Composed February 22, 2011

◆ ◆ ◆ CHAPTER 3

Administering Message Security

This chapter provides information and procedures on configuring the message layer security for web services in the GlassFish Server environment.

Note – Message security (JSR 196) is supported only in the Full Platform Profile of GlassFish Server, not in the Web Profile.

The following topics are addressed here:

Some of the material in this chapter assumes a basic understanding of security and web services concepts. For more information about security, see "About System Security in GlassFish Server" on page 13.

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

About Message Security in GlassFish Server

Message security enables a server to perform end-to-end authentication of web service invocations and responses at the message layer. Security information is inserted into messages so that it travels through the networking layers and arrives with the intact message at the message destination(s). Message security differs from transport layer security in that message security can be used to decouple message protection from message transport so that messages remain protected after transmission.

Web services deployed on GlassFish Server are secured by binding SOAP layer message security providers and message protection policies to the containers in which the applications are deployed, or to web service endpoints served by the applications. SOAP layer message security functionality is configured in the client-side containers of GlassFish Server by binding SOAP layer message security providers and message protection policies to the client containers or to the portable service references declared by client applications.

Message-level security can be configured for the entire GlassFish Server or for specific applications or methods. Configuring message security at the application level is discussed in the *Oracle GlassFish Server 3.1 Application Development Guide*.

The following topics are addressed here:

Security Tokens and Security Mechanisms

WS-Security is a specification that provides a communications protocol for applying security to web services. The security mechanisms implement the specification. Web Services Interoperability Technologies (WSIT) implements WS-Security so as to provide interoperable message content integrity and confidentiality, even when messages pass through intermediary nodes before reaching their destination endpoint. WS-Security as provided by WSIT is in addition to existing transport-level security, which can still be used.

The Simple Object Access Protocol (SOAP) layer message security providers installed with GlassFish Server can be used to employ username/password and X.509 certificate security tokens to authenticate and encrypt SOAP web services messages.

- **Username Tokens.** GlassFish Server uses username tokens in SOAP messages to authenticate the message sender. The recipient of a message containing a username token (within embedded password) validates that the message sender is authorized to act as the user (identified in the token) by confirming that the sender knows the password of the user.

  When using a username token, a valid user database must be configured on GlassFish Server.
- **Digital Signatures.** GlassFish Server uses XML digital signatures to bind an authentication identity to message content. Clients use digital signatures to establish their caller identity. Digital signatures are verified by the message receiver to authenticate the source of the message content (which might be different from the sender of the message.)

  When using digital signatures, valid keystore and truststore files must be configured on GlassFish Server.

- **Encryption.** The purpose of encryption is to modify the data so that it can only be understood by its intended audience. This is accomplished by substituting an encrypted element for the original content. When based on public key cryptography, encryption can be used to establish the identity of the parties who are authorized to read a message.

  When using encryption, a Java Cryptography Extension (JCE) provider that supports encryption must be installed.

Authentication Providers

The *authentication layer* is the message layer on which authentication processing must be performed. GlassFish Server enforces web services message security at the SOAP layer. The types of authentication that are supported include the following:

- Sender authentication, including username-password authentication
- Content authentication, including XML digital signatures

GlassFish Server invokes *authentication providers* to process SOAP message layer security. The message security providers provide information such as the type of authentication that is required for the request and response messages. The following message security providers are included with GlassFish Server:

- **Client-side Provider.** A client-side provider establishes (by signature or username/password) the source identity of request messages and/or protects (by encryption) request messages such that they can only be viewed by their intended recipients. A client-side provider also establishes its container as an authorized recipient of a received response (by successfully decrypting it) and validates passwords or signatures in the response to authenticate the source identity associated with the response. Client-side providers configured in GlassFish Server can be used to protect the request messages sent and the response messages received by server-side components (servlets and EJB components) acting as clients of other services.

  The *default client provider* is used to identify the client—side provider to be invoked for any application for which a specific client provider has not been bound.
- **Server-side Provider.** A server-side provider establishes its container as an authorized recipient of a received request (by successfully decrypting it), and validates passwords or signatures in the request to authenticate the source identity associated with the request. A server-side provider also establishes (by signature or username/password) the source identity of response messages and/or protects (by encryption) response messages such that they can only be viewed by their intended recipients. Server-side providers are only invoked by server-side containers.

The *default server provider* is used to identify the server—side provider to be invoked for any application for which a specific server provider has not been bound.

Message Protection Policies

A *request policy* defines the authentication policy requirements associated with request processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature. The *response policy* defines the authentication policy requirements associated with response processing performed by the authentication provider.

Message protection policies are defined for request message processing and response message processing. The policies are expressed in terms of requirements for source and/or recipient authentication. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

- **Source Authentication Policy.** A source authentication policy represents a requirement that the identity of the entity that sent a message or that defined the content of a message be established in the message such that it can be authenticated by the message receiver.
- **Recipient Authentication Policy.** A recipient authentication policy represents a requirement that the message be sent such that the identity of the entities that can receive the message can be established by the message sender.

Request and response message protection policies are defined when a security provider is configured into a container. Application-specific message protection policies (at the granularity of the web service port or operation) can also be configured within the GlassFish Server deployment descriptors of the application or application client. In any situation where message protection policies are defined, the request and response message protection policies of the client must be equivalent t) the request and response message protection policies of the server. For more information about defining application-specific message protection policies, see Chapter 4, "Securing Applications," in *Oracle GlassFish Server 3.1 Application Development Guide*

Application-Specific Web Services Security

Application-specific web services security functionality is configured (at application assembly) by defining the `message-security-binding` elements in the GlassFish Server deployment descriptors of the application. These `message-security-binding` elements are used to associate a specific security provider or message protection policy with a web service endpoint or service reference, and might be qualified so that they apply to a specific port or method of the corresponding endpoint or referenced service.

For information about defining application-specific message protection policies, see Chapter 4, "Securing Applications," in *Oracle GlassFish Server 3.1 Application Development Guide*.

Message Security Administration

When GlassFish Server is installed, SOAP layer message security providers are configured in the client and server-side containers of GlassFish Server, where they are available for binding for use by the containers, or by individual applications or clients deployed in the containers. During installation, the default providers are configured with a simple message protection policy that, if bound to a container, or to an application or client in a container, would cause the source of the content in all request and response messages to be authenticated by XML digital signature.

GlassFish Server administrative interfaces can be used as follows:

- To modify the message protection policies enforced by the providers
- To bind the existing providers for use by the server-side containers of GlassFish Server
- To create new security provider configurations with alternative message protection policies

Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container. If you want web services security to protect all web services applications deployed on GlassFish Server. See "Enabling Message Security for Application Clients" on page 78.

By default, message layer security is disabled on GlassFish Server. To configure message layer security for the GlassFish Server see "Enabling Default Message Security Providers for Web Services" on page 71.

In most cases, you must restart GlassFish Server after performing administrative tasks. This is especially true if you want the effects of the administrative change to be applied to applications that were already deployed on GlassFish Server at the time the operation was performed.

Message Security Tasks

The general implementation tasks for message security include some or all of the following:

1. If you are using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configuring a JCE provider
2. If you are using a username token, verifying that a user database is configured for an appropriate realm

   When using a username/password token, an appropriate realm must be configured and a user database must be configured for the realm.
3. Managing certificates and private keys, if necessary
4. Enabling the GlassFish Server default providers
5. Configuring new message security providers

Message Security Roles

In GlassFish Server, the administrator and the application deployer are expected to take primary responsibility for configuring message security. In some situations, the application developer might also contribute.

System Administrator

The system administrator is responsible for the following message security tasks:

- Administering server security settings and certificate databases
- Administering keystore and truststore files
- Configuring message security providers on GlassFish Server
- Turning on message security
- (If needed) Installing the samples server

Application Deployer

The application deployer is responsible for the following message security tasks:

- Specifying (at application reassembly) any required application-specific message protection policies if such policies have not already been specified by the developer/assembler.
- Modifying GlassFish Server deployment descriptors to specify application-specific message protection policies information (message-security-binding elements) to web service endpoint and service references.

Application Developer/Assembler

The application developer/assembler is responsible for the following message security tasks:

- Determining if an application-specific message protection policy is required by the application

  If so, the developer ensures that the required policy is specified at application assembly time.
- Specifying how web services should be set up for message security

  Message security can be set up by the administrator so that all web services are secured, or by the application deployer when the security provider or protection policy bound to the application must be different from that bound to the container.
- Turning on message security if authorized to do so by the administrator

Sample Application for Web Services

GlassFish Server includes a sample application named `xms`. The `xms` application features a simple web service that is implemented by both a Java EE EJB endpoint and a Java servlet endpoint. Both endpoints share the same service endpoint interface. The service endpoint

interface defines a single operation, `sayHello`, which takes a string argument, and returns a `String` composed by pre-pending `Hello` to the invocation argument.

The `xms` sample application is provided to demonstrate the use of GlassFish Server WS-Security functionality to secure an existing web services application. The instructions which accompany the sample describe how to enable the WS-Security functionality of GlassFish Server such that it is used to secure the `xms` application. The sample also demonstrates the binding of WS-Security functionality directly to the application as described in "Application-Specific Web Services Security" on page 68 application.

For information about compiling, packaging, and running the `xms` sample application, Chapter 4, "Securing Applications," in *Oracle GlassFish Server 3.1 Application Development Guide*.

The `xms` sample application is installed in the following directory:
as-install`/samples/webservices/security/ejb/apps/xms/`

Enabling Default Message Security Providers for Web Services

By default, message security is disabled on GlassFish Server. Default message security providers have been created, but are not active until you enable them. After the providers have been enabled, message security is enabled.

The following topics are addressed here:

▼ To Enable a Default Server Provider

To enable message security for web services endpoints deployed in GlassFish Server, you must specify a security provider to be used by default on the server side. If you enable a default provider for message security, you also need to enable providers to be used by clients of the web services deployed in GlassFish Server.

1 **Specify the default server provider by using the `set(1)` subcommand.**

Use the following syntax:

```
asadmin  set --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

2 **To apply your changes to applications that are already running, restart GlassFish Server.**

See "To Restart a Domain" in *Oracle GlassFish Server 3.1 Administration Guide*.

▼ To Enable a Default Client Provider

To enable message security for web service invocations originating from deployed endpoints, you must specify a default client provider. If you enabled a default client provider for GlassFish Server, you must ensure that any services invoked from endpoints deployed in GlassFish Server are compatibly configured for message layer security.

1. **Specify the default client provider by using the `set(1)` subcommand.**

   Use the following syntax:

   ```
   asadmin  set --port admin-port
   server-config.security-service.message-security-config.SOAP.
   default_client_provider=ClientProvider
   ```

2. **To apply your changes to applications that are already running, restart GlassFish Server.**

   See "To Restart a Domain" in *Oracle GlassFish Server 3.1 Administration Guide*.

Configuring Message Protection Policies

Message protection policies are defined for request message processing and response message processing. The policies are expressed in terms of requirements for source and/or recipient authentication. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

The following topics are addressed here:

Message Protection Policy Mapping

The following table shows message protection policy configurations and the resulting message security operations performed by the WS-Security SOAP message security providers for that configuration.

TABLE 3–1 Message Protection Policy Mapping to WS-Security SOAP Operations

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="sender"	The message contains a `wsse:Security` header that contains a `wsse:UsernameToken` (with password).

TABLE 3–1 Message Protection Policy Mapping to WS-Security SOAP Operations *(Continued)*

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="content"	The content of the SOAP message Body is signed. The message contains a `wsse:Security` header that contains the message Body signature represented as a `ds:Signature`.
auth-source="sender" *auth-recipient="before-content"* OR *auth-recipient="after-content"*	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains a `wsse:UsernameToken (with password)` and an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" *auth-recipient="before-content"*	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The `xenc:EncryptedData` is signed. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" *auth-recipient="after-content"*	The content of the SOAP message Body is signed, then encrypted, and then replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-recipient="before-content" OR *auth-recipient="after-content"*	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
No policy specified.	No security operations are performed by the modules.

▼ To Configure the Message Protection Policies for a Provider

Typically, you would not reconfigure a provider. However, if needed for your situation, you can modify a provider's message protection policies by changing provider type, implementation class, and provider-specific configuration properties. To understand the results of different combinations, see Table 3–1.

Use the `set(1)` subcommand to set the response policy, then replace the word `request` in the following commands with the word `response`.

1. **Add a request policy to the client and set the authentication source by using the `set(1)` subcommand.**

   For example:

   ```
   asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ClientProvider.request-policy.auth_source=[sender | content]
   ```

2. **Add a request policy to the server and set the authentication source by using the `set` subcommand.**

   For example:

   ```
   asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ServerProvider.request-policy.auth_source=[sender | content]
   ```

3. **Add a request policy to the client and set the authentication recipient by using the `set` subcommand:**

   For example:

```
asadmin> set server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=[before-content | after-content]
```

4. **Add a request policy to the server and set the authentication recipient by using the `set` subcommand:**

   For example:

```
asadmin> set server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=[before-content | after-content]
```

Setting the Request and Response Policy for the Application Client Configuration

The request and response policies define the authentication policy requirements associated with request and response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

To achieve message security, the request and response policies must be enabled on both the server and client. When configuring the policies on the client and server, make sure that the client policy matches the server policy for request/response protection at application-level message binding.

To set the request policy for the application client configuration, modify the GlassFish Server–specific configuration for the application client container as described in "Enabling Message Security for Application Clients" on page 78.

EXAMPLE 3–1 Message Security Policy Setting for Application Clients

In the application client configuration file, the `request-policy` and `response-policy` elements are used to set the request policy, as shown in the following code snippet. (Additional code in the snippet is provided as illustration and might differ slightly in your installation. Do not change the additional code.)

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="as-install/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

Valid values for `auth-source` include `sender` and `content`. Valid values for `auth-recipient` include `before-content` and `after-content`. A table describing the results of various combinations of these values can be found in "Configuring Message Protection Policies" on page 72.

To not specify a request or response policy, leave the element blank, for example:

```
<response-policy/>
```

Administering Non-default Message Security Providers

The following topics are addressed here:

▼ To Create a Message Security Provider

Use the `create–message–security–provider` subcommand in remote mode to create a new message provider for the security service. If the message layer does not exist, the message layer is created, and the provider is created under it.

1. **Ensure that the server is running.**

   Remote subcommands require a running server.

2. **Create the message security provider by using the `create-message-security-provider(1)` subcommand.**

   Information about properties for this subcommand is included in this help page.

3. **(Optional) If needed, restart the server.**

   Some properties require server restart. See "Configuration Changes That Require Server Restart" in *Oracle GlassFish Server 3.1 Administration Guide*. If your server needs to be restarted, see "To Restart a Domain" in *Oracle GlassFish Server 3.1 Administration Guide*.

Example 3–2 Creating a Message Security Provider

This example creates the new message security provider `mySecurityProvider`.

```
asadmin> create-message-security-provider
--classname com.sun.enterprise.security.jauth.ClientAuthModule
--providertype client mySecurityProvider
Command create-message-security-provider executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help create–message–security–provider` at the command line.

To List Message Security Providers

Use the `list–message–security–providers` subcommand in remote mode to list the message providers for the security layer.

1. **Ensure that the server is running.**
   Remote subcommands require a running server.
2. **List the message security providers by using the `list-message-security-providers(1)` subcommand.**

Example 3–3 Listing Message Security Providers

This example lists the message security providers for a message layer.

```
asadmin> list-message-security-providers --layer SOAP
XWS_ClientProvider
ClientProvider
XWS_ServerProvider
ServerProvider
Command list-message-security-providers executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help list–message–security–providers` at the command line.

To Update a Message Security Provider

1. **Ensure that the server is running.**
   Remote subcommands require a running server.
2. **List the message security providers by using the `list-message-security-providers(1)` subcommand.**
3. **Modify the values for the specified message security provider by using the `set(1)` subcommand.**
   The message security provider is identified by its dotted name.

To Delete a Message Security Provider

Use the `delete-message-security-provider` subcommand in remote mode to remove a message security provider.

1. **Ensure that the server is running.**
   Remote subcommands require a running server.

2. **List the message security providers by using the `list-message-security-providers(1)` subcommand.**

3. **Delete the message security provider by using the `delete-message-security-provider(1)` subcommand.**

Example 3–4 Deleting a Message Security Provider

This example deletes the `myServerityProvider` message security provider.

```
asadmin> delete-message-security-provider --layer SOAP myServerityProvider
Command delete-message-security-provider executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help delete–message–security–provider` at the command line.

Enabling Message Security for Application Clients

The message protection policies of client providers must be configured such that they are equivalent to the message protection policies of the server-side providers they will be interacting with. This is already the situation for the providers configured (but not enabled) when GlassFish Server is installed.

To enable message security for client applications, modify the GlassFish Server specific configuration for the application client container. The process is analogous to the process in "Configuring Message Protection Policies" on page 72.

Additional Information About Message Security

For additional information about message security, see the following documentation:

- Chapter 24, "Introduction to Security in the Java EE Platform," in *The Java EE 6 Tutorial*
- Chapter 4, "Securing Applications," in *Oracle GlassFish Server 3.1 Application Development Guide*