage
- IBM InfoSphere FastTrack
- IBM InfoSphere Information Analyzer
- IBM InfoSphere Information Services Director
- IBM InfoSphere Metadata Workbench
- IBM InfoSphere QualityStage™

For information about the accessibility status of IBM products, see the IBM product accessibility information at http://www.ibm.com/able/product_accessibility/index.html.

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to set display preferences in your browser. It also allows you to use screen readers and other assistive technologies to access the documentation.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Accessing product documentation

Documentation is provided in a variety of locations and formats, including in help that is opened directly from the product client interfaces, in a suite-wide information center, and in PDF file books.

The information center is installed as a common service with IBM InfoSphere Information Server. The information center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open the information center from the installed product or from a Web browser.

Accessing the information center

You can use the following methods to open the installed information center.

- Click the **Help** link in the upper right of the client interface.

  Note: From IBM InfoSphere FastTrack and IBM InfoSphere Information Server Manager, the main Help item opens a local help system. Choose **Help > Open Info Center** to open the full suite information center.
- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in Web clients.
- Use a Web browser to access the installed information center even when you are not logged in to the product. Enter the following address in a Web browser: http://host_name:port_number/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.home.doc/ic-homepage.html. The host_name is the name of the services tier computer where the information center is installed, and port_number is the port number for InfoSphere Information Server. The default port number is 9080. For example, on a Microsoft® Windows® Server computer named iisdocs2, the Web address is in the following format: http://iisdocs2:9080/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.nav.doc/dochome/iisinfsrv_home.html.

A subset of the information center is also available on the IBM Web site and periodically refreshed at http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/index.jsp.

Obtaining PDF and hardcopy documentation

- A subset of the PDF file books are available through the InfoSphere Information Server software installer and the distribution media. The other PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1.
- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

Providing feedback about the documentation

You can send your comments about documentation in the following ways:

- Online reader comment form: www.ibm.com/software/data/rcf/
- E-mail: comments@us.ibm.com

Links to non-IBM Web sites

This information center may provide links or references to non-IBM Web sites and resources.

IBM makes no representations, warranties, or other commitments whatsoever about any non-IBM Web sites or third-party resources (including any Lenovo Web site) that may be referenced, accessible from, or linked to any IBM site. A link to a non-IBM Web site does not mean that IBM endorses the content or use of such Web site or its owner. In addition, IBM is not a party to or responsible for any transactions you may enter into with third parties, even if you learn of such parties (or use a link to such parties) from an IBM site. Accordingly, you acknowledge and agree that IBM is not responsible for the availability of such external sites or resources, and is not responsible or liable for any content, services, products or other materials on or available from those sites or resources.

When you access a non-IBM Web site, even one that may contain the IBM-logo, please understand that it is independent from IBM, and that IBM does not control the content on that Web site. It is up to you to take precautions to protect yourself from viruses, worms, trojan horses, and other potentially destructive programs, and to protect your information as you deem appropriate.

Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS^{Link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS^{Link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.

Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 10. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at www.ibm.com/account/
Training and certification	You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at http://www.ibm.com/software/sw-training/
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Providing feedback

The following table describes how to provide feedback to IBM about products and product documentation.

Table 11. Providing feedback to IBM

Type of feedback	Action
Product feedback	You can provide general product feedback through the Consumability Survey at www.ibm.com/software/data/info/consumability-survey

Table 11. Providing feedback to IBM (continued)

Type of feedback	Action
Documentation feedback	To comment on the information center, click the Feedback link on the top right side of any topic in the information center. You can also send comments about PDF file books, the information center, or any other documentation in the following ways: • Online reader comment form: www.ibm.com/software/data/rcf/ • E-mail: comments@us.ibm.com

Index

Printed in USA

SC19-3484-00