BEA AquaLogic Service Bus™

Security Guide

Version: 2.6
Document Revised: January 2007

Contents

CHAPTER 1

Introduction

This document describes how to use standard technologies such as SSL and Web Services Security along with BEA proprietary technologies to ensure that only authorized users can access resources in an AquaLogic Service Bus domain.

Document Audience

This document is intended for the following audiences:

- Application Architects—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate AquaLogic Service Bus security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge, security technologies and tools.
- Security Developers—Developers who focus on defining the system architecture and infrastructure for security products that integrate into AquaLogic Service Bus and on developing custom security providers for use with AquaLogic Service Bus. They work with Application Architects to ensure that the security architecture is implemented according to design and that no security holes are introduced, and work with Server Administrators to ensure that security is properly configured. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing (AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX), and working knowledge of WebLogic Server, AquaLogic Service Bus, and security provider functionality.

- Application Developers—Developers who are Java programmers that focus on developing client applications, adding security to Web applications and Enterprise JavaBeans (EJBs), and working with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working knowledge of Java (including J2EE components such as servlets/JSPs and JSEE) and Java security.
- Server Administrators—Administrators work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web services, Web application and EJB security, Public Key security, SSL, and Security Assertion Markup Language (SAML).
- Application Administrators—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

Related Information

AquaLogic Service Bus uses the WebLogic security framework as building blocks for higher level security services, including authentication, identity assertion, authorization, role mapping, auditing, and credential mapping. In addition to this document, the *AquaLogic Service Bus Security Guide*, the following documents provide information about the WebLogic Security Service:

- Understanding WebLogic Security—This document summarizes the features of the WebLogic Security Service and presents an overview of the architecture and capabilities of the WebLogic Security Service. It is the starting point for understanding the WebLogic Security Service.
- Securing a Production Environment—This document highlights essential security measures for you to consider before you deploy WebLogic Server into a production environment.
- Securing WebLogic Server—This document explains how to configure security for WebLogic Server and how to use Compatibility security.

- Securing WebLogic Resources—This document introduces the various types of WebLogic resources, and provides information that allows you to secure these resources using WebLogic Server.

CHAPTER 2

Understanding AquaLogic Service Bus Security

AquaLogic Service Bus supports open industry standards for ensuring the integrity and privacy of communications and to ensure that only authorized users can access resources in an AquaLogic Service Bus domain. It uses the underlying WebLogic security framework as building blocks for its security services. The WebLogic security framework divides the work of securing a domain into several components (providers), such as authentication, authorization, credential mapping, and auditing. You configure only those providers that you need for a given AquaLogic Service Bus domain.

The following sections introduce the AquaLogic Service Bus security model and its features:

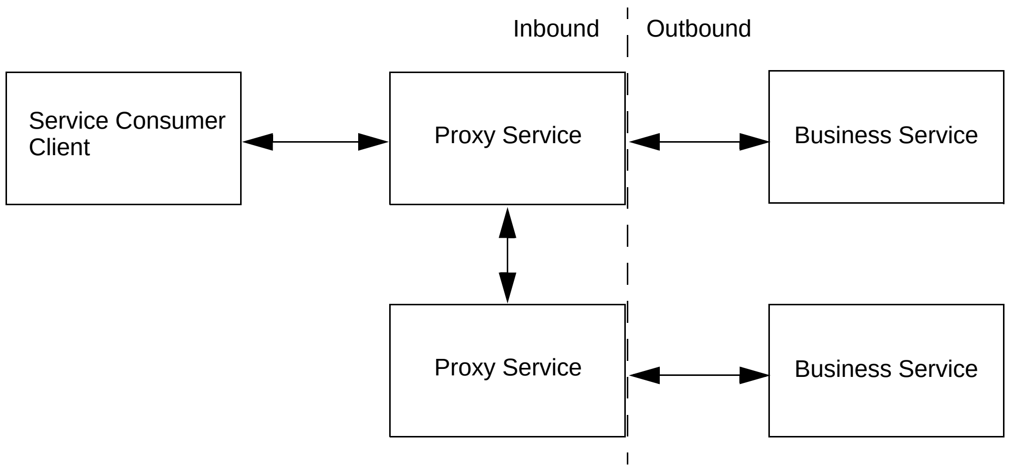
- "Inbound Security" on page 2-2
- "Outbound Security" on page 2-4
- "Options for Identity Propagation" on page 2-4
- "Administrative Security" on page 2-17
- "Configuring the WebLogic Security Framework: Main Steps" on page 2-21
- "Supported Standards and Security Providers" on page 2-26

Inbound Security

Inbound security ensures that AquaLogic Service Bus proxy services handle only the requests that come from authorized clients. (By default, any anonymous or authenticated user can connect to a proxy service.) It can also ensure that no unauthorized user has viewed or modified the data as it was sent from the client.

Proxy services can have two types of clients: service consumers and other proxy services. Figure 2-1 illustrates that communication between proxy services and their clients is secured by inbound security, while communication between proxy services and business services is secured by outbound security.

Figure 2-1 Inbound and Outbound Security



You set up inbound security when you create proxy services and you can modify it as your needs change. For outward-facing proxy services (which receive requests from service consumers), consider setting up strict security requirements such as two-way SSL over HTTPS. For proxy services that are guaranteed to receive requests only from other AquaLogic Service Bus proxy services, you can use less secure protocols.

If a proxy service uses public key infrastructure (PKI) technology for digital signatures, encryption, or SSL authentication, create a **proxy service provider** to provide private keys paired with certificates. For more information, see Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

For each proxy service, you can configure the following inbound security checks:

- **Transport-level security** applies security checks as part of establishing a connection between a client and a proxy service. The security requirements that you can impose through transport-level security depend on the protocol that you configure the proxy service to use.

  For example, for proxy services that communicate over the HTTP protocol, you can require that all clients authenticate against a database of users that you create in the Security Configuration module of the AquaLogic Service Bus Console. You then create an access control policy that specifies conditions under which authenticated users are authorized to access the proxy service.

  AquaLogic Service Bus also supports client-specified custom authentication tokens for inbound transport-level requests.

  For information about configuring transport-level security for each supported protocol, see "Configuring Transport-Level Security" on page 4-1.

- **Custom Authentication for message-level security.** AquaLogic Service Bus supports client-specified custom authentication credentials for inbound transport- and message-level requests. The custom authentication credentials can be in the form of a custom token, or a username and password.

  For information on configuring custom authentication transport- and message-level security, see "Configuring Custom Authentication" on page 5-1.

- **Message-level security** (for proxy services that are Web Services) is part of the WS-Security specification. It applies security checks before processing a SOAP message or specific parts of a SOAP message.

  Part of the configuration for message-level security is embedded in the WSDL document and WS-Policy document that are associated with the Web service. These documents specify whether SOAP messages must be digitally signed and encrypted and which Web service operations can be invoked only by authorized users.

  If a proxy service or business service uses a WS-Policy statement to secure access to one or more of its operations, and if you have configured the service as an active intermediary (as opposed to a pass-through service), you use the AquaLogic Service Bus Console to create a message-level access control policy. The policy specifies conditions under which users, groups, or security roles are authorized to invoke the protected operations.

  For more information about configuring message-level security, see "Configuring Message-Level Security for Web Services" on page 6-1.

Outbound Security

Outbound security secures communication between a proxy service and a business service. Most of the tasks that you complete for outbound security are for configuring proxy services to comply with the transport-level or message-level security requirements that business services specify.

For example, if a business service requires user name and password tokens, you create a service account, which either directly contains the user name and password, passes along the user name and password that was contained in the inbound request, or provides a user name and password that depend on the user name that was contained in the inbound request. For more information, see Service Accounts in *Using the AquaLogic Service Bus Console*.

If a business service requires the use of PKI technology for digital signatures, or SSL authentication, you create a proxy service provider, which provides private keys paired with certificates. For more information, see Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

Options for Identity Propagation

A key group of decisions that you must make when designing security for AquaLogic Service Bus is how to handle (propagate) the identities that clients provide. You can configure AquaLogic Service Bus to do any of the following:

- Authenticate the credentials that clients provide
- Perform authorization checks
- Pass client credentials to business services unchanged
- Map client credentials to a different set of credentials that a business service can authenticate and authorize
- Bridge between security technologies

Table 2-1 describes the decisions that affect how AquaLogic Service Bus propagates client identities to business services.

Table 2-1 Options for Identity Propagation

Decision	Description
Which type of credentials do you require clients to provide?	For transport-level security, AquaLogic Service Bus adapts to your existing security requirements. Clients of AquaLogic Service Bus can supply user name and password tokens, SSL certificates, or any other type of custom authentication token that is supported by an Identity Assertion provider that you configure. For message-level security, AquaLogic Service Bus supports the Username Token, X.509 Token, any other type of custom authentication token that is supported by an Authentication or Identity Assertion provider that you configure, and SAML Token profiles (see "Supported Standards and Security Providers" on page 2-26). If you are establishing security requirements for a new business service that uses Web Services Security, BEA recommends that you require clients to provide SAML tokens. SAML is the emerging standard for propagating user identities within Web services. See "Using SAML for Authentication" on page 8-1.
Do you require AquaLogic Service Bus to authenticate clients or to simply pass the client-supplied credentials to business services for authentication?	When you require clients to authenticate with AquaLogic Service Bus, you add an additional layer of security. In general, the more security layers you add, the more secure you make a domain. To enable AquaLogic Service Bus to authenticate users, you must create user accounts in the AquaLogic Service Bus Console. If your set of users is very large, you must consider whether maintaining a large database of user accounts in the AquaLogic Service Bus Console is worth the effort.

Table 2-1 Options for Identity Propagation

Decision	Description
If AquaLogic Service Bus authenticates clients that provide X.509 tokens or SAML tokens, which AquaLogic Service Bus user maps to the tokens?	BEA recommends that you require clients to authenticate with AquaLogic Service Bus and that you modify the default access-control policies to allow (authorize) only specific, authenticated users access to your proxy services. To authenticate and authorize clients who supply X.509 certificates, SAML tokens, or other types of credentials other than user names and passwords, you must configure an identity assertion provider that maps the client's credential to an AquaLogic Service Bus user. AquaLogic Service Bus will use this user name to establish a security context for the client.

Table 2-1 Options for Identity Propagation

Decision	Description
If AquaLogic Service Bus authenticates clients that provide custom authentication tokens, which AquaLogic Service Bus user maps to the tokens?	BEA recommends that you require clients to authenticate with AquaLogic Service Bus and that you modify the default access-control policies to allow (authorize) only specific, authenticated users access to your proxy services. To authenticate and authorize clients who supply custom authentication tokens other than user names and passwords, you must configure an Identity Assertion provider that maps the client's credential to an AquaLogic Service Bus user. AquaLogic Service Bus will use this user name to establish a security context for the client.
If AquaLogic Service Bus authenticates clients that provide user name and password tokens, decide whether you want to: • Pass the client's user name and password to the business service • Map the client's user name to a new user name and password and pass the new credentials to the business service	If a custom username/password token is used, as described in "What Are Custom Authentication Tokens?" on page 5-2, then the username and password in the custom token can be used for outbound HTTP BASIC or outbound WS-Security Username Token authentication if a pass-through service account is used. If you pass the client-supplied user name and password to the business service, then clients are responsible for maintaining the credentials that the business service requires. If the business service changes its security requirements, then you must notify each client to make corresponding changes. If you expect a business service to change its requirements frequently, then consider mapping the credentials that clients supply to the credentials that the business service requires. The more clients for a business service, the more work will be required to maintain this credential mapping.

Table 2-2 describes all combinations of the requirements that you can impose for inbound and outbound transport-level security.

Table 2-2 Combinations of Transport-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password in the HTTP header and AquaLogic Service Bus authenticates the client.	Pass the client's credentials in an HTTP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Configure outbound HTTPS (or HTTP) security. See "Configuring Outbound HTTPS Security: Main Steps" on page 4-6. Be sure to create a **pass-through** service account and attach the account to the business service.
	Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials in an HTTP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Configure outbound HTTPS (or HTTP) security. See "Configuring Outbound HTTPS Security: Main Steps" on page 4-6. Be sure to create a **user-mapping** service account and attach the account to the business service.

Table 2-2 Combinations of Transport-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password in the HTTP header and AquaLogic Service Bus **does not authenticate** the client.	Pass the client's credentials in an HTTP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to configure the proxy service for HTTP, no authentication or HTTPS, one-way SSL, no authentication. 2. Configure outbound HTTPS (or HTTP) security. See "Configuring Outbound HTTPS Security: Main Steps" on page 4-6. Be sure to configure the business service for HTTP BASIC authentication or HTTPS, one-way SSL, BASIC authentication. Also create a **pass-through** service account and attach the account to the business service.
Client supplies custom authentication token in the HTTP header. AquaLogic Service Bus authenticates the client.	Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials in an HTTP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Configure outbound HTTPS (or HTTP) security. See "Configuring Outbound HTTPS Security: Main Steps" on page 4-6. Be sure to create a **user-mapping** service account and attach the account to the business service.

Table 2-2 Combinations of Transport-Level Security Requirements

This Inbound Requirement…	Can Be Used With This Outbound Requirement…	How to Configure
Any form of local authentication (HTTP or HTTPS BASIC, HTTPS CLIENT CERT with credential mapping)	Pass the client’s credentials to an EJB over RMI. The EJB container authenticates the user.	Create a pass-through service account and attach the account to the business service. See “Service Accounts” in *Using the AquaLogic Service Bus Console*.

Table 2-3 describes all combinations of the requirements that you can impose for inbound and outbound message-level security. In some cases, the inbound requirement for *transport-level* security affects the requirements that you can impose for outbound message-level security.

Table 2-3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password, or custom authentication token, in the HTTP header and AquaLogic Service Bus authenticates the client.	Pass the client's credentials in a SOAP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Create a pass-through service account and attach the account to the business service. See "Service Accounts" in *Using the AquaLogic Service Bus Console*.
	Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials in a SOAP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Create a user-mapping service account and attach the account to the business service. See "Service Accounts" in *Using the AquaLogic Service Bus Console*.
	Map the client credentials to a SAML token. AquaLogic Service Bus asserts the user identity.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.

Table 2-3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies custom user name and password, or custom authentication token, in the message header or body and AquaLogic Service Bus authenticates the client.	Pass the client's credentials in a SOAP header.	1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Create a pass-through service account and attach the account to the business service. See "Service Accounts" in *Using the AquaLogic Service Bus Console*.
	Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials in a SOAP header.	1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Create a user-mapping service account and attach the account to the business service. See "Service Accounts" in *Using the AquaLogic Service Bus Console*.
	Map the client credentials to a SAML token. AquaLogic Service Bus asserts the user identity.	1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the AquaLogic Service Bus Security Configuration module. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.

Table 2-3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password in the HTTP header and AquaLogic Service Bus **does not authenticate** the client.	Pass the client's credentials in a SOAP header.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. Be sure to configure the proxy service for HTTP, no authentication or HTTPS, one-way SSL, no authentication. 2. Configure outbound HTTPS (or HTTP) security. See "Configuring Outbound HTTPS Security: Main Steps" on page 4-6. Be sure to configure the business service for HTTP BASIC authentication or HTTPS, one-way SSL, BASIC authentication. Also create a **pass-through** service account and attach the account to the business service.
Client supplies a certificate as part of HTTPS CLIENT-CERT authentication (two-way SSL) and AquaLogic Service Bus authenticates the client.	Map the client credentials to a SAML token. AquaLogic Service Bus asserts the user identity.	1. Configure inbound HTTPS (or HTTP) security. See "Configuring Inbound HTTPS Security: Main Steps" on page 4-5. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.

Table 2-3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
An **active intermediary** proxy service enforces Web-Services Security with the User Name Token Profile.	Encode the credentials as a user name and password token in the SOAP message.	Create an active intermediary proxy service with a WS-Policy statement that requires passwords (**not password digests**). See "Creating an Active Intermediary Proxy Service: Main Steps" on page 6-5.
	Encode the credentials as a SAML token in the SOAP message.	1. Create an active intermediary proxy service with a WS-Policy statement that requires passwords. See "Creating an Active Intermediary Proxy Service: Main Steps" on page 6-5. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.
An **active intermediary** proxy service enforces Web-Services Security with the X.509 Token Profile.	Encode the credentials as a SAML token in the SOAP message.	1. Create an active intermediary proxy service with a WS-Policy statement that requires digital signatures and optionally requires authentication with an X.509 token. See "Creating an Active Intermediary Proxy Service: Main Steps" on page 6-5. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.

Table 2-3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
An **active intermediary** proxy service enforces Web-Services Security with the SAML Token Profile.	Generate a new SAML token in the outbound SOAP message.	1. Create an active intermediary proxy service with a WS-Policy statement that requires a SAML token. See "Authenticating SAML Tokens in Inbound Requests" on page 8-3. 2. Configure a SAML credential mapping provider. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.
A pass-through proxy service, which can pass user names and passwords, X.509 tokens, or SAML tokens.	A business service that uses either the User Name Token Profile, the X.509 Token Profile, or the SAML Token Profile.	1. Create a pass through proxy service. See "Creating a Pass-Through Proxy Service: Main Steps" on page 6-7. 2. Create a business service that enforces one of the token profiles. See "Configuring Outbound Message-Level Security: Main Steps" on page 6-8 or "Configuring SAML Pass-Through Identity Propagation" on page 8-3.

For inbound Tuxedo requests, you can configure any of the following security requirements:

- Encode the client's credentials in an outbound call to a Tuxedo service.
- Encode the client's credentials in an outbound SOAP message as either a user name token or a SAML token.
- Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials in an outbound HTTP header.
- Map the client's credentials to a different AquaLogic Service Bus user and pass the new credentials to an EJB over RMI. The EJB container authenticates the user.

For information about using Tuxedo with AquaLogic Service Bus, see *Interoperability Solution for Tuxedo*.

Example: Authentication with a User Name Token

Figure 2-2 illustrates how user identities flow through AquaLogic Service Bus when you configure AquaLogic Service Bus as follows:

- Require clients to provide user names and passwords in their requests

  You can require Web services clients to provide credentials at the transport level, the message level, or both. If you require clients to provide credentials at both levels, AquaLogic Service Bus uses the message-level credentials for identity propagation and credential mapping.

- Authenticate clients

The illustration begins with the inbound request and ends with the outbound request:

1. A client sends a request to a proxy service. The request contains the user name and password credentials.

   Clients can send other types of tokens for authentication, such as an X.509 certificate or a custom authentication token. If a client sends an X.509 certificate token or a custom token, you must configure an identity assertion provider to map the identity in the token to an AquaLogic Service Bus security context.

2. The proxy service asks the domain’s authentication provider if the user exists in the domain’s authentication provider store.

   If the user exists, the proxy service asks the domain’s authorization provider to evaluate the access control policy that you have configured for the proxy service.

3. If the proxy service’s access control policy allows the user access, the proxy service processes the message. As part of generating its outbound request to a business service, the proxy service asks the business service to supply the user name and password that the business service requires.

   The business service asks its service account for the credentials. Depending on how the service account is configured, it does one of the following:

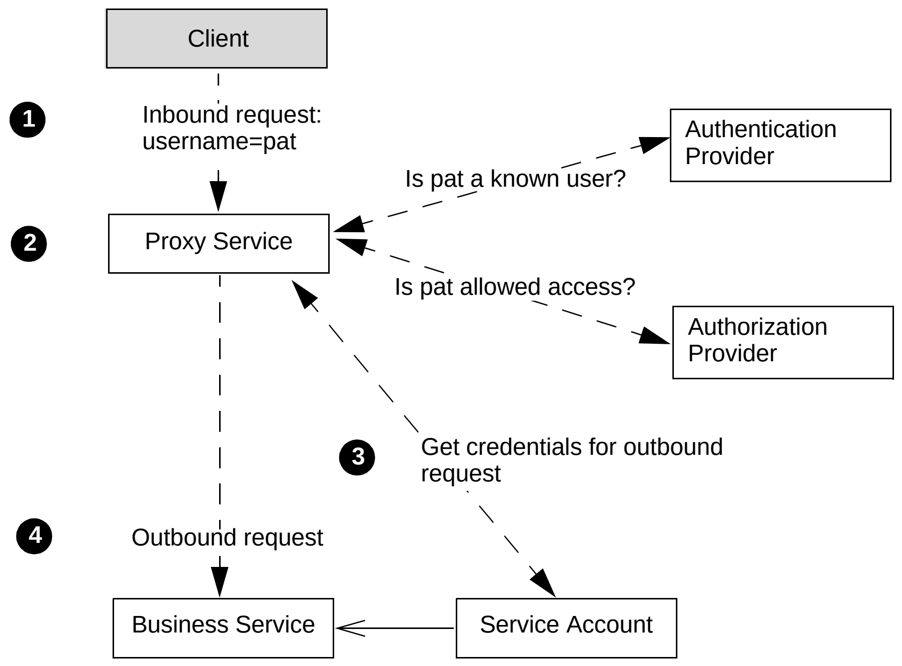
   – Requires the proxy service to encode a specific (static) user name and password.

   – Requires the proxy service to pass along the user name and password that the client supplied.

   – Maps the user name that was returned from the authentication provider to some other (remote) user name, then requires the proxy service to encode the remote user name.

4. The proxy service sends its outbound request with the user name and password that was returned from the service account.

Figure 2-2 How Service Accounts Are Used



Administrative Security

To secure access to administrative functions, such as creating proxy services or business services, AquaLogic Service Bus provides four security roles with pre-defined access privileges:

- IntegrationAdmin
- IntegrationDeployer
- IntegrationMonitor
- IntegrationOperator

A security role is an identity that can be dynamically conferred upon a user or group at runtime. You cannot change the access privileges for these administrative security roles, but you can change the conditions under which a user or group is in one of the roles.

The AquaLogic Service Bus roles have permission to modify only AquaLogic Service Bus resources; they do not have permission to modify WebLogic Server or other resources on WebLogic Server. When assigning administrative users to roles, assign at least one user to the WebLogic Server Admin role. The WebLogic Server security roles are described in Table 9-2.

For more information, see "Configuring Administrative Security" on page 9-1.

Access Control Policies

Access control determines who has access to the resources in AquaLogic Service Bus. An access control policy specifies conditions under which users, groups, or roles can access a proxy service. For example, you can create a policy that always allows users in the GoldCustomer role to access a proxy service and that allows users in the SilverCustomer role to access the proxy service only after 12pm on weeknights.

An access control policy is an association between a WebLogic resource and one or more users, groups, or security roles. A security policy protects the WebLogic resource against unauthorized access. Access control policies are boolean expressions assigned to specific resources. When there is an attempt to access the resource, the expression is evaluated. The expression consists of one or more conditions joined by boolean operators, such as a role (operator) and access time (8am to 5pm). For more information about access control policies, see Security Fundamentals in Understanding WebLogic Security.

AquaLogic Service Bus relies on WebLogic Server security realms to protect its resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and (access control) security policies. To access any resources belonging to a realm, a user must be assigned a security role and defined in that realm, as described in "Administrative Security Roles and Privileges" on page 9-2. When a user attempts to access an AquaLogic Service Bus resource, WebLogic Server authenticates and authorizes the user by checking the security role assigned to the user in the relevant security realm and relevant security policy.

Note: Only a WebLogic Server administrator can define security policies or edit security roles in the AquaLogic Service Bus Console.

For all proxy services, you can create a transport-level policy, which applies a security check when a client attempts to establish a connection with the proxy service. Only requests from users who are listed in the transport-level policy are allowed to proceed.

For proxy services that are WS-Security active intermediaries, or that implement message-level custom authentication, you can also create a message-level policy. This type of policy applies a security check when a client attempts to invoke one of the secured operations. Only users who are listed in the message-level policy are allowed to invoke the operation.

The AquaLogic Service Bus Console contains a Security Configuration module for viewing and configuring users, groups, and security roles. Additionally, the AquaLogic Service Bus Console allows you to view and configure credentials.

Note: Before making changes within the Security Configuration module in the AquaLogic Service Bus Console, you must activate your configuration. For information about how to activate a session, see Using the Change Center in the Using the AquaLogic Service Bus Console.

Configuring Proxy Service Access Control

You can configure transport-level access control for all proxy services. You can also configure access control at the message-level for any WS-Security active intermediary proxy service, or for any proxy service that implements message-level custom authentication,. To configure access control, you must assign an access control policy to the proxy service, either at the transport-level or message-level (or both).

The default transport-level and message-level access control policy for all proxy services is to allow access to all requests. You must assign an access control policy to the proxy service to protect it.

You configure transport-level and message-level access control policies in the AquaLogic Service Bus Console, as described in Editing Transport-Level Access Policies and Editing Message-Level Access Policies respectively.

Impact of Lifecycle Operations on Access Control Policies

Access control policies are persisted in authorization providers and not in the repository governed by the AquaLogic Service Bus sessions. Consequently, proxy service access-control policies are independent of the session in which the associated proxy service is created.

WARNING: Before deleting, moving, renaming, cloning or deleting a proxy service (or deleting, renaming, or moving a project or folder), delete all associated Transport-level and Service-level access control security policies.

Failure to delete these policies will leave the policies in the Authorization provider database and potentially cause unexpected results and potential security vulnerabilities, such as leaving unprotected a service that was previously protected.

For example, when you create a new proxy service whose full-path (and possibly operations) matches the path in an orphan policy, the proxy is implicitly reconnected to the policy.

Note: When you clone a service, access control policies are not cloned.

If you want to delete a proxy service:

1. Create a session if you have not already done so.
2. Delete any transport security policy assigned to the proxy URL.
3. Delete any service security policies assigned to the proxy or its operations.
4. Delete the proxy service.
5. Activate the session.

If you want to move or rename a proxy service:

1. Create a session if you have not already done so.
2. Delete any service security policies assigned to the proxy service or its operations.
3. Move or rename the proxy service.
4. Active the session. The proxy service will now be moved or renamed.
5. Locate the renamed or moved proxy service and re-assign any service security policies deleted in step 2.

If you want to rename a proxy service operation:

1. Create a session if you have not already done so.
2. Delete any service security policy assigned to the operation you are renaming.
3. Change the operation name.
4. Active the session.
5. Re-assign the service security policy deleted in step 2 to the new operation.

Configuring the WebLogic Security Framework: Main Steps

Many of the initial configuration tasks for AquaLogic Service Bus security require you to work in the WebLogic Server Administration Console to configure the WebLogic security framework. After these initial tasks, you can complete most security tasks from the AquaLogic Service Bus Console.

To configure the WebLogic security framework for AquaLogic Service Bus:

1. If you plan to use SSL as part of transport-level security, do the following:

   a. In the WebLogic Server Administration Console, configure identity and trust. See Configuring Identity and Trust in *Securing WebLogic Server*.

   b. In the WebLogic Server Administration Console, configure SSL. See Configuring SSL in *Securing WebLogic Server*.

      In AquaLogic Service Bus, all HTTPS communication must use the default WebLogic Server (SSL) secure network channel. This is the only SSL network channel that AquaLogic Service Bus supports.

   BEA recommends the following for your SSL configuration:

   – If you configure two-way SSL, you must choose between two modes: *Client Certificate Requested But Not Enforced* or *Client Certificates Requested and Enforced*. BEA recommends that whenever possible you choose *Client Certificate Requested and Enforced*. For more information, see “Secure Sockets Layer (SSL)” in Security Fundamentals in *Understanding WebLogic Security.*

   – In a production environment, make sure that Host Name Verification is enabled. See “Using Host Name Verification” in Configuring SSL in *Securing WebLogic Server.*

2. In the WebLogic Server Administration Console, configure authentication providers, which your proxy services use for inbound security.

Table 2-4 describes the authentication providers that are commonly configured for AquaLogic Service Bus. For a description of all authentication providers that you can configure, see Security Providers in *Securing WebLogic Server.*

Table 2-4 Authentication Providers

If You Require Clients to Provide...	Configure...
Simple user names and passwords	The WebLogic Authentication provider and use the AquaLogic Service Bus Console to enter the user names and passwords of the clients that you want to allow access. See “Adding a User” under Security Configuration in *Using the AquaLogic Service Bus Console*.
X.509 tokens for inbound HTTPS and two-way SSL authentication	All of the following: • The WebLogic Identity Assertion provider, which can validate X.509 tokens but does not by default. Make sure that you enable this provider to support X.509 tokens. In addition, enable this provider to use a user name mapper. See “Identity Assertion and Tokens” under “Authentication” in Security Fundamentals in *Understanding WebLogic Security*. • WebLogic CertPath Provider, which completes and validates certificate chains by using trusted Certificate Authority based checking.
Custom authentication and username/password tokens for inbound HTTP, HTTPS and message-level authentication	All of the following: • An Identity Assertion provider, possibly user-written or from a third-party, that can validate the token type. Make sure that you enable this provider to support the token. See “Identity Assertion and Tokens” under “Authentication” in Security Fundamentals in *Understanding WebLogic Security*.

Table 2-4 Authentication Providers

If You Require Clients to Provide...	Configure...
X.509 tokens for inbound Web Services Security X.509 Token Authentication	If any of your proxy services or business services are Web services that use abstract WS-Policy statements, you must also configure the following: • In the Web Service security configuration named `__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__` add the `UseX509ForIdentity` property and set it to `true`. See Use X.509 Certificates to Establish Identity in the *WebLogic Server Administration Console Online Help*.
SAML tokens	All of the following: • WebLogic SAML Identity Assertion Provider V2, which authenticates users based on Security Assertion Markup Language 1.1 (SAML) assertions. • WebLogic SAML Credential Mapping Provider V2, which maps AquaLogic Service Bus users to remote users.

3. If needed, in the WebLogic Server Administration Console, configure one or more Identity Assertion providers to handle the token types, such as X.509 or custom token types, for which you require support. For a description of all Identity Assertion providers that you can configure, see Security Providers in *Securing WebLogic Server.*
4. If you plan to create proxy services or business services that require WS-Security digital signatures on inbound requests, enable the Certificate Registry provider, which is a Certification Path provider that validates inbound certificates against a list of certificates that you register.

   See Configure Certification Path Providers in *WebLogic Server Administration Console Online Help*.
5. If you configure message-level security (in inbound requests or outbound requests) to require user name and password tokens, and if you want messages to provide a password digest instead of cleartext passwords, do the following:
   a. In the WebLogic Server Administration Console, find the two Web Service security configurations that AquaLogic Service Bus provides and set the value of the `UsePasswordDigest` property to `true`.

The AquaLogic Service Bus Web Service security configurations are named: `__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__` and `__SERVICE_BUS_OUTBOUND_WEB_SERVICE_SECURITY_MBEAN__`

For information on setting the values in Web Service security configurations, see Use a Password Digest in SOAP Messages in the *WebLogic Server Administration Console Online Help*.

b. For each authentication provider that you configured in step 2, in the WebLogic Server Administration Console, select the Password Digest Enabled check box.

c. For each identity assertion provider that you configured in step 2, in the WebLogic Server Administration Console set `wsse:PasswordDigest` as one of the active token types.

6. If you plan to create a proxy service provider (which passes key-certificate pairs in outbound requests), use the WebLogic Server Administration Console to configure a PKI credential mapping provider. In any WebLogic Server domain that hosts AquaLogic Service Bus, you can configure at most one PKI credential mapping provider.

   A PKI credential mapping provider maps AquaLogic Service Bus proxy service providers to key-pairs that can be used for digital signatures and encryption (for Web Services Security) and for outbound SSL authentication. For more information, see "Configuring a PKI Credential Mapping Provider" in Configuring WebLogic Security Providers in *Securing WebLogic Server.*

   You store the key-pairs that the PKI credential mapping provider uses in a keystore. You can store the PKI credential mappings in the identity keystore that WebLogic Server uses (which is typically the same keystore that you use to store key-pairs for SSL communication) or in a separate keystore. Configure each WebLogic Server instance to have access to its own copy of each keystore. All entries referred to by the PKI credential mapper must exist in all keystores (same entry with the same alias). For information about configuring keystores in WebLogic Server, see "Identity and Trust" in Security Fundamentals in *Understanding WebLogic Security*.

   Note: When you create an AquaLogic Service Bus domain, by default the domain contains a user name/password credential mapping provider, which you can use if you need credential mapping for user names and passwords. In addition to this user name/password credential mapping provider, you can add one PKI credential mapping provider. An AquaLogic Service Bus domain can contain at most one user name/password credential mapping provider, one PKI credential mapping provider, and multiple SAML credential mapping providers.

7. If you want to enable security auditing, do the following:

a. In the WebLogic Server Administration Console, configure an auditing provider. See Configuring a WebLogic Auditing Provider in *Securing WebLogic Server.*

b. To enable auditing of events related to WS-Security, when you start each AquaLogic Service Bus server, include the following Java option in the server's startup command:

```
-Dcom.bea.wli.sb.security.AuditWebServiceSecurityErrors=true
```

AquaLogic Service Bus supports the auditing of security events but it does not support configuration auditing, which emits log messages and generates audit events when a user changes the configuration of any resource within a domain or invokes management operations on any resource within a domain. See Configuration Auditing *Securing WebLogic Server.*

8. If you have not already done so, in the WebLogic Server Administration Console, activate your changes. If you have made changes that require you to restart WebLogic Server, the Administration Console will indicate that a restart is required. If you see such a message, restart all WebLogic Server instances that host AquaLogic Service Bus so your modifications to the security providers will be in effect for the remaining configuration steps.

Supported Standards and Security Providers

This release of AquaLogic Service Bus supports the following standards.

Table 2-5 Web Services Security and Related Standards

Standard	Version
WS-Security	1.0
WS-Policy	Because the WS-Policy specification has not been fully standardized, AquaLogic Service Bus supports a WebLogic Server-proprietary format that is based on the assertions described in the December 18, 2002 version of the Web Services Security Policy Language (WS-SecurityPolicy) specification. This release of AquaLogic Service Bus does **not** incorporate the latest update of the specification (13 July 2005).
WS-Policy Attachment	1.0
WS-Security: Username Token Profile	1.0
WS-Security: X.509 Token Profile	1.0
WS-Security: SAML Token Profile	1.0
SAML	1.1

For information about the standards that WebLogic Server supports, see “Standards Support” under What's New in WebLogic Server in *WebLogic Server Release Notes*.

Support for WebLogic Security Providers

AquaLogic Service Bus supports the security providers that are included with WebLogic Server, such as the WebLogic authentication providers, identity assertion providers, authorization providers, role-mapping providers, credential mapping providers, and Certificate Lookup and Validation (CLV) providers. Additionally, AquaLogic Service Bus supports the WebLogic SAML Identity Assertion Provider V2 and WebLogic SAML Credential Mapping Provider V2.

AquaLogic Service Bus supports the WebLogic XACML Authorization provider and XACML Role Mapping provider, which use the OASIS standard eXtensible Access Control Markup Language (XACML). Support for the WebLogic Default Authorization provider and Default

Role Mapping provider was deprecated in AquaLogic Service Bus 2.5. These providers are not supported anymore. If you are upgrading from a previous release of AquaLogic Service Bus in which you used the WebLogic Default Authorization provider and Default Role Mapping provider, use the WebLogic Server Administration Console to import authorization and role-mapping data into the XACML providers. See Upgrading AquaLogic Service Bus Environments in *AquaLogic Service Bus Upgrade Guide*.

Third-party security providers have not been tested and therefore have not been certified in AquaLogic Service Bus. However, the AquaLogic Service Bus security architecture supports the use of third-party authentication, authorization and role-mapping providers. Contact BEA customer support if you are interested in third-party security provider support in AquaLogic Service Bus.

For more information about the security providers, see “WebLogic Security Providers” in the WebLogic Security Service Architecture in Understanding WebLogic Security.

Configuring Authentication Providers

Check the provided WebLogic Server Authentication providers to see if one meets your needs. WebLogic Server includes a broad array of Authentication providers, including the following:

- The *`WebLogic Authentication provider`* accesses user and group information in WebLogic Server's embedded LDAP server. This is the default out-of-the-box authentication provider.
- *LDAP Authentication providers* access external LDAP stores. You can use an LDAP Authentication provider to access any LDAP server. WebLogic Server provides LDAP Authentication providers already configured for Open LDAP, Sun iPlanet, Microsoft Active Directory and Novell NDS LDAP servers.
- *RDBMS Authentication providers* access external relational databases. WebLogic Server provides three RDBMS Authentication providers: SQL Authenticator, Read-only SQL Authenticator, and Custom RDBMS Authenticator.
- The *SAML Authentication provider*, which authenticates users based on Security Assertion Markup Language 1.1 (SAML) assertions.

If one of the bundled Authentication providers meets your needs, see Configuring Authentication Providers for instructions on how to configure this Authentication provider in the WebLogic Server Administration Console.

If none of the Authentication providers included in WebLogic Server suits your needs, you (or a third-party) must first write a custom Authentication provider and then use the WebLogic Server Administration Console to add that provider to the security realm. To do this, follow these steps:

Note: Only a broad overview of the required tasks is included here. You will need to consult the WebLogic Server documentation to actually complete the tasks.

1. Create Runtime Classes Using the Appropriate SSPIs
2. Generate an MBean Type Using the WebLogic MBeanMaker
3. Configure the Custom Authentication Provider Using the Administration Console

See Authentication Providers in *Developing Security Providers for WebLogic Server* for additional information.

Using a Custom Authorization Provider to Protect AquaLogic Service Bus Resources

You can use AquaLogic Service Bus resources with custom Authorization providers, but those providers must understand the type and format of the AquaLogic Service Bus resources.

There are three possible resource objects for AquaLogic Service Bus that an Authorization provider must be able to detect and handle:

- "ALSBProxyServiceResource Object" on page 2-29
- "ProjectResourceV2 Object" on page 2-32
- "ConsoleResource Object" on page 2-32

These resource objects are described in the sections that follow.

WebLogic Authorization Provider Usage Information

This section briefly describes the WebLogic Server Authorization provider SSPI. See Developing Security Providers for WebLogic Server for complete information.

You protect resources by binding access control policies to resources via the AquaLogic Service Bus console, third-party tools or scripts. The WebLogic Server Security Service Provider Interface (SSPI) requires containers, such as AquaLogic Service Bus, to implement the Resource SPI. These implementations represent concrete resources.

The Authorization provider database contains a map from resource to policy. When an attempt is made to access a resource, the container calls the runtime SSPI to get an access control decision. The container passes a resource instance indicating which resource is being accessed.

An Authorization provider has one method, `getAccessDecision()`. The `getAccessDecision()` method obtains the implementation of the AccessDecision SSPI. The AccessDecision SSPI itself has one method, `isAccessAllowed()`. `isAccessAllowed` has five parameters, one of which is the Resource object for which access is being requested.

`isAccessAllowed` determines if the requestor should be allowed to access the named resource. To do this, the Authorization provider must find the right access control policy to evaluate. The provider must first look for a policy bound to the resource passed in. The lookup can use either the Resource.getId() or Resource.toString() method as a lookup key. If no policy is found, the Authorization provider must then get the parent resource and look again. This process is repeated until a policy is found or the parent is null, in which case no policy is found. When no policy is found, `isAccessAllowed` must return false.

This algorithm allows you to create coarse-grained policies that protect all proxy services in a given project or folder, all resources in a project, or all AquaLogic Service Bus proxy services in an AquaLogic Service Bus domain. More specific, finer-grained policies take precedence over coarse-grained policies.

Note: The AquaLogic Service Bus console user interface does not provide pages for protecting proxy services at the folder, project or domain level.

ALSBProxyServiceResource Object

The `ALSBProxyServiceResource` object is used for transport-level and message-level access control to ALSB proxy services. The `ALSBProxyServiceResource` resource extends `weblogic.security.service.ResourceBase`, which itself implements `weblogic.security.spi.Resource`.

`ALSBProxyServiceResource` implements the following methods, as described in `weblogic.security.spi.Resource`:

getType()
Returns the type, where type is "<alsb-proxy-service>"

getKeys()
Returns up to four key-value properties: `path`, `proxy`, `action`, and `operation`. The properties are defined as follows:

- `path` is the full-name of the proxy service. For example, `path=project/folder1/folder2`
- `proxy` is the name of the proxy service. For example, `proxy=myProxy`
- `action` is one of two values, `invoke` or `wss-invoke`. For example, `action=invoke`

  The action attribute is used to distinguish between transport-level and message-level access control. `invoke` is used for transport-level access control. `wss-invoke` is used for message-level access control; that is, access control on WS-Security active intermediaries or proxies with custom message-level authentication. The operation attribute is only allowed when action is `wss-invoke`.
- `operation` is the name of the operation to invoke, and is used only when `action` is `wss-invoke`. For example, `operation=processPO`. The `operation` attribute is only allowed when action is `wss-invoke`.

An `ALSBProxyServiceResource` has from 1 to 4 keys. The following table explains how the various combinations protect proxy services. The most specific policies take precedence.

If the Resource Contains These Keys	A Policy Bound to the Resource Protects:
path	The policy protects all proxy services in the given path
path and proxy	The policy protects all access to the given proxy service (transport-level as well as message-level)
path, proxy, and action	If action="invoke": • The policy is the transport-level policy to the given proxy - If action="wss-invoke": • The policy is the message-level policy to the given proxy (for all operations)
path, proxy, action="wss-invoke", and operation	The policy is a message-level policy for the given proxy and operation

getPath()

Gets the path (project and folders) to the proxy service. This is the path where the proxy service exists within the AquaLogic Service Bus configuration framework.

getProxyServiceName()

Gets the name of the proxy service. For example, `proxy=myProxy`.

getAction()

Gets one of two values, `invoke` or `wss-invoke`. For example, `action=invoke`.

getOperation()

Gets the name of the operation to invoke, and is used only when action is `wss-invoke`. For example, `operation=processPO`.

makeParent()

Creates a new `ALSBProxyServiceResource` object that represents the parent of the current `ALSBProxyServiceResource` resource. `makeParent()` uses the path of the proxy service to create the parent.

ALSBProxyServiceResource Examples

The following examples show various uses of the ALSBProxyServiceResource object.

- Using ALSBProxyServiceResource for transport-level access control for proxy project/folder/myProxy:

  ```
  type=<alsb-proxy-service>, path=project/folder, proxy=myProxy,
  action=invoke
  ```

- Using ALSBProxyServiceResource for message-level access control for operation processPO on proxy `project/folder/myProxy`:

  ```
  type=<alsb-proxy-service>, path=project/folder, proxy=myProxy,
  action=wss-invoke, operation=processPO
  ```

- Using the parentage hierarchy for an ALSBProxyServiceResource, from fine-grained to coarse-grained:

  ```
  type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy,
  action=wss-invoke, operation=foo

  type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy,
  action=wss-invoke

  type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy

  type=<alsb-proxy-service>, path=myProject/f1/f2
  ```

```
type=<alsb-proxy-service>, path=myProject/f1
type=<alsb-proxy-service>, path=myProject
type=<alsb-project>, project-name=myProject
type=<alsb-proxy-service>
```

ProjectResourceV2 Object

The `ProjectResourceV2` is the root resource for all `ALSBProxyServiceResource` objects in a given project. `ProjectResourceV2` extends `ResourceBase`.

Setting an access control policy on a `ProjectResourceV2` provides a coarse-grained access control policy for all proxy services in the given project that do not have more specific policies.

`ProjectResourceV2` has the following methods:

getType()
: Returns the type, where type is `"<alsb-project>".`

getKeys()
: Returns the key, where key is `"project-name”.`

getName()
: Gets the name of the `ProjectResourceV2` object.

makeParent()
: There is no parent for an `ProjectResourceV2` object. This method therefore returns the object name that was used to create the `ProjectResourceV2` object, or null if `ProjectResourceV2` does not exist.

ConsoleResource Object

The `com.bea.wli.security.resource.ConsoleResource` object is used for access control to the ALSB console. However, we do not recommend that you set access control policies for `ConsoleResource` objects via a custom Authorization provider. This is because these policies are subject to change in future AquaLogic Service Bus releases.

We instead recommended that even if you need to use a custom Authorization provider, you also continue to use the WebLogic Server XACML Authorization provider to maintain the policies for the ConsoleResource object. In this case of two Authorization providers, you must also configure an Adjudication provider.

CHAPTER 3

AquaLogic Service Bus Security FAQ

This section includes frequently asked questions about AquaLogic Service Bus security and their answers. It includes the following questions:

- Is it possible to customize the format of the subject identity in a SAML assertion?
- Is single sign-on supported in AquaLogic Service Bus?
- Are security errors monitored?
- Can I configure security for MBeans?

How are AquaLogic Service Bus and WebLogic Server Security related?

AquaLogic Service Bus leverages the WebLogic Security Framework. The details of this framework are described in "WebLogic Security Framework" in WebLogic Security Service Architecture in *Understanding WebLogic Security*. Before configuring security in AquaLogic Service Bus, you must configure a WebLogic Server security realm and other server configurations (such as SSL) in WebLogic Server, as described in "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

What is Transport-Level Security?

Transport-level security refers to the transport protocols that secure the connection over which messages are transported. An example of transport-level security is HTTPS (HTTP over SSL). SSL provides point-to-point security, but does not protect the message when intermediaries exist in the message path. For more information, see Chapter 4, "Configuring Transport-Level Security".

What is Web Services Security?

Web Services Security (WS-Security) is an OASIS standard that defines interoperable mechanisms to incorporate message-level security into SOAP messages. WS-Security supports message integrity and message confidentiality. It also defines an extensible model for including security tokens in a SOAP envelope and a model for referencing security tokens from within a SOAP envelope. WS-Security token profiles specify how specific token types are used within the core WS-Security specification. Message integrity is achieved through the use of XML digital signatures; message confidentiality is accomplished through the use of XML encryption. WS-Security allows you to specify which parts of a SOAP message are digitally signed or encrypted. AquaLogic Service Bus supports WS-Security over HTTP, HTTPS, and JMS. For more information on WS-Security see *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)* at the following URL:
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`

What is Web Service Policy?

The Web Services Policy Framework (WS-Policy) provides a general-purpose model and corresponding syntax to describe and communicate the policies of a Web service. WS-Policy defines a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements, preferences, and capabilities. For more information, see Chapter 7, "Using Web Services Policy to Specify Inbound Message-Level Security".

What are Web Service Policy assertions?

The Web Services Policy Assertions Language (WS-PolicyAssertions) specifies a set of common message policy assertions that can be specified within a security policy. The specification defines general messaging-related assertions for use with WS-Policy. Separate specifications describe the syntax and semantics of domain-specific assertions for security assertions and reliable-messaging assertions.

Are Access Control Policy and Web Service Policy the same?

No. Access control policy is a boolean expression that is evaluated to determine which requests to access a particular resource (such as a proxy service, Web application, or EJB) are granted and which should be denied access. Typically access control policies are based on the *roles* of the requestor. WS-Policy is metadata about a Web service that complements the service definition (WSDL). WS-Policy can be used to express a requirement that all service clients must satisfy, such as, all requests must be digitally signed by the client.

What is Web Services Security Pass-Through?

In a WS-Security pass-through scenario, the client applies WS-Security to the request and/or response messages. The proxy service does not process the security header, instead, it passes the secured request message untouched to a business service. Although AquaLogic Service Bus does not apply any WS-Security to the message, it can route the message based on values in the header. After the business service receives the message, it processes the security header and acts on the request. The business service must be configured with WS-Policy security statements. The secured response message is passed untouched back to the client. For example, the client encrypts and signs the message and sends it to the proxy service. The proxy service does not decrypt the message or verify the digital signature, it simply routes the message to the business service. The business service decrypts the messages and verifies the digital signature, and then processes the request. The response path is similar. This is sometimes called a passive intermediary.

What is a Web Services Security Active Intermediary?

In an active intermediary scenario, the client applies WS-Security to the request and/or response messages. The proxy service processes the security header and enforces the WS-Security policy. For example, the client encrypts and signs the message and sends it to the proxy service, the proxy decrypts the message and verifies the digital signature, then routes the message. Before the proxy service sends the response back to the client, the proxy signs and encrypts the message. The client decrypts the message and verifies the proxy's digital signature.

What is outbound Web Services Security?

Outbound WS-Security refers to security between AquaLogic Service Bus proxy services and business services. It includes both the request and response between business applications and proxy services. For more information, see "About Message-Level Security" on page 6-3.

What is SAML?

SAML (Security Assertion Markup Language) is an OASIS standards-based extensible XML framework for exchanging authentication and authorization information, allowing single sign-on capabilities in modern network environments.

Is it possible to customize the format of the subject identity in a SAML assertion?

By default, the subject identity within an outbound SAML token is the same as the inbound username. The format of the subject identity can be customized by writing a custom SAML name mapper-provider. For more information, see Configuring a SAML Credential Mapping Provider in *Securing WebLogic Server*.

What is the Certificate Lookup And Validation Framework?

The Certificate Lookup and Validation (CLV) providers complete certificate paths and validate X509 certificate chains. The two types of CLV providers are:

CertPath Builder—receives a certificate, a certificate chain, or certificate reference (the end certificate in a chain or the Subject DN of a certificate) from a Web service or application code. The provider looks up and validates the certificates in the chain.

CertPath Validator—receives a certificate chain from the SSL protocol, a Web service, or application code and performs extra validation, such as revocation checking.

At least one CertPath Builder and one CertPath Validator must be configured in a security realm. Multiple CertPath Validators can be configured in a security realm. If multiple providers are configured, a certificate or certificate chain must pass validation with all the CertPath Validators for the certificate or certificate chain to be valid. WebLogic Server provides the functionality of the CLV providers in the WebLogic CertPath provider and the Certificate Registry. For more information see "The Certificate Lookup and Validation Process" in WebLogic Security Service Architecture in *Understanding WebLogic Security*.

Does AquaLogic Service Bus support identity propagation in a proxy service?

Yes, AquaLogic Service Bus supports two methods for propagating identities:

– By generating SAML 1.1 assertions in conformance with the Web Services Security: SAML Token Profile 1.0 specification:
`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`

This is done by setting a SAML holder-of-key or sender-vouches WS-Policy on the business service routed to by the proxy.

– If a business service requires user name and password tokens, you can configure the business service's service account to pass through the user credentials from the original client request. See Service Accounts in *Using the AquaLogic Service Bus Console*.

If both transport-level authentication and message-level authentication exist on inbound messages to the proxy service, which identity is propagated?

If both transport authentication and message-level authentication exist, the message-level subject identity is propagated.

Is single sign-on supported in AquaLogic Service Bus?

Strictly speaking single sign-on (SSO) is not applicable to AquaLogic Service Bus messaging scenarios for several reasons. First, AquaLogic Service Bus is stateless; there is no notion of a session or conversation among multiple parties. Second, AquaLogic Service Bus clients are typically other enterprise software applications, not users behind a Web browser. Therefore, it is acceptable to require that these clients send credentials such as username and password on every request, provided that the communication is secured by means such as SSL or WS-Security. However, SSO between the AquaLogic Service Bus Console and the WebLogic Server Administration Console is supported. For more information, see “Single Sign-On” in Security Fundamentals in *Understanding WebLogic Security*.

Are security errors monitored?

Only WS-Security errors are monitored by the AquaLogic Service Bus monitoring framework. Transport-level security errors such as SSL handshake errors, transport-level authentication and transport-level access control are not monitored in this release. For more information, see “Service Monitoring Details” in Monitoring in the *AquaLogic Service Bus Operations Guide*. However, it is possible to configure an Auditor provider to audit transport-level authentication and authorization.

Can I configure security for MBeans?

AquaLogic Service Bus includes two managed beans (MBeans) that configure such runtime behavior as which types of credentials are available to abstract WS-Policy statements. By default, only users in the Admin and Deployer security roles can modify these MBeans, however you can change these defaults. See Create JMX Policies in *WebLogic Server Administration Console Help*.

CHAPTER 4

Configuring Transport-Level Security

Transport-level security applies security checks as part of establishing a connection between service consumers, proxy services, and business services. The type of security checks that AquaLogic Service Bus can apply depends on the protocol that the proxy service or business service uses to communicate. Some protocols can also encrypt the communication between client and endpoint to prevent snooping from third parties.

Inbound transport-level secures the communication between clients and AquaLogic Service Bus proxy services. **Outbound** transport security secures all three techniques of sending outbound requests from AquaLogic Service Bus proxy services: route actions, publish actions, and callout actions.

The following sections describe configuring transport-level security:

- "Configuring Transport-Level Security for HTTPS" on page 4-2
- "Configuring Transport-Level Security for HTTP" on page 4-7
- "Additional Context Properties for HTTP and HTTPS Transport-Level Authentication" on page 4-9
- "Email, FTP, and File Transport-Level Security" on page 4-13
- "Transport-Level Security Elements in the Message Context" on page 4-14

Note: Transport-level security secures only the connection itself. Even if you use the HTTPS or JMS protocols to encrypt the communication, if there is an intermediary between a Web services client and an AquaLogic Service Bus proxy service, such as a router, message queue or another proxy service, the intermediary gets the SOAP message in plain text. When the intermediary sends the message to the second receiver, the second receiver does not know who the original sender was. To prevent unintended intermediaries from viewing or modifying SOAP or JMS messages, configure message-level security *in addition to* transport-level security. See "Configuring Message-Level Security for Web Services" on page 6-1.

Configuring Transport-Level Security for HTTPS

The HTTPS protocol uses SSL to secure communication. SSL can be used to encrypt communication, ensure message integrity, and to require strong server and client authentication. Before you can use HTTPS, you must configure SSL in WebLogic Server, see "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

The following sections describe configuring transport-level security for the HTTPS protocol:

- "HTTPS Authentication Levels" on page 4-3
- "Configuring Inbound HTTPS Security: Main Steps" on page 4-5
- "Configuring Outbound HTTPS Security: Main Steps" on page 4-6

HTTPS Authentication Levels

For each proxy service or business service that communicates over the HTTPS protocol, you can configure the service to require one of the following levels of authentication:

- One-way SSL, no authentication

  This level enables encrypted communication but does not require clients to provide credentials. To establish a one-way SSL connection, the client initiates the connection and AquaLogic Service Bus sends its certificate to the client. In other words, the client authenticates AquaLogic Service Bus.

- One-way SSL, BASIC authentication

  This level enables encrypted communication and requires clients to supply a user name and password after the one-way SSL connection is established. The client supplies a user name and password by encoding it in the HTTP request header (which is encrypted by SSL). When the proxy service receives the encrypted request, it passes the credentials to the domain's authentication provider, which determines whether client's credentials match a user account that you have created.

- Two-way SSL, CLIENT CERT authentication

  This level enables encrypted communication and strong client authentication (two-way SSL).

  To establish a two-way SSL connection, the client initiates the connection and AquaLogic Service Bus sends its X.509 certificate to the client. Then, the client sends its certificate to AquaLogic Service Bus and AquaLogic Service Bus authenticates the client.

  To get the user name from the client's certificate, you configure an identity assertion provider, which extracts a field in the certificate to use as the client identity (X.509 token), typically the CN (common name) or E (email) of the `SubjectDistinguishedName` in the certificate. After extracting the X.509 token, the token is compared to the user accounts in the Security Configuration module of the AquaLogic Service Bus Console.

  For more information about SSL and identity assertion providers, see Security Fundamentals in *Understanding WebLogic Security.*

- Transport-Level Custom Credentials.

  You can authenticate client requests at the transport-level via custom authentication tokens. Transport-level custom credentials are supported only on inbound requests. You specify a custom token in an HTTP header. The HTTPS-specific configuration pages of the service definition wizard allows you to configure client authentication. Custom authentication concepts are described in "Configuring Custom Authentication" on page 5-1.

Configuring Inbound HTTPS Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. Make sure that you have configured the WebLogic security framework to support SSL, an authentication provider, and an identity assertion provider, depending on the HTTPS authentication level that you want to use:
   - For no client authentication (anonymous requests), set Client Authentication to None.
   - For basic authentication, set Client Authentication to Basic. See “Adding a User” under Security Configuration in *Using the AquaLogic Service Bus Console*.
   - For SSL client authentication, set Client Authentication to Client Certificate, configure the WebLogic Identity Assertion provider and the WebLogic CertPath Provider.
   - For custom authentication token, set Client Authentication to Custom Authentication. The custom authentication token can be any active token type previously configured for an Identity Assertion provider that is carried in an HTTPS header. Custom authentication concepts are described in “Configuring Custom Authentication” on page 5-1.

   Note: You must first configure, or create and configure, a WebLogic Server Identity Assertion provider as described in “Configuring Identity Assertion Providers for Custom Tokens” on page 5-6, and add the user names and passwords of the clients that you want to allow access to the Security Configuration module of the AquaLogic Service Bus Console.

   See “Configuring the WebLogic Security Framework: Main Steps” on page 2-21.

2. When you create a proxy service in the AquaLogic Service Bus Console, on the **Transport Configuration** page select HTTPS. Follow the prompts to choose an authentication level. See “Adding a Proxy Service” under Proxy Services in *Using the AquaLogic Service Bus Console*.

3. Modify the proxy service’s default transport-level access control policy, which specifies conditions under which users, groups, or roles can access a proxy service. See “Editing Transport-Level Access Policies” under Security Configuration in *Using the AquaLogic Service Bus Console*.

Configuring Outbound HTTPS Security: Main Steps

In outbound transport-level security, a proxy service is the client that opens a connection with a business service.

To configure outbound transport-level security:

1. If you are configuring transport-level security for a production environment (as opposed to a development or testing environment), make sure that Host Name Verification is enabled. See "Using Host Name Verification" in Configuring SSL in *Securing WebLogic Server.*

2. When you create a **business service** in the AquaLogic Service Bus Console, on the **Transport Configuration** page select HTTPS. See "Adding a Business Service" under Business Services in *Using the AquaLogic Service Bus Console*. Follow the prompts to choose an authentication level.

   If you configured the proxy service so that AquaLogic Service Bus does not authenticate clients, configure the enterprise system to authenticate clients by selecting an authentication level of one-way SSL, BASIC authentication.

3. If the business service uses HTTPS with BASIC authentication, create a service account to provide the user name and password that the business service requires.

   You can add a user name and password directly to the service account, or configure the service account to pass through the credentials that it received from its client's request, or you can map a client user name to an AquaLogic Service Bus user. If you configured the proxy service so that AquaLogic Service Bus does not authenticate clients, create a service account that passes through the credentials. See Service Accounts in *Using the AquaLogic Service Bus Console*.

4. If the business service uses HTTPS with CLIENT CERT authentication, do the following:

   a. Create a proxy service provider to provide the key-pair that proxy services use for SSL client authentication with the business service. See Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

   b. Create a proxy service or edit an existing proxy service so that it specifies the proxy service provider. See “Viewing and Changing Proxy Services” under Proxy Services in *Using the AquaLogic Service Bus Console*.

Configuring Transport-Level Security for HTTP

The HTTP protocol does **not** encrypt communication between clients and proxy services or business services, but it does support BASIC authentication in which clients send user names and passwords in requests. HTTP also supports custom token authentication.

Caution: Unless you have configured strong network security, BEA recommends that you do not use BASIC authentication with HTTP in production environments because the password is sent in clear text. Instead, use BASIC authentication with HTTPS.

The following sections describe configuring transport-level security for the HTTP protocol:

- “Configuring Inbound HTTP Security: Main Steps” on page 4-8
- “Configuring Outbound HTTP Security: Main Steps” on page 4-8

Configuring Inbound HTTP Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. When you create a proxy service in the AquaLogic Service Bus Console, on the **Transport Configuration** page select HTTP. Choose the Client Authentication option None, Basic, or Custom Authentication. If you choose Custom Authentication, you must also specify the HTTP header that is to carry the token and the token type.

   The steps for configuring transport-level custom credentials are described in "Adding a Proxy Service" under Proxy Services in *Using the AquaLogic Service Bus Console*. Custom authentication concepts are described in "Configuring Custom Authentication" on page 5-1.

   The custom authentication token can be any active token type, previously configured for an Identity Assertion provider, that is carried in an HTTP header.

   Note: To use custom authentication you must first configure, or create and configure, a WebLogic Server Identity Assertion provider as described in "Configuring Identity Assertion Providers for Custom Tokens" on page 5-6.

   Note: If you want AquaLogic Service Bus to authenticate clients (Basic or Custom Authentication) you must create user accounts for the clients. See "Configuring Administrative Security: Main Steps" on page 9-13.

2. Modify the proxy service's default transport-level access control policy, which specifies conditions under which users, groups, or roles can access a proxy service. See "Editing Transport-Level Access Policies" under Security Configuration in *Using the AquaLogic Service Bus Console*.

Configuring Outbound HTTP Security: Main Steps

In outbound transport-level security, a proxy service is the client that opens a connection with a business service.

To configure outbound transport-level security:

1. When you create a **business service** in the AquaLogic Service Bus Console, on the **Transport Configuration** page select HTTP. When prompted, select **Basic Authentication Required**.

   See "Adding a Business Service" under Business Services in *Using the AquaLogic Service Bus Console*.

2. Create a service account to provide the user name and password that the business service requires. See Service Accounts in *Using the AquaLogic Service Bus Console*.

   You can add a user name and password directly to the service account, or configure the service account to pass through the credentials that it received from its client's request, or you can map a client user name to an AquaLogic Service Bus user. If you configured the proxy service so that AquaLogic Service Bus does not authenticate clients, create a service account that passes through the credentials. See Service Accounts in *Using the AquaLogic Service Bus Console*.

3. Create a proxy service or edit an existing proxy service so that it specifies the service account.

Additional Context Properties for HTTP and HTTPS Transport-Level Authentication

The HTTP and HTTPS transport providers pass an AquaLogic Service Bus implementation of the WebLogic Server ContextHandler during transport-level custom token identity assertion. There is no user configuration required for this feature.

This ContextHandler has four properties, as shown in Table 4-1.

Table 4-1 Transport-Level Context Properties

Property Name	Property Value
`com.bea.contextelement.alsb.service-info`	An instance of `com.bea.wli.sb.services.ServiceInfo` that contains information about the proxy service.
`com.bea.contextelement.alsb.transport.endpoint`	An instance of `com.bea.wli.sb.transports.TransportEndPoint`. This is the HTTP or HTTPS endpoint.
`com.bea.contextelement.alsb.transport.http.http-request`	This is the `HttpServletRequest` object.
`com.bea.contextelement.alsb.transport.http.http-response`	This is the `HttpServletResponse` object.

Configuring Transport-Level Security for JMS

While transport-level security for JMS does not provide end-to-end security for JMS messaging, it does provide the following:

- The option to use a secure SSL channel for communication between AquaLogic Service Bus and a JMS server for sending or receiving JMS messages.

  AquaLogic Service Bus can communicate with local JMS servers or foreign JMS servers. The connection to JMS servers can be secured using the T3S protocol (T3 over SSL). T3 and T3S are proprietary BEA protocols.

- The ability to specify the username and password that AquaLogic Service Bus proxy services use to authenticate while establishing a connection to a JMS server and/or while looking up JMS destinations in the JNDI tree.

  Note: JMS administrators use the WebLogic Server Administration Console to create access control policies that restrict access to WebLogic JMS servers and destinations in the JNDI tree. For more information, see Configuring JMS System Resources in *Configuring and Managing WebLogic JMS* and *Securing WebLogic Resources*.

  If a JMS administrator configures or changes an access control policy for a JMS destination, WebLogic Server can take up to 60 seconds to recognize the changes.

  By default, WebLogic Server JMS checks the policy for each JMS destination every 60 seconds. To change this behavior, modify the WebLogic Server startup command so that it sets the following system property to the frequency (in seconds) that you want WebLogic Server JMS to check access control policies:
  `weblogic.jms.securityCheckInterval`
  A value of `0` (zero) for this property ensures that an authorization check is performed for every `send`, `receive`, and `getEnumeration` action on a JMS resource.

The following sections describe configuring JMS transport-level security:

- "Configuring Inbound JMS Transport-Level Security: Main Steps" on page 4-10
- "Configuring Outbound JMS Transport-Level Security: Main Steps" on page 4-12

Configuring Inbound JMS Transport-Level Security: Main Steps

To configure inbound JMS transport-level security:

1. When you create or edit a JMS proxy service in the AquaLogic Service Bus Console, on the **Transport Configuration** page, under **Advanced Settings**, select the **Use SSL** check box. See Proxy Services in the *Using the AquaLogic Service Bus Console*.

   AquaLogic Service Bus configures the JMS proxy service to use the T3S protocol.

2. If the JMS administrator created access control policies that restrict access to a JMS connection pool, configure the proxy service to authenticate when it connects to the JMS server:

   a. Create a service account to provide the user name and password that the JMS server requires. See Service Accounts in *Using the AquaLogic Service Bus Console*.

      You must add a user name and password directly in the service account. JMS cannot use a service account that passes through the credentials that it received from its client's request or that maps a client user name to an AquaLogic Service Bus user. See Service Accounts in *Using the AquaLogic Service Bus Console*.

   b. When you create or edit the proxy service in the AquaLogic Service Bus Console, on the **Transport Configuration** page, under **Advanced Settings**, click the **Browse** button next to JMS Service Account. Select the service account that you created in the previous step.

Configuring Outbound JMS Transport-Level Security: Main Steps

To configure inbound JMS transport-level security:

1. When you create or edit a JMS business service in the AquaLogic Service Bus Console, on the **Transport Configuration** page, under **Advanced Settings**, select the **Use SSL** check box. See “Adding a Business Service” under Business Services in *Using the AquaLogic Service Bus Console*.

   AquaLogic Service Bus configures the JMS proxy service to use the T3S protocol.

2. If the JMS administrator created access control policies that restrict access to a JMS connection pool, configure the business service to authenticate when it connects to the JMS server:

   a. Create a service account to provide the user name and password that the JMS server requires. See Service Accounts in *Using the AquaLogic Service Bus Console*.

      You must add a user name and password directly in the service account. JMS cannot use a service account that passes through the credentials that it received from its client’s request or that maps a client user name to an AquaLogic Service Bus user. See Service Accounts in *Using the AquaLogic Service Bus Console*.

   b. When you create or edit the business service in the AquaLogic Service Bus Console, on the **Transport Configuration** page, under **Advanced Settings**, click the Browse button next to **JMS Service Account**. Select the business account that you created in the previous step.

3. If the JMS administrator has restricted access to JMS destinations in the JNDI tree, configure the business service to authenticate when it looks up entries in the JNDI tree:

   a. (You can skip this step if the JNDI tree and JMS server require the same user name and password.) Create a service account to provide the user name and password that the JNDI tree requires. See Service Accounts in *Using the AquaLogic Service Bus Console*.

   b. When you create or edit the business service in the AquaLogic Service Bus Console, on the **Transport Configuration** page, under **Advanced Settings**, click the Browse button next to **JNDI Service Account**. Select the service account that provides the credentials that the JNDI tree requires.

      You can use the same service account for both the JMS server and the JNDI tree if both objects require the same credentials.

Email, FTP, and File Transport-Level Security

The following sections describe the security measures that are available for communication over the email, FTP, and file protocols:

- "Email and FTP Transport-Level Security" on page 4-13
- "File Transport Security" on page 4-13

Email and FTP Transport-Level Security

Email and FTP are not secure protocols. They support weak authentication, typically over insecure channels. The supported security method for email or FTP transport is the username and password needed to connect to the email or FTP server.

To secure email, you must designate a service account as an alias for the username and password in the AquaLogic Service Bus Console. The service will use the username and password to authenticate to the SMTP server.

To secure FTP, in the AquaLogic Service Bus Console, select `external_user` and designate a service account as an alias for the username and password. The service will use the username and password to authenticate to the FTP server.

For information about how to add security to email and FTP transport, see "Adding a Business Service" in Business Services in the *Using the AquaLogic Service Bus Console*.

File Transport Security

The supported security method for file transport is the user login to the computer on which the files are located.

Transport-Level Security Elements in the Message Context

If you configure a proxy service to authenticate clients, then you can access the client's identity and the security groups to which the client belongs from the proxy service's pipeline. The identity and group information is located in the message context at
`$inbound/ctx:security/ctx:transportClient/ctx:username`
and
`$inbound/ctx:security/ctx:transportClient/ctx:principals/ctx:group`
(the message context contains one `ctx:group` element for each group the user belongs to)

If a proxy service does not authenticate clients, then the value of `$inbound/ctx:security/ctx:transportClient/ctx:username` is `<anonymous>` and there will not be any `ctx:group` elements.

For more information, see "Inbound and Outbound Variables" in Message Context in the *AquaLogic Service Bus User Guide* and "Message Flow" in Proxy Services in the *Using the AquaLogic Service Bus Console*.

CHAPTER 5

Configuring Custom Authentication

AquaLogic Service Bus supports client-specified custom authentication credentials for both transport- and message-level inbound requests. The custom authentication credentials can be in the form of tokens, or a username and password token combination.

AquaLogic Service Bus accepts and attempts to authenticate a custom token passed to a proxy service in an HTTP header, SOAP header (for SOAP-based proxy services) or in the payload (for non-SOAP proxy services). You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the token is passed, and the token type.

AquaLogic Service Bus also accepts and attempts to authenticate a username and password token passed in a SOAP header (for SOAP based proxy services), or in the payload for non-SOAP proxy services. You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the username and password are passed.

Note: The custom authentication mechanisms work alone or in concert with the message-level security for Web services described in "Configuring Message-Level Security for Web Services" on page 6-1. See "Combining WS-Security with Custom Username/Password and Tokens" on page 5-15 for information about using both types of security.

The following custom authentication mechanisms are supported:

- Transport-Level Security
  - Custom token in an HTTP header
- Message-Level Security
  - For SOAP-based proxy services

    - Custom token in a SOAP header
    - Username/password in a SOAP header
  - For non-SOAP-based proxy services
    - Custom token in the payload of any XML-based proxy services
    - Username/password in the payload of any XML-based proxy services

This section describes the following custom authentication topics:

- "What Are Custom Authentication Tokens?" on page 5-2
- "Custom Authentication Token Use and Deployment" on page 5-3
- "Understanding Transport-Level Custom Authentication" on page 5-3
- "Understanding Message-Level Custom Authentication" on page 5-5
- "Configuring Identity Assertion Providers for Custom Tokens" on page 5-6
- "Additional Context Properties for Message-Level Authentication" on page 5-11
- "Security Provider Must Have Knowledge of the Property Name" on page 5-11
- "Configuring Custom Authentication Transport-Level Security" on page 5-12
- "Configuring Custom Authentication Message-Level Security" on page 5-14
- "Propagating the Identity Obtained From Custom Authentication Tokens" on page 5-15
- "Combining WS-Security with Custom Username/Password and Tokens" on page 5-15

What Are Custom Authentication Tokens?

An authentication token is some data, represented as a string or XML, that identifies an entity (user or process), such as an X509 client certificate. Typically, authentication tokens are designed to be used within specific security protocols. Some authentication tokens are cryptographically protected and some are not. Some authentication tokens carry key material.

In the context of AquaLogic Service Bus, a custom authentication token can be a username/password or an opaque identity assertion token in a user-defined location in the request. A username/password token is allowed in a SOAP header (for SOAP-based services) or in the payload of some non-SOAP proxy service. An identity assertion token is allowed in an HTTP header, in a SOAP header (for SOAP-based services), or in the payload of some non-SOAP proxy service. The AquaLogic Service Bus domain must include an Identity Assertion provider that supports the token type.

AquaLogic Service Bus uses the authenticated user to establish a security context for the client. The security context established by authenticating a custom token or username and password can be used as the basis for outbound credential mapping and access control.

To authenticate and authorize clients who supply tokens for authentication, you must configure an Identity Assertion provider that maps the client's credential to an AquaLogic Service Bus user. AquaLogic Service Bus uses this resulting username to establish a security context for the client.

Custom Authentication Token Use and Deployment

The addition of custom authentication token support in AquaLogic Service Bus addresses two customer needs. In the first scenario, an inbound request has a username/password somewhere in the message payload, for example in a SOAP header. AquaLogic Service Bus must get this username/password and authenticate the user.

In the second scenario, the message contains some kind of authentication token (other than username/password), such as a secure-token-xyz token. The token may be in an HTTP header or in the message payload. AquaLogic Service Bus must get the token and authenticate it. In either case, a security context is established if authentication succeeds.

Most security-related configuration is typically done at deployment time, and custom authentication fits that model: it can be configured directly on the production environment at deployment time. Alternatively, you can configure authentication during staging and import it into the production environment.

Custom authentication, which includes both username/password tokens and custom tokens, is an integral part of the proxy service definition. When a proxy service is exported, any configuration of custom tokens is included in the jar file. When a new version of the proxy service is imported, the previous configuration is overwritten with whatever configuration is contained in the jar file.

Only users in the **IntegrationDeployer** or **IntegrationAdministrator** roles can configure custom token authentication. Users in the **IntegrationOperator** or **IntegrationMonitor** roles have read-only access to this configuration.

Understanding Transport-Level Custom Authentication

You can authenticate client requests at the transport-level via custom authentication tokens. You specify a custom token in an HTTP header. The HTTP and HTTPS-specific configuration pages of the service definition wizard allows you to configure client authentication. The options for HTTP and HTTPS proxy services are:

- None
- Basic
- Custom Authentication
- Client Certificate (HTTPS Only)

These are mutually exclusive options.

If you choose custom authentication, you must also specify the name of the HTTP header that is to carry the token, and the token type.

The steps for configuring transport-level custom credentials are described in "Adding a Proxy Service" under Proxy Services in *Using the AquaLogic Service Bus Console*.

The custom authentication token can be any active token type, previously configured for an Identity Assertion provider, that is carried in an HTTP header.

You need to configure, or create and configure, an Identity Assertion provider that handles the token type you plan to use. See "Configuring Identity Assertion Providers for Custom Tokens" on page 5-6.

After you have configured the transport-level custom credentials, you can then additionally configure the message level security configuration, as described in "Configuring Message-Level Security for Web Services" on page 6-1.

Importing and Exporting and Transport-Level Custom Token Authentication

Transport-level custom authentication tokens are published to the UDDI. The `client-auth` property is present in the `instanceParms` of the HTTP or HTTPS transport attributes whenever authentication is configured. As described in the transport attributes table of the User Guide, the possible values of client-auth are `BASIC`, `CLIENT-CERT` and `CUSTOM-TOKEN`. Whenever the value is `CUSTOM-TOKEN`, two additional properties are present: `token-header` and `token-type`.

Note: AquaLogic Service Bus business service definitions do not support custom token authentication. If you import a service from UDDI that has client-auth equal to CUSTOM-TOKEN, the service is imported as if it does not have any authentication configuration.

Understanding Message-Level Custom Authentication

AquaLogic Service Bus supports client-specified custom authentication credentials for inbound message-level requests. The custom authentication credentials can be in the form of a custom token, or a username and password.

AquaLogic Service Bus accepts and attempts to authenticate a custom token passed to a proxy service in a SOAP header (for SOAP-based proxy services), or in the payload (for non-SOAP proxy services). You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the token is passed, and the token type.

AquaLogic Service Bus also accepts and attempts to authenticate a username and password token passed in a SOAP header (for SOAP based proxy services), or in the payload for non-SOAP proxy services. You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the username and password are passed.

The following inbound message-level authentication mechanisms are now supported:

- For SOAP-based proxy services
  - Custom token in a SOAP header
  - Username/password in a SOAP header
- For non-SOAP-based proxy services
  - Custom token in the payload of any XML-based proxy services
  - Username/password in the payload of any XML-based proxy services

Message-level custom tokens and message-level username and password are supported on proxy services of the following binding types:

- WSDL-SOAP
- WSDL-XML
- Abstract SOAP
- Abstract XML
- Mixed – XML (in the request)
- Mixed – MFL (in the request)

Format of XPath Expressions

The configuration for both custom username/password and custom token is similar. In both cases, you specify XPath expressions that enable AquaLogic Service Bus to locate the necessary information. The root of these XPath expressions is as follows:

- Use `soap-env:Envelope/soap-env:Header` if the service binding is anySOAP or WSDL-SOAP.
- Use `soap-env:Body` (specifically, the contents of the $body variable) if the service binding is not SOAP based.

Note: All XPath expressions must be in a valid XPath 2.0 format. The XPath expressions must use the XPath “declare namespace” syntax to declare any namespaces used, as follows:

```
declare namespace
ns='http://webservices.mycompany.com/MyExampleService';
```

For example,

```
declare namespace y="http://foo";./y:my-custom-token/text()
```

Configuring Identity Assertion Providers for Custom Tokens

An Identity Assertion provider is a specific form of Authentication provider that allows users or system processes to assert their identity using tokens. A client's identity is established through the use of client-supplied tokens. The Identity Assertion provider validates the token. If the token is successfully validated, the Identity Assertion provider maps the token to an AquaLogic Service Bus username, and returns the username. Identity is said to be "asserted" when the token is mapped to the username. AquaLogic Service Bus then uses this user name to establish a security context for the client.

If you want the proxy service to consume a custom token, check the provided WebLogic Server Identity Assertion providers to see if one meets your needs. WebLogic Server includes a broad array of Identity Assertion providers, including the following:

- The *WebLogic Identity Assertion provider* validates X.509 and IIOP-CSIv2 tokens and optionally can use a user name mapper to map that token to a user.
- The *Negotiate Identity Assertion provider*, which uses Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to users.

- The *SAML Identity Assertion provider*, which acts as a consumer of SAML security assertions.

If you want the AquaLogic Service Bus proxy service to consume a custom token that is not handled by one of the bundled Identity Assertion providers, for example a `secure-token-xyz` token, you (or a third-party) must first write a WebLogic Server Identity Assertion provider that supports the token type and use the WebLogic Server Administration Console to add that provider to the security realm.

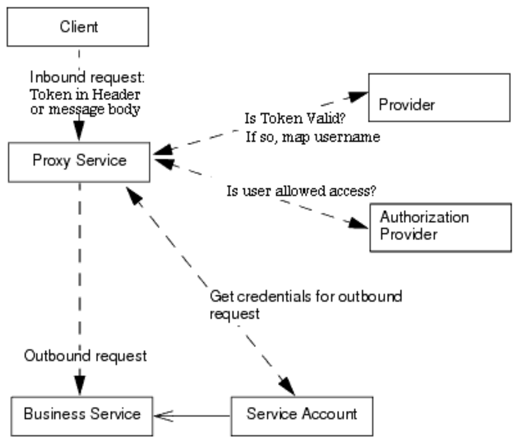
You develop Identity Assertion providers to support the specific types of custom tokens that you will be using to assert the identities of users. You can develop an Identity Assertion provider to support multiple token types. While you can have multiple Identity Assertion providers in a security realm with the ability to validate the same token type, only one Identity Assertion provider can actually perform this validation.

The Identity Assertion process is shown in Figure 5-1, and works as follows:

1. The proxy service gets the authentication token from the inbound request.
2. The token is passed to an Identity Assertion provider that is responsible for validating tokens of that type and that is configured as "active."
3. The Identity Assertion provider validates the token.
4. If the token is successfully validated, the Identity Assertion provider maps the token to a username, and returns the username.
5. AquaLogic Service Bus then continues the authentication process with this username and, if successful, obtains the authenticated subject.
6. AquaLogic Service Bus creates the security context. The security context established by authenticating a custom token or username and password can be used as the basis for outbound credential mapping and access control.

See Identity Assertion and Tokens in *Understanding WebLogic Security* for additional information.

Figure 5-1 Identity Assertion and Custom Tokens


Client
Inbound request: Token in Header or message body
Proxy Service
Is Token Valid? If so, map username
Provider
Is user allowed access?
Authorization Provider
Get credentials for outbound request
Outbound request
Business Service
Service Account

Object Type of Custom Tokens

For transport-level identity assertion, the header value is passed as a `java.lang.String` to the identity assertion providers. For message-level identity assertion, the XPath expression is evaluated as follows:

- If the XPath expression returns multiple nodes, an error is raised and identity assertion is not called.
- If the XPath expression returns an empty result, identity assertion is called with a null argument.
- If the XPath expression returns a single token of type TEXT or ATTR (See `XmlCursor.TokenType` at http://xmlbeans.apache.org/docs/2.0.0/reference/org/apache/xmlbeans/XmlCursor.TokenType.html), the string value of the text node or attribute is passed (as returned by `XmlCursor.getStringValue()`). Otherwise, a single `XmlObject` is passed.

Configuring a Custom Token Type in an Identity Assertion Provider

The steps required to complete these tasks are described in detail in the following WebLogic Server documents:

- Developing Security Providers for WebLogic Server describes how to create custom token types for an Identity Assertion provider in How to Create New Token Types.
- Securing WebLogic Server describes how to configure Identity Assertion providers in the WebLogic Server Administration Console.

For your convenience, the steps for creating custom token types for an Identity Assertion provider and configuring that provider in the WebLogic Server Administration Console are briefly listed here. However, you will need to consult the WebLogic Server documentation to actually complete the tasks.

Steps for Configuring a Custom Token Type in an Identity Assertion Provider

You can develop a custom Identity Assertion provider by following these steps:

1. Create the New Token Types
2. Create Runtime Classes Using the Appropriate SSPIs. Listing 5-4 from that section shows the `SampleIdentityAsserterProviderImpl.java` class, which is the runtime class for the sample Identity Assertion provider.
3. Generate an MBean Type Using the WebLogic MBeanMaker.
4. Configure the Custom Identity Assertion Provider Using the Administration Console.
5. Define the active token type. For this task, see Configuring Identity Assertion Providers and How to Make New Token Types Available for Identity Assertion Provider Configurations.

Setting the Supported and Active Types in the MBean

When you configure a custom Identity Assertion provider (see Configure the Custom Identity Assertion Provider Using the Administration Console), the **Supported Types** field displays a list of the token types that the Identity Assertion provider supports. You enter zero or more of the supported types in the **Active Types** field, as shown in Figure 5-1 from that section.

The content for the **Supported Types** field is obtained from the **SupportedTypes** attribute of the MBean Definition File (MDF), which you use to generate your custom Identity Assertion provider's MBean type. An example from the sample Identity Assertion provider is shown in Listing 5-1. (For more information about MDFs and MBean types, see Generate an MBean Type Using the WebLogic MBeanMaker.)

Listing 5-1 SampleIdentityAsserter MDF: SupportedTypes Attribute

```
<MBeanType>
...
<MBeanAttribute
Name = "SupportedTypes"
Type = "java.lang.String[]"
Writeable = "false"
Default = "new String[] {&quot;SamplePerimeterAtnToken&quot;}"
/>
...
</MBeanType>
```

Similarly, the content for the **Active Types** field is obtained from the **ActiveTypes** attribute of the MBean Definition File (MDF). You can default the **ActiveTypes** attribute in the MDF so that it does not have to be set manually with the WebLogic Server Administration Console. An example from the sample Identity Assertion provider is shown in Listing 5-2.

Listing 5-2 SampleIdentityAsserter MDF: ActiveTypes Attribute with Default

```
<MBeanAttribute
Name= "ActiveTypes"
Type= "java.lang.String[]"
Default = "new String[] { &quot;SamplePerimeterAtnToken&quot; }"
/>
```

While defaulting the **ActiveTypes** attribute is convenient, you should only do this if no other Identity Assertion provider will ever validate that token type. Otherwise, it would be easy to configure an invalid security realm (where more than one Identity Assertion provider attempts to validate the same token type). Best practice dictates that all MDFs for Identity Assertion providers turn off the token type by default; then an administrator can manually make the token type active by configuring the Identity Assertion provider that validates it.

Additional Context Properties for Message-Level Authentication

Both custom username/password authentication and custom token authentication allow users (who are in the **IntegrationAdmin** or IntegrationDeployer roles) to pass additional context information to the security provider in the **Context Properties** field on the **Message Level Security Configuration** page.

Context Properties provides a way (the ContextHandler interface) to pass additional information to the WebLogic Security Framework so that a security provider can obtain contextual information beyond what is provided by the arguments to a particular provider method. A ContextHandler is a high-performing WebLogic class that obtains additional context and container-specific information.

You can configure additional context properties by entering the **Property Name** as a literal string, and the **Value Selector** as a valid XPath expression. (XPath expressions can also be literal strings.)

The XPath expression is evaluated at runtime against the same message part that is used for the custom token or custom username/password. That is, the **Value Selector** XPath expressions are evaluated against the header for SOAP-based proxy services, and against the body for non-SOAP-based proxy services.

Security Provider Must Have Knowledge of the Property Name

A ContextHandler is essentially a name/value list and, as such, it requires that a security provider know what names to look for. Therefore, for both transport- and message-level custom authentication, the XPath expressions are evaluated only if an Authentication provider or Identity Assertion provider asks for the value of one of these properties.

This means that your configured Authentication or Identity Assertion provider must explicitly know which property names to request via the `ContextHandler.getValue(propertyName)`method. The only way to satisfy this requirement is for you, or a third party, to write a custom Authentication or Identity Assertion provider.

For example, Listing 5-3 shows how to get the HttpServletRequest property from a provider that you write.

Listing 5-3 Getting the HttpServletRequest Property

```
:
Object requestValue =
handler.getValue("com.bea.contextelement.alsb.transport.http.http-request"
);
if ((requestValue == null) || (!(requestValue instanceof
HttpServletRequest)))
return;
HttpServletRequest request = (HttpServletRequest) requestValue;
log.println(" " + HTTP_REQUEST_ELEMENT + " method: " + request.getMethod());
log.println(" " + HTTP_REQUEST_ELEMENT + " URL: " +
request.getRequestURL());
log.println(" " + HTTP_REQUEST_ELEMENT + " URI: " +
request.getRequestURI());
return;
```

If the security provider does not need the value of the user-defined property, then the XPath expression is not evaluated.

Configuring Custom Authentication Transport-Level Security

You ultimately use the Service Bus Console to configure custom authentication for transport-level security, as described on the Protocol-Dependent Transport Configuration page. However, before you get to this step of the process, you must first configure, or potentially create and configure, an Identity Assertion provider that understands the token type you plan to use.

The steps required to complete these tasks are described in detail in the following WebLogic Server documents:

- If one of the bundled Identity Assertion providers meets your needs, see Configure Identity Assertion providers for instructions on how to configure this Identity Assertion provider in the WebLogic Server Administration Console.

- Developing Security Providers for WebLogic Server describes how to create custom token types for an Identity Assertion provider in How to Create New Token Types.
- Securing WebLogic Server describes how to configure Identity Assertion providers in the WebLogic Server Administration Console.

Steps for Configuring Custom Authentication Transport-Level Security

The steps for configuring custom authentication transport-level security are as follows:

1. Determine which custom token format you will be using.
2. Determine if an existing provider meets your needs. Choosing an Authentication Provider offers guidance on this task.
3. Configure, or create and configure, an Identity Assertion provider that supports the token format.
4. The Identity Assertion provider maps the token to a username. Add the client's username to the AquaLogic Service Bus Security Configuration module.
5. On the Protocol-Dependent Transport Configuration page, specify the **Authentication Header** where AquaLogic Service Bus is to find the token and the **Authentication Token Type**. Only those token types that are currently active for a configured Identity Assertion provider are displayed.

Configuring Custom Authentication Message-Level Security

You ultimately use the Service Bus Console to configure custom authentication message-level security, as described on the Message Level Security Configuration page. However, before you get to this step of the process, you must first configure, or potentially create and configure, an Authentication provider or Identity Assertion provider that understands the token type you plan to use.

The steps required to complete these tasks are described in detail in the following WebLogic Server documents:

- If one of the bundled Authentication or Identity Assertion providers meets your needs, see Configuring Authentication Providers for instructions on how to configure this Authentication provider in the WebLogic Server Administration Console.
- Developing Security Providers for WebLogic Server describes how to create custom token types for an Identity Assertion provider in How to Create New Token Types.
- Securing WebLogic Server describes how to configure Identity Assertion providers in the WebLogic Server Administration Console.

Steps for Configuring Custom Authentication Message-Level Security

The steps for configuring custom authentication message-level security are as follows:

1. Determine which custom username/password or token format you will be using.
2. Determine if an existing provider meets your needs. Choosing an Authentication Provider offers guidance on this task.

   If you specify any **Context Properties** you will probably need to create your own provider because the provider must know which property names to expect.
3. Configure, or create and configure, an authentication provider or identity assertion provider that supports the username/password or token format, respectively. This provider must also understand any **Context Properties** that you want to provide.
4. Add the client's user name to the AquaLogic Service Bus Security Configuration module.

5. On the Message Level Security Configuration page, configure a new or existing proxy service for the **User Name XPath**, **User Password XPath**, or **Token Type** and **Token Path**, as appropriate.
6. Specify the **Property Name** and **Value Selector** of any **Context Properties** that you want to provide.

Propagating the Identity Obtained From Custom Authentication Tokens

The security context established via a custom token or custom username/password is in no way unique, and you can use it for credential mapping. If you implement both transport-level authentication and message-level authentication, the message-level security context is always used for credential mapping and identity propagation.

For example, if the proxy service authenticates the client via a secure-token-xyz token in a SOAP header, the authenticated subject is used during any mapped service account lookup. The subject is also used when generating SAML tokens on outbound messages. Java callouts can also run under the authentication context associated with a custom token or custom username/password.

If a custom username/password is used, the username/password in the custom token can be used for outbound HTTP BASIC or outbound WS-Security Username Token authentication if a pass-through service account is used.

Combining WS-Security with Custom Username/Password and Tokens

You can secure AquaLogic Service Bus proxy services with either transport-level security (for example, HTTPS) and message-level security (for example, WS-Security and custom tokens), or a combination of both. That is, you can configure an AquaLogic Service Bus proxy service with both transport-level authentication and message-level authentication.

For example, client requests can be authenticated at the transport level with custom tokens in HTTP headers, and at the message level with WSS security tokens, custom tokens, or username/passwords, except in the Web Services Security header.

However, note the following restriction: Although it is possible to combine WS-Security and message-level custom tokens, the WS-Security policy must **not** require inbound authentication based on WS-Security tokens. Message-level custom tokens and WS-Security inbound authentication are mutually exclusive.

Consider the following distinction:

- It is allowable to configure a proxy service that expects a custom token of type `MyToken` in SOAP header `<foo:MyToken>` and that has a WS-Security policy that requires signing or encryption of some message parts (for example, the `<foo:MyToken>` header and SOAP body).
- It is not allowable to configure a proxy service that requires a custom token in header `<foo:MyToken>` and that also has a WS-Security policy that requires a SAML token or any other form of authentication.

CHAPTER 6

Configuring Message-Level Security for Web Services

Message-level security applies security checks to a SOAP message after a Web services client establishes a connection with an AquaLogic Service Bus proxy service or business service and before the proxy service or business service processes the message. To provide message-level security, AquaLogic Service Bus implements the features that are defined in the OASIS standard for Web Services Security (WS-Security).

In this release of AquaLogic Service Bus, the implementation of message-level security has been expanded to include proxy services that have been configured with message-level custom authentication (either custom token or username/password). The message-level security mechanisms described in this section work alone or in concert with the message-level custom authentication mechanism, which is described in "Configuring Custom Authentication" on page 5-1. See "Combining WS-Security with Custom Username/Password and Tokens" on page 5-15 for information about using both types of security.

Inbound message-level security applies to messages between clients and AquaLogic Service Bus proxy services. It applies security to both the request from the client and the response message back to the client. **Outbound** message-level security applies to messages between AquaLogic Service Bus proxy services and SOAP-HTTP or SOAP-JMS business services. It applies security to both the request and the response.

The following sections describe configuring message-level security for a proxy service or a business service:

- "About Message-Level Security" on page 6-3
- "Configuring Inbound Message-Level Security" on page 6-5

- “Configuring Outbound Message-Level Security: Main Steps” on page 6-8

About Message-Level Security

AquaLogic Service Bus supports message-level security for SOAP messages that are sent over the HTTP, HTTPS, or JMS protocols. Usually you use message-level security in addition to the transport-level security that these protocols offer. You can require Web services clients to provide credentials at the transport level, the message level, or both levels. If you require clients to provide credentials at both levels, AquaLogic Service Bus uses the message-level credentials for inbound authentication and authorization.

With message-level security, a proxy service or business service specifies which of its operations are secured and which of the following security measures a Web services client must apply to its SOAP messages, which contain requests to invoke operations:

- Authentication

  Requires a client to present an identity that can be compared with user accounts in the domain's authentication provider.

- Message integrity through digital signatures

  Establishes the identity of the client that is requesting to invoke an operation and guarantees that no intermediary has altered the request. Also guarantees that the return values of the operation are returned to the client without being altered by an intermediary.

- Message confidentiality through XML encryption

  Encrypts the request and the return value in the response and guarantees that no intermediary has viewed the request or the response.

All of these security measures require a client to encode security tokens in its SOAP messages, and the proxy service or business service specifies which types of security tokens it requires to be encoded in the SOAP messages.

AquaLogic Service Bus supports the following WS-Security token profiles:

- *Web Services Security: Username Token Profile 1.0*, at `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf`
- *Web Services Security X.509 Token Profile 1.0*, at `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf`
- *Web Services Security SAML Token Profile 1.0*, at `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`

To send a SOAP message to a proxy service that requires message-level security, a Web services client generates a SOAP header and adds the header to the SOAP message envelope. The header includes digital signatures, security tokens, and other constructs. When the proxy service processes the secured envelope, it decrypts the message, which removes the security header. The proxy service then verifies that the message conforms to its security requirements. For example, the proxy service confirms that the required message parts were signed and/or encrypted and that the required tokens are present with the required claims.

The entire process is repeated in reverse for the response from the proxy service to the client.

For more information about WS-Security (which is the OASIS standard that defines message-level security), see *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)* at the following URL:
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`

Message-Level Access Control Policies for Proxy Services

While message integrity and message confidentiality guarantee that intermediaries do not view or modify messages, and while message authentication requires clients to prove that they are known users, they do nothing to specify **which** known users are allowed (authorized) to invoke proxy service operations.

To limit access to authorized users, you use the AquaLogic Service Bus Console to create message-level access control policies. These policies allow a proxy service to process only those SOAP messages that contain tokens that can be mapped to a specific list of authorized users.

Configuring Inbound Message-Level Security

You can configure a proxy service to support one of the following techniques for inbound message-level security:

- **Active-Intermediary**

  The proxy service processes the header in the client's SOAP messages and enforces the message-level access control policy on the messages.

  For example, a client encrypts and signs its SOAP message and sends it to a proxy service. The proxy service decrypts the message and verifies the digital signature, then routes the message. Before the proxy service sends the response back to the client, the proxy service signs and encrypts the message. The client then decrypts the message and verifies the proxy service's digital signature.

- **Pass-Through**

  Instead of processing the header in the client's SOAP messages, the proxy service passes the message untouched to a business service. Although the proxy service does not process the secured sections of the SOAP message, it can route the message based on values in the header. When the business service receives the message, it processes the security header and acts on the request. Note that the business service must use the Web Services Policy (WS-Policy) framework to describe which of its operations are secured with message-level security. The business service sends its response to the proxy service, and the proxy service passes the response untouched to the client.

  For example, the client encrypts and signs the message and sends it to the proxy service. The proxy service does not decrypt the message or verify the digital signature; it simply routes the message to the business service. The business service decrypts the messages and verifies the digital signature, and then processes the request. The response path is similar.

Creating an Active Intermediary Proxy Service: Main Steps

To create a proxy service to act as an active intermediary:

1. In a text editor or IDE, create a WSDL document to define the proxy service. To express the proxy service's message-level security requirements attach one or more Web Services Policy (WS-Policy) statements to the WSDL document.

   A WS-Policy statement is an XML statement that expresses a message-level security requirement (assertion). For example, one WS-Policy statement might contain an assertion that requires clients to supply a digital signature. AquaLogic Service Bus includes a set of WS-Policy statements that you can use.

If the AquaLogic Service Bus WS-Policy statements do not meet your security needs, you can write your own WS-Policy statements (custom WS-Policy statements) and import them into the AquaLogic Service Bus WS-Policy repository.

See "Using Web Services Policy to Specify Inbound Message-Level Security" on page 7-1.

2. In the AquaLogic Service Bus Console, import the WSDL document into the AquaLogic Service Bus WSDL repository and resolve any WSDL dependencies.

   See "Adding a WSDL" in WSDLs in the *Using the AquaLogic Service Bus Console*.

3. If you have not already configured the WebLogic security framework to support AquaLogic Service Bus, do one or more of the following depending on the whether the WSDL document contains WS-Policy statements that secure **requests** from clients to the proxy service:

   – If you want operation request policies to require authentication with a WS-Security X.509 certificate token, configure the Web Service security configuration named `__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__`. See step 2 in "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

   – If you want operation request policies to require authentication with a WS-Security Username/Password token with password digest, make sure to enable password digests. See step 5 in "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

   – If you want operation request policies to require the use of SAML tokens, you must configure a SAML asserting party for this proxy service. See "Authenticating SAML Tokens in Inbound Requests" on page 8-3.

   – If you want operation request policies to require digital signatures, register the accepted client signature verification certificates in the WebLogic Server Certificate Registry. See step 4 in "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

   – If you want operation request policies to require digital encryption, configure a proxy service provider that contains an encryption credential. The proxy service will use this credential to decrypt the encrypted SOAP message. See "Adding a Proxy Service Provider" in Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

4. In the AquaLogic Service Bus Console, do one or more of the following depending whether the WSDL document contains WS-Policy statements that secure **responses** from the proxy service to clients:
   - If any operation response policy requires digital signatures, configure a proxy service provider that contains a digital signature credential. You can create one proxy service provider that contains credentials for both encryption and digital signatures. See "Adding a Proxy Service Provider" in Proxy Service Providers in *Using the AquaLogic Service Bus Console*.
   - If any operation response policy specifies encryption, the client must send its certificate to the proxy service on the request. The proxy service will use the client's public key to encrypt its response. The client certificate must *not* be the same as the proxy service's encryption certificate.
5. In the AquaLogic Service Bus Console, create a proxy service from the WSDL that you imported in step 1. While you are creating the proxy service, do the following:
   - On the **General Configuration** page, specify the proxy service provider that you created in step 4.
   - On the **Message-Level Security Configuration** page, select the **Process WS-Security Header** check box.

   See Proxy Services in *Using the AquaLogic Service Bus Console*.
6. In the AquaLogic Service Bus Console, modify the proxy service's default message-level access control policy, which specifies conditions under which users, groups, or roles can invoke the secured operations. See "Editing Message-Level Access Policies" under Security Configuration in *Using the AquaLogic Service Bus Console*.

Creating a Pass-Through Proxy Service: Main Steps

To create a pass-through proxy service:

1. Create a business service to which the proxy service will pass the unprocessed SOAP message. The business service must be a Web service that contains WS-Policy statements.

   See "Configuring Outbound Message-Level Security: Main Steps" on page 6-8.

2. In the AquaLogic Service Bus Console, create a proxy service from a WSDL document. You can use the same WSDL document that you used for the business service that you created in step 1.

   While you are creating the proxy service:

   – Do **not** select the **Process WS-Security Header** check box on the **Message-Level Security Configuration** page.

   – Configure the proxy service to route to the business service that you created in step 1.

     If you route to the business service based on the operation that the client's SOAP message is requesting to invoke, you must configure the routing so that it specifies an operation selection algorithm other than the SOAP body algorithm. Make sure the actions in the proxy service pipeline do not modify the WS-Security header or any parts of the SOAP envelope that are signed or encrypted. Changes to clear-text message parts covered by digital signatures almost always break the digital signature because the signature cannot be verified later.

   See Proxy Services in *Using the AquaLogic Service Bus Console*.

3. In the AquaLogic Service Bus Console, modify the proxy service's default message-level access control policy, which specifies conditions under which users, groups, or roles can use the proxy service to route to the business service. See "Editing Message-Level Access Policies" under Security Configuration in *Using the AquaLogic Service Bus Console*.

Configuring Outbound Message-Level Security: Main Steps

Outbound message-level security applies to messages between AquaLogic Service Bus proxy services and SOAP-HTTP or SOAP-JMS business services. It applies security to both the request and the response.

To configure outbound message-level security for a business service that represents a SOAP-HTTP or SOAP-JMS Web service:

1. In the AquaLogic Service Bus Console, import the Web service's WSDL document into the AquaLogic Service Bus WSDL repository and resolve any WSDL dependencies.

   See "Adding a WSDL" in WSDLs in the *Using the AquaLogic Service Bus Console*.

   The WSDL document must express message-level security requirements with one or more Web Services Policy (WS-Policy) statements. If the Web service requires digital encryption, you must create a custom WS-Policy statement that embeds the encryption certificate and this WS-Policy statement must be located in the Web service's WSDL document (it cannot be included by reference).

2. In the AquaLogic Service Bus Console, do one or more of the following depending on whether the WSDL document contains WS-Policy statements that secure **requests** from a proxy service to the business service:

   – If any operation request policy includes an identity assertion with WS-Security Username Token as one of the supported token types, configure a service account for the business service. In the service account, provide the user name and password that you want the proxy service to send to the business service. Proxy services that route to this business service will get the username and password from this service account. See Service Accounts and Business Services in the *Using the AquaLogic Service Bus Console*.

   – If any operation request policy requires authentication with a WS-Security Username/Password token with password digest, make sure to enable password digests. See step 5 in "Configuring the WebLogic Security Framework: Main Steps" on page 2-21.

   – If any operation request policy requires digital signatures, configure a proxy service provider that contains a digital signature credential. You can create one proxy service provider that contains credentials for both encryption and digital signatures. See "Adding a Proxy Service Provider" in Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

3. If any operation **response** policy in the business service requires encryption (that is, the business service encrypts the response with the proxy service's encryption public key), configure a proxy service provider and assign an encryption credential to the proxy service provider. See "Adding a Proxy Service Provider" in Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

   Caution: **Encrypted back-end response messages:** If the response policy of the business service specifies encryption, the proxy service will send its encryption certificate to the business service on the request. The business service will encrypt its response using the proxy service's public key. The proxy service encryption credential must not be the same as the business service encryption credential.

4. If any policy in the business service specifies using SAML assertions, configure a WebLogic SAML Credential Mapping Provider V2 asserting party. For more information, see "Configuring SAML Credential Mapping: Main Steps" on page 8-2.

5. In the AquaLogic Service Bus Console, create a business service from the WSDL that you imported in step 1.

   See Business Services in *Using the AquaLogic Service Bus Console*.

6. Create a proxy service that routes SOAP messages to the business service. You can use either an active-intermediary proxy service or a pass-through proxy service.

   See "Creating an Active Intermediary Proxy Service: Main Steps" on page 6-5 or "Creating a Pass-Through Proxy Service: Main Steps" on page 6-7.

Disabling Outbound Message-Level Security

Some infrequently used design patterns preempt a proxy service from automatically generating the outbound WS-Security SOAP envelope and instead use an XQuery expression to create the envelope. If you use this design pattern, to prevent a proxy service from automatically generating the outbound WS-Security SOAP envelope, you must create an action in the proxy service's message flow that sets the value of the `./ctx:security/ctx:doOutboundWss` element in the `$outbound` message context variable to `xs:boolean("false")`. You can create the action in either of the following places:

- In a request stage of a pipeline pair. See "Adding a Pipeline Pair Node" under Proxy Services: Message Flow in *Using the AquaLogic Service Bus Console*.
- In a request action of a route node. See "Adding Route Node Actions" under Proxy Services: Message Flow in *Using the AquaLogic Service Bus Console*.

For information about the `$outbound` message context variable, see Message Context in *AquaLogic Service Bus User Guide*.

Under some circumstances, when you attempt to activate a session in which you have created or modified a proxy service with outbound message-level security disabled, the AquaLogic Service Bus Console reports validation errors (you cannot commit a session that contains errors). If your session validation reports errors because you have disabled outbound message-level security, modify the AquaLogic Service Bus startup command so that it sets the following system property to `true`:

```
com.bea.wli.sb.security.wss.LaxOutboundWssValidation
```

Then restart AquaLogic Service Bus. With this property set to `true`, the AquaLogic Service Bus Console reports warnings instead of errors (you can commit a session that reports warning messages).

Future releases of AquaLogic Service Bus will provide an easier way to disable outbound message-level security.

CHAPTER 7

Using Web Services Policy to Specify Inbound Message-Level Security

To express the inbound message-level security requirements for a proxy service or business service that is a Web service, you use the Web Services Policy (WS-Policy) framework. The following sections describe configuring WS-Policy for proxy services and business services:

About Web Services Policy

Web Services Policy (WS-Policy) is a standards-based framework for defining a Web service's security constraints and requirements. It expresses security constraints and requirements in a collection of XML statements called policies, each of which contains one or more assertions.

In AquaLogic Service Bus, WS-Policy assertions are used to specify a Web service's requirements for digital signatures and encryption, along with the security algorithms and authentication mechanisms that it requires.

Because the WS-Policy specification has not been fully standardized, AquaLogic Service Bus supports a WebLogic Server-proprietary format that is *based* on the assertions described in the

December 18, 2002 version of the *Web Services Security Policy Language* (WS-SecurityPolicy) specification. This release of AquaLogic Service Bus does *not* incorporate the latest update of the specification (13 July 2005). While the WS-Policy specification defines both security and reliable messaging assertions, AquaLogic Service Bus supports only security assertions. The syntax and usage of AquaLogic Service Bus security assertions differ from the WS-Policy specification, but the assertions are similar in meaning and are fully compatible with security assertions used in WebLogic Server 9.0 and 9.1 Web services.

WS-Policy policies may be included directly in a WSDL document or included by reference, and a WSDL document may import other WSDL documents that contain or refer to WS-Policy policies. An XML file that contains these policies can be used by multiple proxy services or business services.

Abstract and Concrete WS-Policy Statements

The WebLogic Web Services runtime environment recognizes two types of WS-Policy statements:

- **Concrete** WS-Policy statements specify the security tokens that are used for authentication, encryption, and digital signatures.

  You can create concrete WS-Policy statements if you know at design time the type of authentication (such as using X.509 or SAML tokens) that you want to require; whether multiple private key and certificate pairs from the keystore are going to be used for encryption and digital signatures; and so on.

- **Abstract** WS-Policy statements do not specify security tokens. Specifically, this means the `<Identity>` and `<Integrity>` elements (or assertions) of the WS-Policy files do not contain a `<SupportedTokens><SecurityToken>` child element, and the `<Confidentiality>` element WS-Policy file does not contain a `<KeyInfo><SecurityToken>` child element.

  The AquaLogic Service Bus runtime environment determines which security token types an abstract policy will accept. For information on configuring the runtime environment to accept specify types of tokens, see step 3 in "Configuring Inbound Message-Level Security" on page 6-5.

AquaLogic Service Bus WS-Policy Statements

AquaLogic Service Bus includes three XML files that contain simple, abstract WS-Policy policies:

- `Auth.xml`—contains a policy that requires Web service clients to authenticate.
- `Encrypt.xml`—contains a policy that requires clients to encrypt the SOAP body with 3DES-CBC. The key wrapping algorithm is RSA 1.5. A symmetric key for Triple DES (Data Encryption Standard) is generated by the client and encrypted for the recipient with RSA 1.5.

  You cannot use this policy with a business service. Instead, create your own concrete encryption policy. See "Creating and Using Custom WS-Policy Statements" on page 7-4.

- `Sign.xml`—contains a policy that requires clients to sign the SOAP body. It also requires that the WS-Security engine on the client add a signed timestamp to the `wsse:Security` header—which prevents certain replay attacks. All system headers are also signed. The digital signature algorithm is RSA-SHA1. Exclusive XML canonicalization is used.

  The system headers are:

  – `wsrm:SequenceAcknowledgement`
  – `wsrm:AckRequested`
  – `wsrm:Sequence`
  – `wsa:Action`
  – `wsa:From`
  – `wsa:To`
  – `wsa:FaultTo`
  – `wsa:MessageID`
  – `wsa:RelatesTo`
  – `wsa:ReplyTo`
  – `wsu:Timestamp`
  – `wsax:SetCookie`

The name space prefixes correspond to the name spaces in the following table:

Prefix	Name Space
`wsrm`	`http://schemas.xmlsoap.org/ws/2005/02/rm`
`wsa`	`http://schemas.xmlsoap.org/ws/2004/08/addressing`
`wsu`	`http://schemas.xmlsoap.org/ws/2002/07/utility`
`wsax`	`http://schemas.xmlsoap.org/ws/2004/01/addressingx`

The AquaLogic Service Bus policy files are the same policy files that WebLogic Server provides. To see contents of these XML files, see "WebLogic Server WS-Policy Files" in Configuring Security in *Programming Web Services for WebLogic Server*.

BEA recommends that you use these pre-packaged policies whenever possible. However, you can not use them under the following conditions:

- If you need to specify that particular parts of the body of a SOAP message are encrypted or digitally signed, rather than the entire body, you cannot use the AquaLogic Service Bus WS-Policy statements. Instead, create custom WS-Policy statements. See "Example: Encrypting Part of the SOAP Body and Header" on page 7-6.
- If you require clients to provide SAML tokens, you cannot use the AquaLogic Service Bus WS-Policy statements. WS-Policy statements that require SAML tokens must specify the `confirmationMethod` and therefore must be concrete.
- If you want a **business service** to require digital encryption, you cannot use the AquaLogic Service Bus WS-Policy encryption statement. Business services require concrete encryption policies (the certificate must be embedded in the policy).

For information on using these policies in your proxy services or business services, see "Attaching WS-Policy Statements to WSDL Documents" on page 7-16.

Creating and Using Custom WS-Policy Statements

If the AquaLogic Service Bus WS-Policy statements do not meet your security needs, you can write your own WS-Policy statements (custom WS-Policy statements). You cannot modify the AquaLogic Service Bus WS-Policy statements.

You can either write custom WS-Policy statements directly in your Web service's WSDL document or, if you want to reuse your statements in multiple Web services, write them in a separate XML file, import them to AquaLogic Service Bus, and refer to them from the WSDL documents.

Note the following restrictions for WS-Policy statements in AquaLogic Service Bus:

- While the WS-Policy specification allows WS-Policy statements to include both security and reliable messaging assertions, AquaLogic Service Bus supports only security assertions.
- In AquaLogic Service Bus, WS-Policies are required to have an `Id` attribute from the following name space:
  `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`

  The value of this attribute must be unique across all WS-Policy statements in the AquaLogic Service Bus domain. This attribute is optional in the WS-Policy schema but required in an AquaLogic Service Bus Web service.
- If you create a confidentiality assertion in a proxy service, it must be abstract (the certificate must not be embedded in the policy). You will get error messages while creating a proxy service that contains a concrete confidentiality assertion.
- If you create a confidentiality assertion in a business service, it must be concrete (the certificate must be embedded in the policy) and it must be located directly in the WSDL document. You cannot attach such a policy by reference. See "Example: Encryption Policy for a Business Service" on page 7-8.

Examples of Custom WS-Policy Statements

The following sections provide examples of custom WS-Policy statements:

- "Example: Encrypting Part of the SOAP Body and Header" on page 7-6
- "Example: Encryption Policy for a Business Service" on page 7-8
- "Example: Encrypting a Custom SOAP Header" on page 7-11
- "Example: Signing the Message Body and Headers" on page 7-11
- "Example: Signing a SOAP Body with SAML Holder-of-Key" on page 7-13
- "Example: Authenticating, Signing, and Encrypting a SOAP Body with SAML Sender Vouches" on page 7-14

Example: Encrypting Part of the SOAP Body and Header

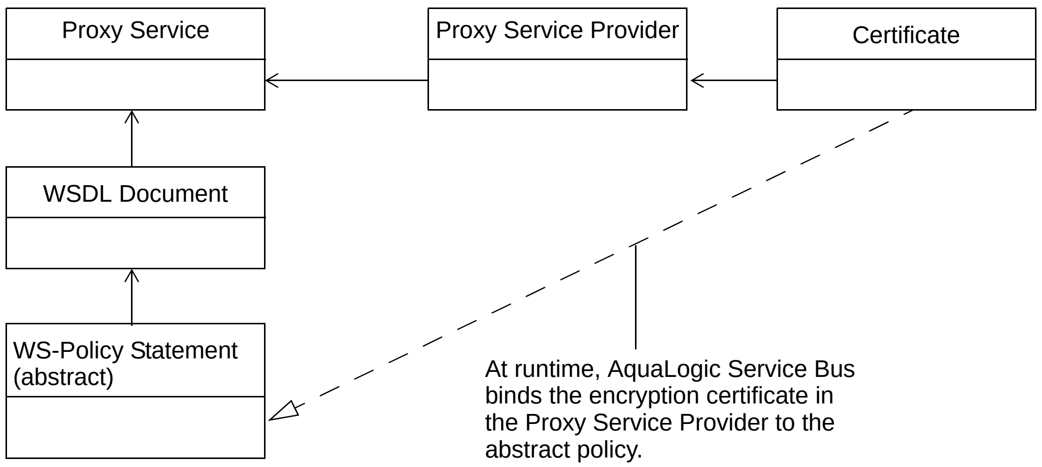
If you need to specify that particular parts of the body of a SOAP message are encrypted or digitally signed, rather than the entire body, you must create a custom WS-Policy file.

Listing 7-1 is an abstract WS-Policy statement that does the following:

- Requires the message from the client to include a user name and password token for authentication
- Requires the client to encrypt the user name token (which is in the security header)
- Requires the client to encrypt the `/definitions/message/CreditCardNumber` element

This policy cannot be used with a business service because it is abstract: its `KeyInfo` element does not contain the certificate used for encryption. Instead, when you activate a proxy service that uses this WS-Policy statement, AquaLogic Service Bus binds to the WS-Policy statement the encryption certificate from the proxy service provider that you associate with the proxy service. See Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

Figure 7-1 Binding a Certificate to an Abstract Policy


Proxy Service
Proxy Service Provider
Certificate
WSDL Document
WS-Policy Statement (abstract)
At runtime, AquaLogic Service Bus binds the encryption certificate in the Proxy Service Provider to the abstract policy.

Also in Listing 7-1:

- The `KeyWrappingAlgorithm` element specifies that the client must use the RSA 1.5 algorithm to wrap symmetric keys.
- The `EncryptionAlgorithm` specifies that the client must use the Triple DES (Data Encryption Standard) algorithm perform encrypt the security header and message body.

Listing 7-1 Encrypting Part of the SOAP Body and Header

```
<wsp:Policy
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wssp="http://www.bea.com/wls90/security/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-utility-1.0.xsd"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-secext-1.0.xsd"
   xmlns:m="http://example.org"
   wsu:Id="encrypt-custom-body-element-and-username-token">

    <!-- Require messages to provide a user name and password token
         for authentication -->
   <wssp:Identity>
      <wssp:SupportedTokens>
        <wssp:SecurityToken IncludeInMessage="true"
            TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
               wss-username-token-profile-1.0#UsernameToken">
            <wssp:UsePassword Type="http://docs.oasis-open.org/wss/2004/01/
                oasis-200401-wss-username-token-profile-1.0#PasswordText"/>
        </wssp:SecurityToken>
      </wssp:SupportedTokens>
   </wssp:Identity>

   <wssp:Confidentiality>
      <wssp:KeyWrappingAlgorithm
        URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>

      <!-- Require the user name and password in the security header
           to be encrypted -->
      <wssp:Target>
        <wssp:EncryptionAlgorithm
           URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <wssp:MessageParts
          Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
          wls:SecurityHeader(wsse:UsernameToken)
        </wssp:MessageParts>
      </wssp:Target>
      <!-- Require the /definitions/message/CreditCardNumber element to
           be encrypted -->
      <wssp:Target>
        <wssp:EncryptionAlgorithm
           URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <wssp:MessageParts>
          wsp:GetBody(.)/m:CreditCardNumber
        </wssp:MessageParts>
      </wssp:Target>

      <!-- This is an abstract policy because the KeyInfo element is
          empty. The KeyInfo data is bound to the policy at runtime -->
      <wssp:KeyInfo/>
   </wssp:Confidentiality>
</wsp:Policy>
```

Example: Encryption Policy for a Business Service

If you want messages to a business service to be encrypted, you must create a custom WS-Policy. The policy must be concrete (it must contain the encryption certificate instead of using a certificate from a proxy service provider) and it must be located directly in a WSDL document instead of being included by reference.

Typically, you would require messages to a business service to be encrypted if the proxy service that sends messages to the business service is a pass-through proxy service. That is, the proxy service that receives messages from a client does not process the SOAP message. Instead, the proxy service routes the message to the business service, and the business service takes on the responsibility of Web Services Security. See "Configuring Inbound Message-Level Security" on page 6-5.

Listing 7-2 is a WSDL document that contains a concrete policy. Note the following about this example:

- The policy requires clients to encrypt the message body.
- The `KeyInfo` element specifies the type of token that a client must provide to is the parent element that is used to describe and embed the encryption certificate. The `BinarySecurityToken` element contains the base-64 encoded encryption certificate (the value is truncated in the example). If your certificate is in PEM format, the content of the PEM file (without the PEM prefix and suffix) is the base-64 encoded representation of the certificate. If your encryption certificate is stored in a JDK keystore, you can easily export it to a PEM file.
- The policy provides a unique ID and the WSDL uses a URI fragment to refer to the ID. See "Attaching WS-Policy Statements to WSDL Documents" on page 7-16.

Listing 7-2 Encrypting the Body with a Concrete Policy, Embedding the Policy in the WSDL Document

```
<definitions name="WssServiceDefinitions"
  targetNamespace="http://com.bea.alsb/tests/wss"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  ...>

  <wsp:UsingPolicy xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
      n1:Required="true"/>

  <!-- The policy provides a unique ID -->
  <wsp:Policy wsu:Id="myEncrypt.xml">
    <wssp:Confidentiality
        xmlns:wssp="http://www.bea.com/wls90/security/policy">
    <wssp:KeyWrappingAlgorithm
        URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>

    <!-- Require the user name and password in the security header
         to be encrypted -->
    <wssp:Target>
      <wssp:EncryptionAlgorithm
         URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <wssp:MessageParts
         Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
         wsp:Body()
      </wssp:MessageParts>
    </wssp:Target>

      <!-- Embed the token type and encryption certificate -->
      <wssp:KeyInfo>
        <wssp:SecurityToken
            TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
               wss-x509-token-profile-1.0#X509v3"/>
        <wssp:SecurityTokenReference>
          <wssp:Embedded>
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
                   200401-wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
                   200401-wss-x509-token-profile-1.0#X509v3"
                 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
                   200401-wss-wssecurity-secext-1.0.xsd">
                 MIICfjCCAeegAwIBAgIQV/PDyj3...
            </wsse:BinarySecurityToken>
          </wssp:Embedded>
        </wssp:SecurityTokenReference>
      </wssp:KeyInfo>
    </wssp:Confidentiality>
  </wsp:Policy>

  <binding name="WssServiceSoapBinding" type="tns:WssService">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getPurchaseOrder">
      <soap:operation soapAction="" style="document"/>
      <input>
        <soap:body parts="parameters" use="literal"/>

        <!-- Use a URI fragment to refer to the unique policy ID -->
        <wsp:Policy>
          <wsp:PolicyReference URI="#myEncrypt.xml"/>
        </wsp:Policy>
      </input>
      <output>
          <soap:body parts="parameters" use="literal"/>
        </output>
      </operation>
    </binding>
    ...
</definitions>
```

Example: Encrypting a Custom SOAP Header

Listing 7-3 is an abstract WS-Policy statement that encrypts a custom header named `CreditCardNumber`.

Listing 7-3 Encrypting a Custom SOAP Header

```
<wsp:Policy
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wssp="http://www.bea.com/wls90/security/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-utility-1.0.xsd"
   wsu:Id="dig-sig-for-get-header">
   <wssp:Confidentiality>
     <wssp:KeyWrappingAlgorithm
       URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>

     <!-- Require the custom CreditCardNumber header to be encrypted -->
     <wssp:Target>
        <wssp:EncryptionAlgorithm
           URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <wssp:MessageParts
           Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
           wsp:GetHeader(.)/n:CreditCardNumber
         </wssp:MessageParts>
     </wssp:Target>
     <wssp:KeyInfo/>
  </wssp:Confidentiality>
</wsp:Policy>
```

Example: Signing the Message Body and Headers

Listing 7-4 is a WS-Policy statement that requires a digital signature to access the following in the SOAP message:

- A custom header named `header1`
- All system headers
- The message body

Listing 7-4 Requiring a Signature for SOAP Headers and Body

```
<wsp:Policy
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wssp="http://www.bea.com/wls90/security/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
   wssecurity-utility-1.0.xsd"
   wsu:Id="sign-custom-header-policy">

   <wssp:Integrity>
     <wssp:SignatureAlgorithm
       URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <wssp:CanonicalizationAlgorithm
       URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>

     <!-- Require the custom header header1 to be signed -->
     <wssp:Target>
       <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <wssp:MessageParts
         Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"
         xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
           wssecurity-secext-1.0.xsd"
         xmlns:n="http://example.org">
         wsp:GetHeader(.)/n:header1
       </wssp:MessageParts>
     </wssp:Target>

     <!-- Require the system headers to be signed -->
     <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
           <wssp:MessageParts
              Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
              wls:SystemHeaders()
           </wssp:MessageParts>
     </wssp:Target>

     <!-- Require the Timestamp header to be signed -->
     <wssp:Target>
        <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts
          Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
          wls:SecurityHeader(wsu:Timestamp)
        </wssp:MessageParts>
     </wssp:Target>

     <!-- Require the message body to be signed -->
     <wssp:Target>
       <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```
        <wssp:MessageParts
          Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
          wsp:Body()
        </wssp:MessageParts>
     </wssp:Target>
   </wssp:Integrity>
<wssp:MessageAge/>
</wsp:Policy>
```

Example: Signing a SOAP Body with SAML Holder-of-Key

Listing 7-5 is a WS-Policy statement that requires the SAML asserter to use the holder-of-key method to sign the message body. For information about the SAML holder-of-key method, see *Web Services Security SAML Token Profile 1.0*, at `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`.

Listing 7-5 Signing a SOAP Body with SAML Holder-of-Key Method

```
<wsp:Policy
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wssp="http://www.bea.com/wls90/security/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-utility-1.0.xsd"
   xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part"
   wsu:Id="saml-holder-of-key-signed">

  <wssp:Integrity>
     <wssp:SignatureAlgorithm
      URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <wssp:CanonicalizationAlgorithm
      URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>

     <wssp:Target>
       <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <wssp:MessageParts
         Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
         wsp:Body()
       </wssp:MessageParts>
     </wssp:Target>

     <wssp:SupportedTokens>
       <wssp:SecurityToken IncludeInMessage="true"
         TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-2004-01-saml-
         token-profile-1.0#SAMLAssertionID">
         <wssp:Claims>
          <wssp:ConfirmationMethod>holder-of-key</wssp:ConfirmationMethod>
         </wssp:Claims>
       </wssp:SecurityToken>
     </wssp:SupportedTokens>

   </wssp:Integrity>
</wsp:Policy>
```

Example: Authenticating, Signing, and Encrypting a SOAP Body with SAML Sender Vouches

Listing 7-5 is a WS-Policy statement that requires the SAML asserter to use the sender-vouches method to sign the message body and headers. For information about the SAML sender-vouches method, see *Web Services Security SAML Token Profile 1.0*, at `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`.

Listing 7-6 Signing a SOAP Body with SAML Sender-Vouches Method

```
<wsp:Policy
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wssp="http://www.bea.com/wls90/security/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-utility-1.0.xsd"
   xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part"
   wsu:Id="samlPolicy-sender-vouches-signed-encrypted">

   <wssp:Identity>
     <wssp:SupportedTokens>
       <wssp:SecurityToken
         TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-2004-01-
           saml-token-profile-1.0#SAMLAssertionID">
          <wssp:Claims>
            <wssp:ConfirmationMethod>
               sender-vouches
            </wssp:ConfirmationMethod>
          </wssp:Claims>
       </wssp:SecurityToken>
     </wssp:SupportedTokens>
   </wssp:Identity>

   <wssp:Integrity>
     <wssp:SignatureAlgorithm
```

```
        URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <wssp:CanonicalizationAlgorithm
        URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>

     <wssp:Target>
        <wssp:DigestAlgorithm
          URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <wssp:MessageParts
          Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
          wsp:Body()
        </wssp:MessageParts>
     </wssp:Target>

     <wssp:Target>
         <wssp:DigestAlgorithm
            URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <wssp:MessageParts
             Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
             wls:SecurityHeader(Assertion)
         </wssp:MessageParts>
       </wssp:Target>
   </wssp:Integrity>

   <wssp:Confidentiality>
      <wssp:KeyWrappingAlgorithm
        URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>

     <wssp:Target>
         <wssp:EncryptionAlgorithm
           URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
         <wssp:MessageParts
            Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
            wls:SecurityHeader(Assertion)
         </wssp:MessageParts>
     </wssp:Target>

     <wssp:Target>
        <wssp:EncryptionAlgorithm
          URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <wssp:MessageParts
           Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
           wsp:Body()
        </wssp:MessageParts>
     </wssp:Target>

     <wssp:KeyInfo/>
   </wssp:Confidentiality>

</wsp:Policy>
```

Attaching WS-Policy Statements to WSDL Documents

AquaLogic Service Bus implements the WS-Policy Attachment specification (http://www.w3.org/Submission/WS-PolicyAttachment/), which defines the mechanisms for associating WS-Policy statements with Web services.

To attach WS-Policy statements to a WSDL document for a Web service:

1. If you created a custom WS-Policy in a separate XML file, add the custom WS-Policy file as a resource in the AquaLogic Service Bus domain. See "Adding a Custom WS-Policy" under Custom WS-Policies in *Using the AquaLogic Service Bus Console*.
2. In the `<definitions>` element of the WSDL document, add the following child element:

   ```
   <wsp:UsingPolicy
      wsdl:Required="true"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
   ```

   The `wsdl:required="true"` attribute ensures that proxy services and business services are capable of processing the policy attachments.

   If you do not add this element, AquaLogic Service Bus ignores any WS-Policy statements in the WSDL.
3. Within each element in the WSDL document that you want to secure:

   a. Determine the URI of the WS-Policy statements that you want to use. See "Determining the URI of a WS-Policy Statement" on page 7-17.

   b. Specify the URI in the WSDL document. See "Specifying the URI of a WS-Policy Statement in a WSDL Document" on page 7-18.

Determining the URI of a WS-Policy Statement

For the AquaLogic Service Bus WS-Policy statements, the URIs are always as follows:

- `policy:Auth.xml`
- `policy:Encrypt.xml`
- `policy:Sign.xml`

For WS-Policy statements that are located directly in the WSDL document, the URI is as follows:
`#policy-ID`
where *`policy-ID`* is the value of the policy's `wsu:ID` attribute. See Listing 7-8.

For WS-Policy statements that you created in a separate XML file and added as resources to AquaLogic Service Bus, the URI is as follows:
`policy:`*`policy-ID`*
where *`policy-ID`* is the value of the policy's `wsu:ID` attribute (which you specified in the policy's XML file).

You can also use UDDI to attach WS-Policy statements to a WSDL document, in which case the URI is expressed differently. For more information, see the WS-Policy Attachment specification (http://www.w3.org/Submission/WS-PolicyAttachment/).

Specifying the URI of a WS-Policy Statement in a WSDL Document

Use one of the following techniques to specify the URI in a WSDL document:

- `PolicyURIs` attribute

  If the WSDL schema (described in http://www.w3.org/TR/wsdl) allows attribute extensibility for the element that you want secure, add the `PolicyURIs` global attribute to the element.

  For the value of this element, specify a list of URIs, each of which refers to a single policy.

  For example:
  ```
  <input message="tns:foo" wsp:PolicyURIs="policy:Sign.xml"/>
  ```

- Nested `<Policy>` element

  If the WSDL schema allows element extensibility for the element that you want to secure, add `<Policy>` as a global child element. For each WS-Policy that you want to use, add one `<PolicyReference>` element as a child of the `<Policy>` element.

  For each `<PolicyReference>` element, include a URI attribute that refers to a single policy. You can also include a digest and digest algorithm in the element.

  For example:
  ```
  <wsp:Policy>
     <wsp:PolicyReference URI="policy:Sign.xml"/>
  </wsp:Policy>
  ```

Table 7-1 lists the XPath name of WSDL elements and the technique that you use to specify the URI of the WS-Policy statement. The table also indicates the WSDL elements for which AquaLogic Service Bus does not support the attachment of WS-Policy statements.

Table 7-1 WSDL Elements That Can Be Protected in AquaLogic Service Bus

To Attach a Policy to This WSDL Element...	Use This Technique...
`/definitions/message`	Nested `<Policy>` element
`/definitions/message/part`	`PolicyURIs` attribute
`/definitions/portType`	`PolicyURIs` attribute
`/definitions/portType/operation`	Nested `<Policy>` element
`/definitions/portType/operation/input`	`PolicyURIs` attribute
`/definitions/portType/operation/output`	`PolicyURIs` attribute
`/definitions/portType/operation/fault`	AquaLogic Service Bus does not support attaching WS-Policy statements to this element
`/definitions/binding`	Nested `<Policy>` element
`/definitions/binding/operation`	Nested `<Policy>` element
`/definitions/binding/operation/input`	Nested `<Policy>` element
`/definitions/binding/operation/output`	Nested `<Policy>` element
`/definitions/binding/operation/fault`	AquaLogic Service Bus does not support attaching WS-Policy statements to this element
`/definitions/binding/service`	AquaLogic Service Bus does not support attaching WS-Policy statements to this element
`/definitions/service/port`	Nested `<Policy>` element

Best Practices: Attaching WS-Policy Statements

BEA recommends that you attach WS-Policy statements to any of the following elements or its descendants:

- `portType`
- `binding`

BEA recommends that you do not attach WS-Policy statements to the following elements:

- `service`
- `port`
- `message` or `message/part`

Example: Requiring X.509 Credentials for Identity and Confidentiality

If a WS-Policy statement requires an X.509 token for authentication, it must also require a digital signature. An X.509 token cannot satisfy an identity assertion unless the client also signs some content with the corresponding private key.

To create a proxy service that requires clients to use X.509 certificates for authentication and digital signatures, you can do the following:

1. In the WSDL document that you will use to create a proxy service, attach the AquaLogic Service Bus policies that are in the `Sign.xml` and `Auth.xml` files. See Listing 7-7.
2. Configure the proxy service to use a proxy service provider that contains an X.509 certificate for digital signatures. See Proxy Service Providers in *Using the AquaLogic Service Bus Console*.

Because the AquaLogic Service Bus `Sign.xml` and `Auth.xml` policies are abstract, they will require the client to provide the credentials that are specified in the proxy service provider that is associated with the proxy service.

Listing 7-7 shows a WSDL with references to the AquaLogic Service Bus `Sign.xml` and `Auth.xml` policies.

Listing 7-7 WSDL with Policy References to AquaLogic Service Bus WS-Policies

```
<definitions
   ...
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401
      -wss-wssecurity-utility-1.0.xsd">

   <wsp:UsingPolicy
      wsdl:Required="true"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>

   ...

   <portType name="Sample">
      <operation name="doFoo" parameterOrder="data">
         <input message="tns:foo" wsp:PolicyURIs="policy:Sign.xml"/>
         <output message="tns:fooResponse"/>
      </operation>
   </portType>

   <binding name="SampleBinding" type="tns:Sample">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="doFoo">
         <wsp:Policy>
            <wsp:PolicyReference URI="policy:Sign.xml"/>
            <wsp:PolicyReference URI="policy:Auth.xml"/>
         </wsp:Policy>
         ...
      </operation>
      </binding>

   ...

</definitions>
```

Example: Attaching Custom Inline WS-Policy Statements to a WSDL Document

Listing 7-8 shows a WSDL with two custom WS-Policy policies, `wsu:Id="policy1"` and `wsu:Id="policy2"`. The policies are located in the WSDL document; therefore the URIs that refer to these polices use XML fragments.

Listing 7-8 WSDL with Policy References to a Custom Inline Policy

```
<definitions
   ...
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
   200401-wss-wssecurity-utility-1.0.xsd">

   <wsp:UsingPolicy
      wsdl:Required="true"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>

<wsp:Policy wsu:Id="policy1">...</wsp:Policy>
<wsp:Policy wsu:Id="policy2">...</wsp:Policy>
...
   <portType name="Sample">
      <operation name="doFoo" parameterOrder="data">
         <input message="tns:foo" wsp:PolicyURIs="#policy1"/>
         <output message="tns:fooResponse"/>
      </operation>
   </portType>

   <binding name="SampleBinding" type="tns:Sample">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="doFoo">
         <wsp:Policy>
            <wsp:PolicyReference URI="#policy2"/>
         </wsp:Policy>
         <soap:operation
            soapAction="http://com.bea.samples/sample/doFoo"
            style="document"/>
         <input>
            <soap:body namespace="http://com.bea.samples/sample"
               use="literal"/>
         </input>
         <output>
            <soap:body namespace="http://com.bea.samples/sample"
```

```
              use="literal"/>
         </output>
      </operation>
   </binding>
   ...
</definitions>
```

Policy Subjects and Effective Policy

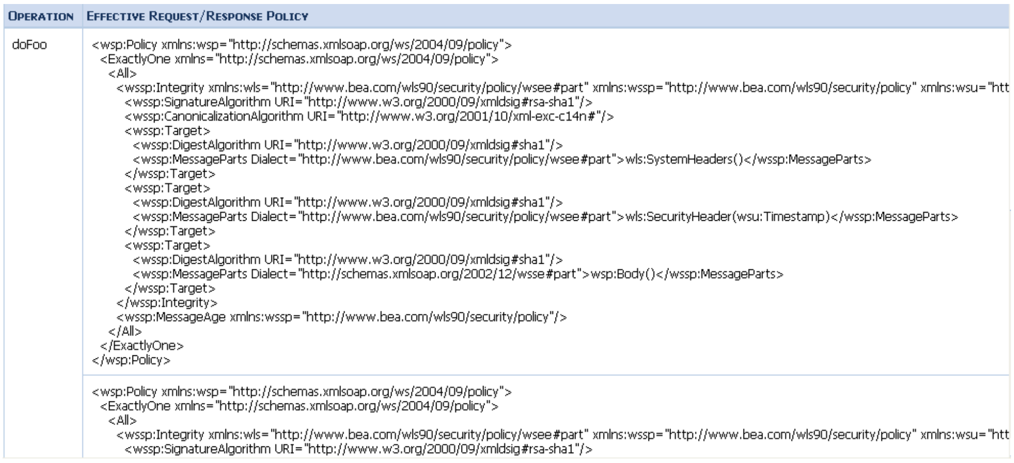
A **policy subject** is an entity, such as service, endpoint, operation, or message, with which a policy can be associated. You can associate a single WS-Policy statement with multiple policy subjects; conversely, multiple WS-Policy statements can be associated with a single policy subject. A **policy scope** is the collection of policy subjects to which a policy applies. For example, the policy scope implied by a policy attached to `wsd:binding/wsdl:operation/wsdl:input` is the input message, the operation, the endpoint, and the service.

The **effective policy** for a given policy subject is the merge of all policies whose scopes contain that policy subject. For example, the effective policy of the input message of a binding operation is the merge of all policies attached to the following:

- The input message of the binding operation
- The binding operation
- The binding
- The input message of the port-type operation
- The port-type operation
- The port-type
- The service

The AquaLogic Service Bus Console displays the effective policy (read only) when configuring a business or proxy service with WS-Policy statements, as shown in the following figure.

Figure 7-2 Effective Policy



OPERATION | EFFECTIVE REQUEST/RESPONSE POLICY

doFoo

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <ExactlyOne xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <All>
      <wssp:Integrity xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part" xmlns:wssp="http://www.bea.com/wls90/security/policy" xmlns:wsu="htt
        <wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <wssp:CanonicalizationAlgorithm URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <wssp:Target>
          <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part">wls:SystemHeaders()</wssp:MessageParts>
        </wssp:Target>
        <wssp:Target>
          <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part">wls:SecurityHeader(wsu:Timestamp)</wssp:MessageParts>
        </wssp:Target>
        <wssp:Target>
          <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <wssp:MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">wsp:Body()</wssp:MessageParts>
        </wssp:Target>
      </wssp:Integrity>
      <wssp:MessageAge xmlns:wssp="http://www.bea.com/wls90/security/policy"/>
    </All>
  </ExactlyOne>
</wsp:Policy>
```

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <ExactlyOne xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <All>
      <wssp:Integrity xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part" xmlns:wssp="http://www.bea.com/wls90/security/policy" xmlns:wsu="htt
        <wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

CHAPTER 8

Using SAML for Authentication

Security Assertion Markup Language (SAML) defines a framework for exchanging authentication and authorization information between online business partners. AquaLogic Service Bus enables the following techniques for using SAML:

- If your clients to do not provide SAML tokens but your business services require them, you can configure a proxy service to map the client's identity to a SAML token. See "Configuring SAML Credential Mapping: Main Steps" on page 8-2.
- If your clients provide SAML tokens to a pass-through proxy service, you can propagate the client's SAML token to the business service. See "Configuring SAML Pass-Through Identity Propagation" on page 8-3.
- If your clients provide SAML tokens to an active intermediary proxy service, you can configure the proxy service to assert the client's identity. See "Authenticating SAML Tokens in Inbound Requests" on page 8-3.

For an overview of SAML, see the OASIS technical overview at the following URL:

`http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf`

The complete SAML specification set of documents are available at the following URL:

`http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip`

Configuring SAML Credential Mapping: Main Steps

If your clients to do not provide SAML tokens but your business services require them, you can configure a proxy service to map the client's identity to a SAML token.

This technique requires the business service to be a Web service with WS-Policy statements that require authentication using SAML tokens.

To configure SAML credential mapping:

1. Configure a trust relationship between AquaLogic Service Bus and the system (message consumer) that the business service represents.

   The message consumer acts as a relying party and must have a trust relationship with AquaLogic Service Bus.

2. Configure the WebLogic SAML Identity Assertion Provider V2 and the WebLogic SAML Credential Mapping Provider V2 in your security domain. See Configuring a SAML Identity Assertion Provider and Configuring a SAML Credential Mapping Provider in *Securing WebLogic Server.*

3. Configure a proxy service to authenticate clients using any of the following techniques:

   – HTTP or HTTPS BASIC (client provides user name and password in the request)

   – HTTPS Client certificate

   – Message-level authentication (using any of the supported token profiles)

     If a client request includes a WS-Security security header, you must configure the proxy service to process this header on the inbound side of the message. In AquaLogic Service Bus, you cannot add a SAML header (or any other WS-Security header) to a SOAP envelope that already contains a WS-Security header, neither can you add SAML (or other) security tokens to an existing security header.

   – Third-party authentication

4. Configure the proxy service to include a SAML token in the WS-Security header of its outbound request.

   Note: If you configured the proxy service for dynamic routing, the message context determines the target URL for the request. If the assertion is signed, you must configure the certificate. For more information, see Configuring a SAML Credential Mapping Provider in *Securing WebLogic Server*.

When the proxy service sends its outbound request, it generates a SAML assertion on behalf of the client. When the business service processes the WS-Security header, it validates the SAML assertion, creates a security context for the identity in the SAML assertion, and invokes the Web service with this security context.

Configuring SAML Pass-Through Identity Propagation

If your clients provide SAML tokens to a pass-through proxy service, you can propagate the client's SAML token to the business service.

This technique requires the business service to be a Web service with WS-Policy statements that require authentication using SAML tokens.

To configure SAML pass-through identity propagation:

1. Configure a trust relationship between AquaLogic Service Bus and the back-end service.
2. Configure the back-end service acts as a SAML relying party.

   See Create a SAML Relying Party in *WebLogic Server Administration Console Online Help*.
3. Configure a pass-through proxy service.

   See "Creating a Pass-Through Proxy Service: Main Steps" on page 6-7.
4. Configure a SOAP-HTTP or SOAP-JMS business service with WS-Policy statements that require authentication using SAML tokens.

   "Configuring Outbound Message-Level Security: Main Steps" on page 6-8.

Authenticating SAML Tokens in Inbound Requests

If your clients provide SAML tokens to an active intermediary proxy service, you can configure the proxy service to assert the client's identity.

To configure a proxy service to use SAML tokens to authenticate clients:

1. Configure a trust relationship between the client software and AquaLogic Service Bus.

   AquaLogic Service Bus relies on SAML assertions issued by the client, or on behalf of the client.

2. Configure the WebLogic SAML Identity Assertion Provider V2 to validate tokens issued by the client's SAML authority. See Configuring a SAML Identity Assertion Provider in *Securing WebLogic Server.*

   When configuring the identity assertion provider, note the following requirements:

   – The confirmation method from the WS-Policy must match the SAML profile in the SAML asserting party.

   – Specify the asserting party target URL to be the relative URL of the proxy (not including the protocol and host information).

   – For signed assertions, add the certificate to the Identity Asserter registry.

3. Configure the WebLogic SAML Credential Mapping Provider V2 in your security domain. See Configuring a SAML Credential Mapping Provider in *Securing WebLogic Server.*

4. Create an active intermediary proxy service that communicates over the HTTP, HTTPS, or JMS protocol. The proxy service must be a Web service with a WS-Policy statement that requires authentication and accepts SAML tokens.

   A proxy service that communicates over the "local" transport type cannot use a SAML token profile to authenticate.

Troubleshooting SAML Web Services Security

Question: I am trying to propagate my inbound transport identity to a destination business service and keep receiving error, `Unable to add security token for identity`. What does this mean?

Answer: There are various causes for this error. Generally this means one of the following problems:

- The SAML Credential Mapper is not configured correctly. Double check that the configuration is in accordance with Configuring a SAML Credential Mapping Provider in *Securing WebLogic Server.*

- Another common source of this error is that there is no subject information to propagate. To generate a SAML token, you must have a transport-level or message-level subject. Make sure that the client has a subject. This can be done by inspecting `$security` message context variable.

Question: I am trying to propagate my inbound transport identity to a destination business service using SAML holder-of-key and keep receiving error, `Failure to add signature`. What does this mean?

Answer: There are various causes for this error, but most likely is that the credentials are not configured for the business service's proxy service provider. When AquaLogic Service Bus generates an outbound holder-of-key assertion, it generally also generates a digital signature over the message contents, so that the recipient can verify not only that a message is received from a particular user, but that the message has not been tampered with. To generate the signature, the business service must have a proxy service provider with a digital signature credential associated with it. For more information on configuring credentials, see "Adding a Credential" in Security Configuration in *Using the AquaLogic Service Bus Console*.

Question: I am trying to configure an active intermediary proxy service that receives SAML identity tokens and keep receiving errors that look like: `The SAML token is not valid`. How do I fix this?

Answer: This is generally caused by a lack of a SAML Identity Asserter or SAML Identity Asserter asserting party configuration for the proxy. For a proxy service to receive SAML assertions in active intermediary mode, it must have a SAML Identity Asserter configured. For more details, see Configuring a SAML Identity Assertion Provider in *Securing WebLogic Server*.

CHAPTER 9

Configuring Administrative Security

To give users access to administrative functions such as creating proxy services, you assign them to one of four security roles with pre-defined access privileges. A security role is an identity that can be dynamically conferred upon a user or group based on conditions that are evaluated at runtime. You cannot change the access privileges for the AquaLogic Service Bus administrative security roles, but you can change the conditions under which a user or group is in one of the roles.

The following sections describe administrative security for AquaLogic Service Bus:

- "Administrative Security Roles and Privileges" on page 9-2
- "Administrative Security Groups" on page 9-12
- "Configuring Administrative Security: Main Steps" on page 9-13

For more information about security roles, see Users, Groups, and Security Roles, in *Securing WebLogic Resources*.

Administrative Security Roles and Privileges

Table 9-1 describes the AquaLogic Service Bus administrative security roles and summarizes their access privileges.

Table 9-1 AquaLogic Service Bus Administrative Security Roles

Role	Pre-Defined Access Privileges
IntegrationAdmin and IntegrationDeployer	Has complete access to all AquaLogic Service Bus resources, including the ability to create, edit, or delete user names, passwords, and credential alias bindings in service accounts and proxy service providers. The user names and passwords that this role can create are used only by service accounts for outbound authentication; they are not used to authorize access to AquaLogic Service Bus resources. Cannot create, edit, or delete users, groups, roles, or access control policies in the Security Configuration module of the AquaLogic Service Bus Console.
IntegrationOperator	This group has the following privileges: • Has read access to all AquaLogic Service Bus resources. • Cannot export resources. • Has access to create, view, edit and delete alert rules. • Has access to session management, including create, commit, discard and undo of sessions. • Has access to create, edit, view and delete operational settings of services.
IntegrationMonitor	• Has read access to all AquaLogic Service Bus resources. • Cannot export resources.

Note: In this release, IntegrationAdministrators and IntegrationDeployers have the same privileges. This might change in future releases.

The AquaLogic Service Bus roles have permission to modify only AquaLogic Service Bus resources; they do not have permission to modify WebLogic Server or other resources on WebLogic Server. To give permission to modify WebLogic Server its other resources, add a user to one of the WebLogic Server security roles described in Table 9-2. In each AquaLogic Service Bus domain, make sure that you add at least one user to the Admin role.

Table 9-2 WebLogic Server Security Roles

WebLogic Server Role	Default Access Privileges
Admin	Has complete access to all WebLogic Server and AquaLogic Service Bus objects and functions, including the ability to create, edit, or delete users, groups, roles, or access control policies.
Deployer	Has read access to all objects. Can create, delete, edit, import or export resources, services, proxy service providers, or projects.
Operator	Has read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.
Monitor	Has read access to all objects. Can export any resource, service, proxy service provider, or project.

Role-Based Access in AquaLogic Service Bus Console

Table 9-3 shows the actions that each AquaLogic Service Bus security role can perform in the AquaLogic Service Bus Console.

Permission to perform an action is indicated by a check mark (✓) in the table. Note that there are no check marks in the Security Configuration section of this table because only the WebLogic Server Admin role has access to these functions.

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
OPERATIONS					
Monitoring					
Dashboard	View Statistics	✓	✓	✓	✓
	Reset Statistics	✓	✓	✓	
	View Alerts	✓	✓	✓	✓
	Delete Alerts	✓	✓	✓	

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
	View Alert History	✓	✓	✓	✓
	View Server Summary	✓	✓	✓	✓
Dashboard Settings	View Dashboard Settings	✓	✓	✓	✓
	Set Dashboard Settings	✓	✓	✓	✓
Configuration					
Smart Search	Set Smart Search Settings	✓	✓	✓	
	View Smart Search Settings	✓	✓	✓	✓
Global Settings	Set Global Settings	✓	✓	✓	
	View Global Settings	✓	✓	✓	✓
Tracing	Set Tracing Settings	✓	✓	✓	
	View Tracing Settings	✓	✓	✓	
Reporting					
Message Reports	View Message Reports	✓	✓	✓	✓
Purge Messages	Purge Messages	✓	✓	✓	

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
RESOURCE BROWSER					
Service					
Proxy Services	Create Proxy Service	✓	✓		
	View Proxy Service	✓	✓	✓	✓
	Edit Proxy Service	✓	✓		
	Delete Proxy Service	✓	✓		
Business Services	Create Business Service	✓	✓		
	View Business Service	✓	✓	✓	✓
	Edit Business Service	✓	✓		
	Delete Business Service	✓	✓		
Interface					
WSDLs	Create WSDLs	✓	✓		
	View WSDLs	✓	✓	✓	✓
	Edit WSDLs	✓	✓		
	Delete WSDLs	✓	✓		
XML Schemas	Create XML Schemas	✓	✓		

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
	View XML Schemas	✓	✓	✓	✓
	Edit XML Schemas	✓	✓		
	Delete XML Schemas	✓	✓		
WS-Policies	Create WS-Policy	✓	✓		
	View WS-Policy	✓	✓	✓	✓
	Edit WS-Policy	✓	✓		
	Delete WS-Policy	✓	✓		
Transformation					
XQueries	Create XQuery	✓	✓		
	View XQuery	✓	✓	✓	✓
	Edit XQuery	✓	✓		
	Delete XQuery	✓	✓		
XSLTs	Create XSLT	✓	✓		
	View XSLT	✓	✓	✓	✓
	Edit XSLT	✓	✓		
	Delete XSLT	✓	✓		
MFLs	Create MFL	✓	✓		
	View MFL	✓	✓	✓	✓
	Edit MFL	✓	✓		

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
	Delete MFL	✓	✓		
JARs	Create JARs	✓	✓		
	View JARs	✓	✓	✓	✓
	Edit JARs	✓	✓		
	Delete JARs	✓	✓		
Security					
Service Accounts	Create Service Account	✓	✓		
	View Service Account	✓	✓	✓	✓
	Edit Service Account	✓	✓		
	Delete Service Account	✓	✓		
Proxy Service Providers	Create Proxy Service Provider	✓	✓		
	View Proxy Service Provider	✓	✓	✓	✓
	Edit Proxy Service Provider	✓	✓		
	Delete Proxy Service Provider	✓	✓		
Notification					
Alert Destinations	Create Alert Rule	✓	✓	✓	

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
	View Alert Rule	✓	✓	✓	✓
	Edit Alert Rule	✓	✓	✓	
	Delete Alert Rule	✓	✓	✓	
PROJECT EXPLORER					
Projects	Create Project	✓	✓		
	View Project	✓	✓	✓	✓
	Edit Project	✓	✓		
	Delete Project	✓	✓		
Folders	Create Folder	✓	✓		
	View Folder	✓	✓	✓	✓
	Edit Folder	✓	✓		
	Delete Folder	✓	✓		
SECURITY CONFIGURATION					
Users	Create User				
	View User	✓	✓	✓	✓
	Edit User				
	Delete User				
Groups	Create Group				
	View Group	✓	✓	✓	✓
	Edit Group				

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
	Delete Group				
Roles	Create Role				
	View Role	✓	✓		
	Edit Role				
	Delete Role				
Access Control	Create Policy				
	View Policy				
	Edit Policy				
	Delete Policy				
SYSTEM ADMINISTRATION					
Import/Export					
Import Resources	Import	✓	✓		
Export Resources	Export	✓	✓		
UDDI					
UDDI Registries	Create	✓			
	View	✓	✓	✓	✓
	Edit	✓	✓		
	Delete	✓			

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Import from UDDI	Import	✓	✓		
Auto-Import Status	Synchronize	✓	✓	✓	✓
	Detach	✓	✓		
Publish to UDDI	Publish	✓	✓		
Auto-Publish Status	Auto-Publish Status	✓	✓	✓	✓
	Publish	✓	✓		
Global Resources					
JNDI Providers	Create JNDI Providers	✓	✓		
	View JNDI Providers	✓	✓	✓	✓
	Edit JNDI Providers	✓	✓		
	Delete JNDI Providers	✓	✓		
SMTP Servers	Create SMTP Servers	✓	✓		
	View SMTP Servers	✓	✓	✓	✓
	Edit SMTP Servers	✓	✓		
	Delete SMTP Servers	✓	✓		

Table 9-3 Role-Based Access in AquaLogic Service Bus Console

Console Mode	Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Customization					
Find and Replace	Find Value	✓	✓		
	Replace With	✓	✓		
Create Customizatio n File	Create File	✓	✓		
Execute Customizatio n File	Select File	✓	✓		
	Select Items	✓	✓		
	Execute File	✓	✓		
CHANGE CENTER					
Session Management	Edit Session	✓	✓	✓	
	View All Sessions	✓	✓	✓	
	View Changes	✓	✓	✓	
	Activate Changes	✓	✓	✓	
	Discard Changes	✓	✓	✓	
	Exit Session	✓	✓	✓	

Administrative Security Groups

To facilitate the process of assigning users to the pre-defined administrative roles, AquaLogic Service Bus also provides four corresponding security groups. While membership in a role is dynamic, membership in a group is static: an administrator places a user in a group and the user remains in the group until the administrator changes the assignment.

In the simplest scenario for configuring administrative security, you create a user, add the user to one of the four administrative groups, and the user is automatically always a member of the corresponding role with all of the pre-defined access privileges.

In a more complex scenario, you might create two of your own groups, MyAdministratorsEast and MyAdministratorsWest, and assign users appropriately. You configure the pre-defined IntegrationAdmin security role so that the MyAdministratorsWest group is in the role from 8am to 8pm EST, while the MyAdministratorsEast group is in the role from 8pm to 8am EST.

Table 9-4 describes the administrative groups that AquaLogic Service Bus provides. You can create your own groups in addition to these.

Table 9-4 AquaLogic Service Bus Groups

By Default, This Group...	Is Always in This Role...
IntegrationAdministrators	IntegrationAdmin. See "IntegrationAdmin and IntegrationDeployer" on page 9-2.
IntegrationDeployers	IntegrationDeployer. See "IntegrationAdmin and IntegrationDeployer" on page 9-2.
IntegrationOperators	IntegrationOperator. See "IntegrationOperator" on page 9-2.
IntegrationMonitors	IntegrationMonitor. See "IntegrationMonitor" on page 9-2.

Configuring Administrative Security: Main Steps

You can create or modify users, groups, and roles when you are in or out of an AquaLogic Service Bus session. Any additions or modifications to this data take effect immediately and are available to all sessions. If you discard a session in which you added or modified the data, the security data is **not** discarded.

To configure administrative security:

1. Log in to the AquaLogic Service Bus Console with a user account that is in the WebLogic Server Admin role.
2. (Optional) Create your own security groups.

   See "Adding a Group" under Security Configuration in the *Using the AquaLogic Service Bus Console*.
3. Create users and assign them to one of the AquaLogic Service Bus groups or one of your own groups.

   See "Adding a User" under Security Configuration in the *Using the AquaLogic Service Bus Console*.
4. (Optional) Modify the conditions under which users and groups are in the pre-defined AquaLogic Service Bus security roles.

   By default, the four default groups are always in the AquaLogic Service Bus security roles, but you can change this default. To more easily manage your list of users, BEA recommends that you never add users directly to a role. Instead, add users to a group and add the group to the role.

   See "Adding a Role" under Security Configuration in the *Using the AquaLogic Service Bus Console*.

CHAPTER 10

Securing AquaLogic Service Bus in a Production Environment

To prepare an AquaLogic Service Bus installation for production, you must pay special attention to your security needs. The following list outlines some of the tasks you need to perform:

- Read and follow the guidelines in *Securing a Production Environment* in the WebLogic Server documentation.
- Create user accounts for the AquaLogic Service Bus administrators and assign them to one or more of the following groups as appropriate: IntegrationAdministrators, IntegrationOperators, IntegrationMonitors, and IntegrationDeployers. For more information, see "Role-Based Access in AquaLogic Service Bus Console" under "Overview of Security Configuration" in Security Configuration in *Using the AquaLogic Service Bus Console*.
- In your file system, configure access control to the directory that contains AquaLogic Service Bus configuration data. This is the `sbconfig` directory under the domain root. For example:

  ```
  C:\bea\user_projects\domains\base_domain\sbconfig
  ```
- In your file system, configure access control to the directories used by the FTP, file, and email transports.
- If necessary, configure access control to the JMS resources used by your AquaLogic Service Bus installation.