



# **BEA AquaLogic Service Bus™**

## **User Guide**

# Copyright

Copyright © 2005 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, BEA JRockit, BEA Liquid Data for WebLogic, BEA WebLogic Server, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA MessageQ, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic JRockit, BEA WebLogic Log Central, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic WorkGroup Edition, BEA WebLogic Workshop, and Liquid Computing are trademarks of BEA Systems, Inc. BEA Mission Critical Support is a service mark of BEA Systems, Inc. All other company and product names may be the subject of intellectual property rights reserved by third parties.

All other trademarks are the property of their respective companies.

# Contents

## 1. Introduction

## 2. Modeling Message Flow in AquaLogic Service Bus

About AquaLogic Service Bus Message Flow .....	2-2
Pipelines .....	2-2
Message Execution .....	2-4
Building a Message Flow .....	2-4
Operational Branching .....	2-5
Performing Transformations .....	2-5
Branching in Message Flows .....	2-6
Configuring Single or Multiple Stages in Pipelines .....	2-7
Using Multiple Stages .....	2-8
Handling Errors .....	2-9
Selecting a Service Type .....	2-10
Dynamic Routing .....	2-13
Message Context .....	2-14
Key Context Manipulation Facts .....	2-16
Copying JMS Properties .....	2-17
RPC and Document Style SOAP .....	2-17
RPC Web Services .....	2-17
Document Style SOAP .....	2-21
Quality of Service .....	2-23

Delivery Guarantees . . . . .	2-23
Outbound Message Retries . . . . .	2-25
Content Types, JMS Type, and Encoding . . . . .	2-25
Asynchronous Request/Response . . . . .	2-26
JMS Correlation ID . . . . .	2-27
About the JMS Transport URI Format . . . . .	2-27

### 3. Securing Inbound and Outbound Messages

AquaLogic Service Bus Security FAQ . . . . .	3-2
About AquaLogic Service Bus Security . . . . .	3-4
WebLogic Server Prerequisites . . . . .	3-5
Transport-Level Security . . . . .	3-8
HTTPS Transport-Level Security . . . . .	3-9
Inbound HTTPS Transport-Level Security . . . . .	3-9
Outbound HTTPS Transport-Level Security . . . . .	3-11
HTTP Transport-Level Security . . . . .	3-12
Security for JMS, Email, FTP, and File Transport . . . . .	3-13
JMS Transport-Level Security . . . . .	3-13
Email and FTP Transport-Level Security . . . . .	3-15
File Transport Security . . . . .	3-16
Transport-Level Security in Message Flow and WS-Callout . . . . .	3-16
Message-Level Security . . . . .	3-16
About Message-Level Security . . . . .	3-16
Inbound Web Services Security . . . . .	3-17
About Inbound Web Services Security . . . . .	3-18
Client Request and Proxy Service Response . . . . .	3-18
Configuring Inbound Web Services Security . . . . .	3-18
Outbound Web Services Security . . . . .	3-19

About Outbound Web Services Security . . . . .	3-20
Proxy Service Request and Business Service Response . . . . .	3-20
Configuring Outbound Web Services Security . . . . .	3-20
Web Service Policy . . . . .	3-21
About Web Service Policy . . . . .	3-22
Web Services Policy Attachment . . . . .	3-23
Effective Policy . . . . .	3-25
Out-of-the-Box WS-Policy Statements . . . . .	3-26
Access Control Security . . . . .	3-28
About Access Control Security . . . . .	3-29
Users . . . . .	3-29
Groups . . . . .	3-30
Security Roles . . . . .	3-30
Credentials . . . . .	3-31
Service Accounts . . . . .	3-32
Proxy Service Providers . . . . .	3-32
Configuring Credentials . . . . .	3-33
Access Control Security Policies . . . . .	3-34
Securing AquaLogic Service Bus for a Production Environment . . . . .	3-35

## 4. Change Management and Resource Organization

Repository Users . . . . .	4-2
Projects and Folders . . . . .	4-2
Sessions . . . . .	4-2
Concurrent Modifications . . . . .	4-3
Tracking Configuration Changes . . . . .	4-4
Tracking Dependencies . . . . .	4-4
Semantic Integrity . . . . .	4-5

Undoing Modifications to Resources and Session Activations . . . . .	4-5
Undoing Modifications to Resources. . . . .	4-5
Undoing Session Activations. . . . .	4-6
Importing and Exporting Configurations . . . . .	4-6
Scripting Support. . . . .	4-7

## 5. Monitoring

Monitoring Scenarios . . . . .	5-1
About Monitoring . . . . .	5-3
Aggregation Interval . . . . .	5-3
Monitoring Architecture . . . . .	5-4
Monitoring Services. . . . .	5-6
Refresh Rate of Monitored Information. . . . .	5-7
Dashboard . . . . .	5-7
Service Summary . . . . .	5-9
About the Service Summary . . . . .	5-9
Service Monitoring Summary . . . . .	5-10
Service Monitoring Details . . . . .	5-11
Server Summary . . . . .	5-13
About the Server Summary . . . . .	5-13
Log Summary. . . . .	5-14
Server Summary. . . . .	5-17
Server Details. . . . .	5-18
Alert Summary . . . . .	5-20
About the Alert Summary . . . . .	5-20
System Alerts History . . . . .	5-22
Rule Details . . . . .	5-23
View Alert Rule Details . . . . .	5-24

Alert Rules . . . . .	5-26
About Alert Rules . . . . .	5-26
Some Uses for Alerts . . . . .	5-27
Understanding Alert Rules . . . . .	5-28

## 6. Reporting

Reporting Scenarios . . . . .	6-2
Reporting Framework . . . . .	6-3
JMS Reporting Provider . . . . .	6-4
About the JMS Reporting Provider . . . . .	6-5
How to Enable Message Reporting . . . . .	6-5
Using the Reporting Module . . . . .	6-7
Summary of Messages . . . . .	6-8
View Message Details . . . . .	6-9
Purging Messages . . . . .	6-12
Configuring a Database for the JMS Reporting Provider Store . . . . .	6-13
Configuring a Database in a Development Environment . . . . .	6-13
Configuring a Database for Production . . . . .	6-14
Removing, Stopping, or Untargeting a Reporting Provider . . . . .	6-14
Stopping a Reporting Provider when the Server is Running . . . . .	6-15
Untargeting a Reporting Provider when the Server is Running . . . . .	6-16
Untargeting the JMS Reporting Provider—Server Not Running . . . . .	6-17

## 7. Tracing

### A. Tuning AquaLogic Service Bus

### B. Debugging AquaLogic Service Bus





# Introduction

BEA AquaLogic Service Bus is part of BEA's new AquaLogic family of Service Infrastructure Products. AquaLogic Service Bus manages the routing and transformation of messages in an enterprise system. Combined with its monitoring and administration capability, AquaLogic Service Bus provides a unified software product for implementing and deploying your Service-Oriented Architecture (SOA).

AquaLogic Service Bus is a configuration-based, policy-driven Enterprise Service Bus (ESB). From the AquaLogic Service Bus Console, you can monitor your services, servers, and operational tasks. The console provides you with the ability to configure proxy and business services, set up security, manage resources, and capture data for tracking or regulatory auditing. The AquaLogic Service Bus Console enables you to respond rapidly and effectively to changes in your service-oriented environment.

AquaLogic Service Bus relies on WebLogic Server run-time facilities. It leverages WebLogic Server capabilities to deliver functionality that is highly available, scalable, and reliable.

This guide provides detailed information on using and configuring AquaLogic Service Bus. It is intended for those responsible for messaging and SOA, such as enterprise architects, operations specialists, security architects and developers, application architects and developers, server and application administrators, and support engineers.

While sometimes providing procedural information, this guide does not provide detailed information on how use the AquaLogic Service Bus Console. You can get this information from the [AquaLogic Service Bus Console Online Help](#).

The following outlines the information contained in this guide:

- [Modeling Message Flow in AquaLogic Service Bus](#)—presents guidelines to follow when you model message flows in AquaLogic Service Bus. A message flow defines the implementation of a proxy service, which is the AquaLogic Service Bus definition of an intermediary Web services that is hosted locally on AquaLogic Service Bus. In AquaLogic Service Bus, service clients exchange messages with an intermediary proxy service instead of directly with a business service.
- [Securing Inbound and Outbound Messages](#)—contains the information that you need to secure messages when using AquaLogic Service Bus, including transport configuration, Web Service Policy, access-control security, and securing AquaLogic Service Bus for a Production Environment.
- [Change Management and Resource Organization](#)—describes AquaLogic Service Bus's capabilities to manage changes and organize the resources in the repository. The resources in the AquaLogic Service Bus repository include WSDLs, Schemas, XQueries, XSLTs, MFLs, WS-Policies, Business Services, and Proxy Services.
- [Monitoring](#)—contains information about monitoring and collecting run-time information for systems operations and business auditing purposes. Describes how you can monitor the health of the system, including the state of the services, servers, and Service Level Agreement (SLA) violations.
- [Reporting](#)—provides information on how to capture message data for tracking messages or regulatory auditing. This section also contains information about setting up your own reporting provider; using the JMS reporting provider included with AquaLogic Service Bus; using the Reporting module in AquaLogic Service Bus Console; and configuring a reporting provider for alert data. Alerts contain information about SLA violations.
- [Tracing](#)—contains information on how to trace messages without having to shut down the server. This feature is useful in both a development and production environment. This feature allows you to troubleshoot and diagnose a message flow in one or more proxy services.
- [Tuning AquaLogic Service Bus](#)—this appendix provides tips on tuning AquaLogic Service Bus in a production environment.
- [Debugging AquaLogic Service Bus](#)—this appendix provides information about enabling debugging for different modules in AquaLogic Service Bus.

# Modeling Message Flow in AquaLogic Service Bus

In BEA AquaLogic Service Bus, Message Flow defines the implementation of a proxy service. This section presents some guidelines to follow when you model message flows in AquaLogic Service Bus. Configuration of AquaLogic Service Bus is performed in the AquaLogic Service Bus Console, which is described in the [AquaLogic Service Bus Console Online Help](#).

This section includes the following topics:

- [About AquaLogic Service Bus Message Flow](#)
- [Performing Transformations](#)
- [Branching in Message Flows](#)
- [Configuring Single or Multiple Stages in Pipelines](#)
- [Handling Errors](#)
- [Selecting a Service Type](#)
- [Dynamic Routing](#)
- [Message Context](#)
- [RPC and Document Style SOAP](#)
- [Quality of Service](#)
- [Content Types, JMS Type, and Encoding](#)
- [Asynchronous Request/Response](#)

## About AquaLogic Service Bus Message Flow

Message flow defines the implementation of a proxy service. Message flows are constructed by connecting together one or more instances of the following nodes types: Pipeline Pair, Route, Branch, or Echo. To learn how to implement message flow, see “Viewing and Changing Message Flow” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

This section includes the following topics:

- [Pipelines](#)
- [Message Execution](#)
- [Building a Message Flow](#)
- [Operational Branching](#)

### Pipelines

A principle component in a proxy’s implementation is the *pipeline*. A pipeline is a named sequence of stages representing a non-branching one-way processing path.

Pipelines are typed into one of three categories:

- Request—Request pipelines are used for processing the request path of the message flow.
- Response—Response pipelines are used for processing the response path of the message flow.
- Error—Error pipelines are used as error handlers.

To create the request and response paths, request and response pipelines are paired together and organized into a single node called a *pipeline pair node*. Pipeline pair nodes are organized along with two other types of nodes into a single-rooted tree structure that represents the message flow. A branch node allows you to conditionally execute these pipeline pairs, and route nodes at the ends of the branches perform the request and response dispatching. This flow structure provides a clear overview of the message flow behavior in design time, making both routes and branch conditions explicit parts of the overall design, rather than locating them out of view inside a pipeline stage or route node.

A message flow is constructed by chaining together instances of the top-level components detailed in the following table.

**Table 2-1 Message Flow Components**

Node Type	Description
Pipeline pair node	<p>A pipeline pair node ties together a single request and a single response pipeline into one top-level element. A pipeline pair node can have only 1 direct descendant in the message flow. During request processing, only the request pipeline is executed when visiting a pipeline pair node. When reversing the path for response processing, only the response pipeline is executed.</p> <p>To learn how to add a pipeline pair node, see “Adding a Pipeline Pair Node” in <a href="#">Proxy Services</a> in the <i>AquaLogic Service Bus Console Online Help</i>.</p>
Branch node	<p>A branch node enables processing to proceed down exactly one of several possible paths. Branching is driven by an XPath-based switch table. Each branch in the table specifies a condition (for example, &lt;500) that is evaluated in order against an single XPath expression (for example, <code>./ns:PurchaseOrder/ns:totalCost &lt; \$body</code>). Whichever condition is satisfied first determines which branch is followed. For more information, see <a href="#">Message Context</a> in the <i>AquaLogic Service Bus Console Online Help</i>. If no branch condition is satisfied, then the default branch which is always present is followed. A branch node may have several descendants in the message flow: one for each branch, including the default branch.</p> <p>To learn how to add a branch node, see “Adding a Conditional Branch Node” in <a href="#">Proxy Services</a> in the <i>AquaLogic Service Bus Console Online Help</i>.</p>
Route node	<p>A route node is used to perform request/response communication with another service. It represents the boundary between request and response processing for the proxy service. When the route node dispatches a request message, the request processing is considered complete. When the route node receives a response message, the response processing begins. The route node supports conditional routing as well as request and response transformations.</p> <p>As the route node represents the boundary between request and response processing, it cannot have any descendants in the message flow.</p> <p>To learn how to add a route node, see “Adding a Route Node” in <a href="#">Proxy Services</a> in the <i>AquaLogic Service Bus Console Online Help</i>.</p>
Echo node	<p>An echo node is a node that routes (or echoes) a message from the end of the request pipeline to the start of the response pipeline. In other words, the message is not routed from the proxy service to another service. It remains within the proxy service.</p>

# Message Execution

The following table demonstrates the journey of a message in a typical message flow.

Table 2-2 Message Journey

Node	What Happens to the Message?
<b>Request Processing</b>	Request processing begins at the root of the message flow.
Pipeline Pair	Executes the request pipeline only.
Branch	Evaluates the branch table and proceeds down the relevant branch.
Route	Performs the route along with any request transformations.  <b>Note:</b> Whether or not any routing is performed, the route node represents a change from request processing to response processing. When a response comes in, it reverses the path it took for the request. The same thing occurs for any request path that ends without a route node – it initiates response processing and walks back up the flow, but without waiting for any response.
<b>Response Processing</b>	See <i>Route</i> .
Branch	Continues with the node that preceded the branch.
Pipeline Pair	Executes the response pipeline.
Root of the Flow	Sends the response back to the client.
Route	Executes any response transformations.
Echo	Routes the message from the end of the request pipeline to the start of the response pipeline.

# Building a Message Flow

Any element may appear at the root of the message flow. One of the simplest of message flow designs is to have only a route node representing the entire flow. There is also no restriction on what two elements may be chained together. For example, two pipeline pair nodes may be chained together without a branching node in between. With regards to branching, each branch can start with a different element. One branch can terminate with a route node, another can be followed by a pipeline pair, and yet another may have no descendant. In the latter case, a branch

with no descendants means that response processing begins immediately. However, in general a message flow is likely to come in two forms: for non-operational services, the flow is likely to consist of a single pipeline pair at the root followed by a route node. For operational services, the flow is likely to consist of a single pipeline pair at the root, followed by a branch node based on operation, with each branch consisting of a pipeline pair followed by a route node.

## Operational Branching

Since message flow is typically used with WSDL-based services, there is frequently a need to perform processing that is operation-specific. Rather than requiring you to manually configure a branching node based on operation, AquaLogic Service Bus provides a minimal configuration branching node that automatically branches based on operation.

To learn how to add an operational branch node, see “Adding an Operational Branch Node” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Performing Transformations

This section presents guidelines to follow when you are designing transformations.

Transformation maps describe the mapping between two data types. AquaLogic Service Bus supports data mapping using XQuery and the eXtensible Stylesheet Language Transformation (XSLT) standard. XSLT maps describe XML-to-XML mappings, whereas XQuery maps can describe XML-to-XML, XML to non-XML, and non-XML to XML mappings. For more information, see [XQuery Transformations](#) and [XSL Transformations](#) in the *AquaLogic Service Bus Console Online Help*. For information on using the BEA XQuery Mapper to create XQueries, see [Transforming Data Using the XQuery Mapper](#) in the *BEA XQuery Mapper Online Help*.

The location where you perform a transformation depends on whether or not:

- The message format relies on target services. This applies when the transformation is performed in a route or a publish action.
- You perform the transformation on the response or request message regardless of the route destination.

If the message format you need to transform to or from depends on the target services, you must perform the transformation in a route action or publish action. In the case of a publish action, the transformations own a local copy of the `$outbound` variable and message-related variables (`$header`, `$body` and `$attachments`). Any changes you make to an outbound message in a publish action only affect the published message. In other words, the changes you make in the

publish action are rolled back before the message flow proceeds to any actions that follow the publish action in your message flow.

For example, when dealing with a large purchase order, it may be necessary to send a summary of the purchase order to a manager via email. A summary is created that details the main aspects of the purchase order in the SOAP body of the incoming message by including a publish action in the request pipeline. In the publish action, the data is transformed in the purchase order into a summary order and all the attachments in `$attachments` are deleted because they are not included in the summary order.

Another example is a situation in which you need to route messages to one of two possible destinations based on a WS-addressing header and the second destination requires that the document in the SOAP body must be transformed to a newer version. In this situation, configure the route node to conditionally route to one of the two destinations. Transform the document in the route action for the second destination.

You can also set the control elements in the outbound context variable (`$outbound`) to influence the behavior of the system for the outbound message (for example, you can set the Quality of Service). See “Inbound and Outbound Variables” and “Constructing Messages to Dispatch” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help* for information about the subelements of the inbound and outbound variables and how the content of messages is constructed using the values of variables in the message context.

If you need to perform a transformation on the request or response message regardless of the route destination, you should configure the transformation in the request or response pipeline stage.

For more information on pipelines, see “Pipelines” in “Overview of Message Flow” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. For more information on actions, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and for more information on route nodes, see “Adding a Route Node” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Branching in Message Flows

This section details the situations in which to use operational branching and the ones in which you can use conditional branching.

You should use operational branching to handle messages separately for each operation in situations where a proxy service is based on a Web Services Description Language (WSDL) with multiple operations. For more information, see “Viewing and Changing Operational Branch Details” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. However, if the



proxy service is not based on a WSDL and receives multiple document types, consider using a conditional branching node instead.

For example, if a proxy service is of type **Any SOAP** or **Any XML**, and you need to determine what the message type is so that you can perform conditional branching, you can use a stage action to identify the message type and use a conditional branching node in the flow to separate processing for each message type received. For more information on **Any SOAP** or **Any XML**, see “Adding a Proxy Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

You can also use conditional branching to expose the routing alternatives at the top level flow view. For example, if you invoke service A or service B based on a condition, instead of configuring conditional branching within the route node, you can expose or highlight this branching in the message flow itself and use simple route nodes as the subflows for each of the branches.

Consider your business scenario before deciding whether you configure branching in the message flow or in a stage or route node. When making your decision, remember that configuring branches in the message flow can become awkward in the design interface if the number of branches that extend from the branch node is large.

For more information, see “Overview of Message Flow” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Configuring Single or Multiple Stages in Pipelines

In most cases it is sufficient to use a single stage in a pipeline. However, some situations require the use of multiple stages. This section outlines some of the reasons why you might use multiple stages in a pipeline. For more information on configuring a stage, see “Adding a Stage” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

Stages are a container for actions and the actions can perform any of the following tasks:

- Transform the message. For more information, see [XQuery Transformations](#) and [XSL Transformations](#) in the *AquaLogic Service Bus Console Online Help*.
- Report the message. For more information, see [Reporting](#) in the *AquaLogic Service Bus Console Online Help* and [Chapter 6, “Reporting”](#).
- Perform a lookup with a `WSCallout` for message enrichment or to facilitate routing decisions. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Web Service Callout* action.

- Validate the message and raise errors. For more information, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
- Publish a copy of the message to a specified destination. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and “Constructing Messages and Dispatches” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.
- Generate a log. For more information, see “Viewing Server Log Files” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.
- Assign the result of an XQuery expression to a context variable. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Assign* action.
- Insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Insert* action.
- Validate elements selected by an XPath expression against a top level XML schema element or WSDL resource. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Validate* action.
- Rename elements selected by an XPath expression without modifying contents. For more information, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Rename* action.

## Using Multiple Stages

Consider the following characteristics of stages and actions to decide between designing multiple stages or configuring a single stage with multiple actions:

- Multiple stages provide a natural modularity compared to configuring all the actions in a single stage.
- Each stage in a request or response pipeline can have a separate error handling pipeline. Designing your pipeline with multiple stages allows you to avoid writing a single error handler that must handle all errors by all actions in a single stage. For more information, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

- If you use the resume action (in a stage level-one error handler) or the skip action, processing is resumed at the next stage in a request or response pipeline. Therefore, you must place the actions that you want performed after a resume or skip action in subsequent stages in the message flow.

For more information, see “Adding a Stage” and “Viewing and Changing Stage Configuration Details” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Handling Errors

This section presents a brief overview of how to handle errors and outlines some guidelines to consider when configuring your error handling options. For a more detailed explanation of error messages and handling, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

You can handle errors in the following ways:

- Configure a test that checks if an assertion is true and use a reply action (with failure) in the request or response pipelines.
- Catch and handle the errors at the stage level, route node level, or pipeline level. For more information, see “Adding Stage Error Handling”, “Adding Pipeline Error Handling”, and “Adding Error Handling for the Route Node” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
- Catch and handle the errors at the service level. For more information, see “Adding Error Handling for the Proxy Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
- Let the system error handler handle the error.

A predefined context variable (the fault variable) is used to hold information about any error that occurs during message processing. When an error occurs, this variable is populated with information before the appropriate error handler is invoked. The fault variable is defined only in error handler pipelines and is not set in request and response pipelines, or in route or branch nodes. To learn more about `$fault`, see “Predefined Context Variables” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

In general, it is easier to handle specific errors at the lowest level of the message flow and use higher level error handlers for more general default processing of errors that are not handled at the lower levels. It is good practice to explicitly handle anticipated errors in the pipelines and allow the service-level handler to handle unanticipated errors. However, if you decide to handle

anticipate errors in the pipelines, you can only handle WS-Security related errors at the service level.

In the event of errors for request/response type inbound messages, it is often necessary to generate a message that is sent back to the originator outlining the reason why an error occurred. You can accomplish this using a reply with failure action after configuring the message context variables with the response you want to send. For example, when a HTTP message fails, reply with failure generates the HTTP 500 status. When a JMS message fails, reply with failure sets the `JMS_BEA_Error` property to true. The AquaLogic Service Bus error actions are discussed in “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

An error handling pipeline is invoked if a service invoked by a proxy service returns a SOAP fault or transport error. Any received SOAP fault is stored in `$body`, so if a Reply with failure is executed without modifying `$body`, the original SOAP fault is returned to the client that invoked the service. If a reply action is not configured, the system error handler generates a new SOAP fault message. The proxy service recognizes that a SOAP fault is returned because a HTTP error status is set, or the JMS property `JMS_BEA_Error` is set to true.

Some use cases require error reporting. You can use the report action in these situations. For example, consider a scenario in which the request pipeline reports a message for tracking purposes, but the service invoked by the route node fails after the reporting action. In this case, the reporting system logged the message, but there is no guarantee that the message was processed successfully, only that the message was successfully received.

You can use the AquaLogic Service Bus Console to track the message to obtain an accurate picture of the message flow. This allows you to view the original reported message indicating the message was submitted for processing, and also the subsequent reported error indicating that the message was not processed correctly.

For more information, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Selecting a Service Type

AquaLogic Service Bus supports a variety of service types that range from conventional Web services (using XML or SOAP bindings in WSDLs) to non-XML or generic services. This section provides guidelines on selecting a service type.

AquaLogic Service Bus service types include:

- **SOAP Services**—SOAP services receive and respond with SOAP messages. SOAP messages are constructed by wrapping the contents of the header and body variables inside a <soap:Envelope> element.
- **XML Services (Non SOAP)**—The messages to XML-based services are XML, but can be of any type allowed by the proxy service configuration.
- **Messaging Services**—Messaging services are those that can receive messages of one data type and respond with messages of a different data type. The supported data types include XML, MFL, text, and untyped binary.

**Note:** All service types may send or receive attachments using MIME.

The following table displays the types of service types and transports supported by AquaLogic Service Bus.

**Table 2-3 Supported Service Types and Transports**

Service Type	Transport Protocols
SOAP or XML WSDL	JMS
	HTTP(S)
SOAP (no WSDL)	JMS
	HTTP(S)
XML (no WSDL) <sup>1</sup>	HTTP(S)
	JMS
	Email
	File
	FTP
Messaging Type (Binary, Text, MFL, XML)	HTTP(S)
	JMS
	Email
	File
	FTP

1. HTTP GET is only supported for XML with no WSDL.

If a service has a well defined Web Services Description Language (WSDL) interface, it is recommended, although not required, that you use the WSDL to define the service. For more WSDL information, see [WSDLs](#) in the *AquaLogic Service Bus Console Online Help*.

The benefits of using a WSDL in this situation include:

- The system can provide metrics for each operation in a WSDL.
- Operational branching is possible in the pipeline. For more information, see [“Branching in Message Flows”](#) on page 2-6.
- The `soapaction` header is automatically populated for services invoked by a proxy service.
- A WSDL is required for services using WS-Security. WS-Policies are attached to WSDLs. For more information, see [WS-Policies](#) in the *AquaLogic Service Bus Console Online Help*.
- The system supports the `<url>?WSDL` syntax, which allows you to dynamically obtain the WSDL of a HTTP proxy service. This is useful for a number of SOAP client generation tools including BEA WebLogic Workshop.
- In the XQuery and XPath editors and condition builders, it is easy to manipulate the body content variable (`$body`) because the editor provides a default mapping of `$body` to the request message in the WSDL of a proxy service. For more information, see [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

**Note:** The run-time contents of `$body` for the specific action can be different from the default mapping displayed in the editor. This is because AquaLogic Service Bus is not a programming language in which typed variables are declared and used. Instead, variables are untyped and are created dynamically at run time when a value is assigned. In addition, the type of the variable is the type that is implied by its contents at any point in the message flow. To enable you to easily create XQuery and XPath expressions, the design time editor allows you to map the type for a given variable by mapping the variable to the type in the editor.

If you decide to use a WSDL service type, it is useful to bind the service to a WSDL port instead of a binding because:

- If the service is bound to port X in the template WSDL, then port X is also defined in the generated WSDL. Any other ports defined in the template WSDL are not included in the generated WSDL. Furthermore, if you base the proxy service on a WSDL port, the generated WSDL uses that port name and preserves any WS-Policies associated with that port.

- If the service is bound to binding Y in the template WSDL, the generated WSDL defines one service and port (<service-name>QSService and <port-name>QSPort). None of the ports defined in the template WSDL are included in the generated WSDL.

You can get the WSDL for an HTTP(S)-based proxy service by entering the URL for the service appended with ?WSDL in your browser's Address field.

In the WSDL generated with the ?WSDL syntax, the port name is preserved if the proxy is bound to a port on the WSDL and the URL accurately reflects the URL of the proxy service. This may be important to some client tools. The URL in the WSDL port bound to the service is not used, except to populate the URL in the WSDL port as the default URL for a business service during service definition. You can overwrite the transport type and transport URL in the transport configuration UI for the service definition.

Any WS-Security policies at the port level apply. For more information, see “Overview of Proxy Services” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

If you want to expose one port to clients for a variety of enterprise applications, use **Any Soap** or **Any XML** service types.

If one of the request or response messages is non-XML, you must use the messaging service type.

AquaLogic Service Bus does not automatically perform “mustunderstand” SOAP header checking. However, you can use XQuery conditional expressions and validate actions to explicitly perform this type of check. For more information on the validate action, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. For more information on XQuery conditional expressions, see “Editing an XQuery Condition” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

AquaLogic Service Bus does not automatically validate the message sent or received against the service interface definition, whether it is a WSDL definition or a messaging interface definition. However, you can configure a validate action and use XQuery conditional expressions to perform validation checks explicitly in the message flow.

For more information on service types, see “Overview of Proxy Services” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Dynamic Routing

In a situation where you are designing a proxy service and do not know the concrete service to invoke, but you do know the shape of the interface, you can specify the service to be invoked directly or indirectly in the request message.

**Note:** The shape of a service refers to the abstract interface: namely message types, porttypes, and binding, and excludes the concrete interface. The concrete interface is the transport URL at which the service is located.

In this case, register a business service with the right shape (the transport URL does not matter). You can override the service URL by configuring `$outbound` with the URL of the service to invoke. This means that the URL is supplied at run time and eliminates the need to know it when you design and configure your service.

## Message Context

The message context is a set of variables that hold message context and information about messages as they are routed through the AquaLogic Service Bus. Together, the header, body, and attachments variables, (referenced as `$header`, `$body` and `$attachments` in XQuery) represent the message as it flows through AquaLogic Service Bus. The canonical form of the message is SOAP. Even if the service type is not SOAP, the message appears as SOAP in the context.

`$header` contains a SOAP Header element, `$body` contains a SOAP Body element, and `$attachments` contains a wrapper element called attachments with one child attachment element per attachment. The attachment element has a body element with the actual attachment.

When a message is received by a proxy service, the message contents are used to initialize the header, body, and attachments variables. For SOAP services, the Header and Body elements are taken directly from the envelope of the received SOAP message and assigned to `$header` and `$body` respectively. For non-SOAP services, the entire message contents are typically wrapped in a Body element and assigned to `$body`, and an empty Header element is assigned to `$header`.

Binary and MFL messages are initialized differently. For MFL messages, the equivalent XML document is injected into the Body element that is assigned to `$body`. For binary messages, the message data is stored internally and a piece of reference XML is injected into the Body element that is assigned to `$body`. The reference XML looks like `<binary-content ref="..." />`, where "..." contains a unique identifier assigned by the proxy.

The message context is defined by an XML Schema. You typically use XQuery expressions to manipulate the context variables in the message flow that defines a proxy service.

The predefined context variables provided by AquaLogic Service Bus can be grouped into the following types:

- Message-related variables
- Inbound and outbound variables



- Operation variable
- Fault variable

For information about the predefined context variables, see “Predefined Context Variables” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

The message payload is contained in the body variable. The decision about which variable’s content to include in an outgoing message is made at the point at which a message is dispatched from AquaLogic Service Bus. That determination is dependent upon whether the target endpoint is expecting a SOAP or a non-SOAP message:

- If the message is binary, any text or XML content inside the Body element in `$body` is sent.
- For MFL messages, the Body element in `$body` contains the XML equivalent of the MFL document.
- For text messages, the Body element in `$body` contains the text. For text attachments, the body element in `$attachments` contains the text. If the contents are XML instead of simple text, the XML is sent as a text message.
- For XML messages, the Body element in `$body` contains the XML. For XML attachments, the body element in `$attachments` contains the XML.
- SOAP messages are constructed by wrapping the contents of the header and body variables inside a `<soap:Envelope>` element. If the body variable contains a piece of reference XML, it is sent as is—in other words, the referenced content is not substituted into the message.

For non-SOAP services, if the Body element of `$body` contains a binary-content element, then the referenced content stored internally is sent ‘as is’, regardless of the target service type.

For more information, see [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

The types for the message context variables are defined by the message context schema (`MessageContext.xsd`). When working with the message context variables in the BEA XQuery Mapper, you need to reference `MessageContext.xsd` and the transport-specific schemas, which are available in a JAR file at the following location in your AquaLogic Service Bus installation:

```
BEA_HOME\weblogic90\servicebus\lib\sb-schemas.jar
```

where `BEA_HOME` represents the directory in which you installed AquaLogic Service Bus.

To learn about the message context schema and the transport specific schemas, see “Message Context Schema” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

## Key Context Manipulation Facts

Consider the following guidelines when you want to alter the context:

- In an XQuery expression, the root element in a variable is not present in the path when referring to an element in that variable. For example, the following XQuery expression obtains the Content-Description of the first attachment in a message:

```
$attachments/ctx:attachment[1]/ctx:Content-Description
```

To obtain the second attachment that contains, for example, a purchase order:

```
$attachments/ctx:attachment[2]/ctx:body/*
```

- A context variable is either empty or it contains a single XML element or string value. However, an XQuery expression often returns a sequence. When you use an XQuery expression to assign a value to a variable, only the first element in the sequence returned by the expression is stored as the variable value. For example, if you want to assign the value of a WS-Addressing Message ID from a SOAP header (assuming there is one in the header) to a variable named `idvar`, the assign action specification is:

```
assign data($header/wsa:MessageID) to variable idvar.
```

**Note:** In this case, if two WS-Addressing MessageID headers exist, the `idvar` variable will be assigned the value of the first one.

- The variables `$header`, `$body`, and `$attachments` are never empty. However, `$header` may contain an empty SOAP Header element, `$body` may contain an empty SOAP Body element, and `$attachments` may contain an empty attachments element.
- For many of the cases in which you use a transformation resource (XSLT or XQuery), the transformation resource is defined to transform the document in the SOAP body of a message. To make this transformation case easy and efficient, the input parameter to the transformation can be an XQuery expression. For example, you can use the following XQuery expression to feed the business document in the Body element of a message (`$body`) as input to a transformation: `$body/* [1]`. The result of the transformation can be put back in `$body` with a replace content action (replace the content of `$body` which means the content of the Body element). For more information, see [XQuery Transformations](#) and [XSL Transformations](#) in the *AquaLogic Service Bus Console Online Help*.
- As well as inserting or replacing a single element, you can also insert or replace a selected sequence of elements using an insert or replace action. You can configure an XQuery expression to return a sequence of elements. For example, you can use insert and replace actions to copy a set of transport headers from `$inbound` to `$outbound`. For information

on adding an action, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. For an example, see “[Copying JMS Properties](#)” on page 2-17.

## Copying JMS Properties

AquaLogic Service Bus assumes that the interface of the proxy services and the invoked business service may be different. Therefore, it does not propagate any information (like the transport headers and JMS properties) from the inbound variable to the outbound variable.

The transport headers for the proxy service’s request and response messages are in `$inbound` and the transport headers for the invoked business service’s request and response are in `$outbound`.

For example, the following XQuery expression can be used in a case where the user-defined JMS properties for a one way message (an invocation with no response) need to be copied from inbound to outbound:

Insert `$inbound/ctx:transport/ctx:request/tp:headers/tp:user-header` as the first child of `./ctx:transport/ctx:request/tp:headers` in the outbound variable.

To learn how to configure insert and other actions in the AquaLogic Service Bus Console, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## RPC and Document Style SOAP

AquaLogic Service Bus supports Remote Procedure Calls (RPC) style SOAP and document style SOAP. For more information, see:

- [RPC Web Services](#)
- [Document Style SOAP](#)

## RPC Web Services

You can configure proxy services as RPC style proxies and configure business services as RPC style business services.

The following listing provides an example of a WSDL for a sample RPC style Web service.

### Listing 2-1 WSDL for a Sample RPC Style Web Service

---

```
<definitions name="Lookup"
targetNamespace="http://example.com/lookup/service/defs"
```

## Modeling Message Flow in AquaLogic Service Bus

```
xmlns:tns="http://example.com/lookup/service/defs"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:docs="http://example.com/lookup/docs"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>

        <xs:schema targetNamespace="http://example.com/lookup/docs"
elementFormDefault="qualified">

            <xs:complexType name="RequestDoc">

                <xs:sequence>

                    <xs:element name="PurchaseOrg"
type="xs:string"/>

                </xs:sequence>

            </xs:complexType>

            <xs:complexType name="ResponseDoc">

                <xs:sequence>

                    <xs:element name="LegacyBoolean"
type="xs:boolean"/>

                </xs:sequence>

            </xs:complexType>

        </xs:schema>

    </types>

    <message name="lookupReq">

        <part name="request" type="docs:RequestDoc"/>

    </message>

    <message name="lookupResp">

        <part name="result" type="docs:ResponseDoc"/>

    </message>

    <portType name="LookupPortType">

        <operation name="lookup">
```

```

        <input message="tns:lookupReq"/>
        <output message="tns:lookupResp"/>
    </operation>
</portType>

<binding name="LookupBinding" type="tns:LookupPortType">
    <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="lookup">
        <soap:operation/>
        <input>
            <soap:body use="literal"
namespace="http://example.com/lookup/service"/>
        </input>
        <output>
            <soap:body use="literal"
namespace="http://example.com/lookup/service"/>
        </output>
    </operation>
</binding>
</definitions>

```

---

The above service has an operation (equivalent to a method in a java class) called `lookup`. The binding indicates that this is a SOAP RPC Web service. In other words, the Web service's operation receives a set of request parameters and returns a set of response parameters. The `lookup` operation has a parameter called `request` and a return parameter called `result`. The namespace of the operation in the binding is `"http://example.com/lookup/service"`.

When the above WSDL is used for a request, the value of the body variable (`$body`) that the SOAP RPC proxy obtains is displayed in the following listing.

**Note:** Namespace declarations have been removed from the XML in the listings that follow for the sake of clarity only.

### Listing 2-2 Body Variable Value

---

```
<soap-env:Body>
  <ns:lookup>
    <request>
      <req:PurchaseOrg>BEA Systems</req:PurchaseOrg>
    </request>
  </ns:lookup>
</soap-env:Body>
```

---

Where `soap-env` is the predefined SOAP name space, `ns` is the operation name space (`<http://example.com/lookup/service>`) and, `req` is the namespace of the `PurchaseOrg` element (`<http://example.com/lookup/docs>`).

If the business service that the proxy is routing to uses the above WSDL, the value for the body variable (`$body`) given above, is the value of the body variable (`$body`) from the proxy service.

When the above WSDL is used for a request, the value of the body variable (`$body`) for the response from the invoked business service that the proxy service receives is displayed in the following listing.

### Listing 2-3 Body Variable Value

---

```
<soap-env:Body>
  <ns:lookupResponse>
    <result>
      <req:LegacyBoolean>true</req:LegacyBoolean>
    </result>
  </ns:lookupResponse>
</soap-env:Body>
```

---

This is also the value of the body variable (`$body`) for the response returned by the proxy service using this WSDL.

There are many tools available (including BEA's WebLogic Workshop tools) that take the WSDL of a proxy service (obtained by adding the `?WSDL` suffix to the URL of the proxy in the browser) and generate a java class with the appropriate request and response parameters to invoke the operations of the service. This java class can be used to invoke the proxy service that uses this WSDL.

## Document Style SOAP

You can configure proxy services as SOAP style proxies and configure business services as SOAP style business services.

The following listing provides an example of a WSDL for a sample document style Web service.

### Listing 2-4 WSDL for a Sample Document Style Web Service

---

```
<definitions name="Lookup"
targetNamespace="http://example.com/lookup/service/defs"
xmlns:tns="http://example.com/lookup/service/defs"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:docs="http://example.com/lookup/docs"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>

        <xs:schema targetNamespace="http://example.com/lookup/docs"
elementFormDefault="qualified">

            <xs:element name="PurchaseOrg" type="xs:string"/>

            <xs:element name="LegacyBoolean" type="xs:boolean"/>

        </xs:schema>

    </types>

    <message name="lookupReq">

        <part name="request" element="docs:PurchaseOrg"/>

    </message>

    <message name="lookupResp">

        <part name="result" element="docs:LegacyBoolean"/>

    </message>
```

```
<portType name="LookupPortType">
    <operation name="lookup">
        <input message="tns:lookupReq"/>
        <output message="tns:lookupResp"/>
    </operation>
</portType>

<binding name="LookupBinding" type="tns:LookupPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="lookup">
        <soap:operation/>
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
</definitions>
```

---

The service has an operation (equivalent to a method in a java class) called `lookup`. The binding indicates that this is a SOAP document style Web service.

When the above WSDL is used for a request, the value of the body variable (`$body`) that the document style proxy obtains is displayed in the following listing.

### Listing 2-5 Body Variable Value

---

```
<soap-env:Body>
    <req:PurchaseOrg>BEA Systems</req:PurchaseOrg>
```



---

```
<soap-env:Body>
```

---

Where `soap-env` is the predefined SOAP name space and `req` is the namespace of the `PurchaseOrg` element (`<http://example.com/lookup/docs>`).

If the business service that the proxy is routing to uses the above WSDL, the value for the body variable (`$body`) given above, is the value of the body variable (`$body`) from the proxy service.

The value of the body variable (`$body`) for the response from the invoked business service that the proxy service receives is displayed in the following listing.

---

#### Listing 2-6 Body Variable Value

---

```
<soap-env:Body>
    <req:LegacyBoolean>true</req:LegacyBoolean>
</soap-env:Body>
```

---

This is also the value of the body variable (`$body`) for the response returned by the proxy service using this WSDL.

There are many tools available (including BEA's WebLogic Workshop tools) that take the WSDL of a proxy service (obtained by adding the `?WSDL` suffix to the URL of the proxy in the browser) and generate a java class with the appropriate request and response parameters to invoke the operations of the service. This java class can be used to invoke the proxy service that uses this WSDL.

## Quality of Service

This section covers the following quality of service issues:

- [Delivery Guarantees](#)
- [Outbound Message Retries](#)

### Delivery Guarantees

AquaLogic Service Bus supports reliable messaging. When messages are routed to another service from a route node, the default quality of service (QoS) is *exactly once* if the proxy transport is JMS/XA. Otherwise, it is *best-effort*. *Exactly once* reliability means that messages are delivered from inbound to outbound exactly once, assuming a terminating error does not occur

before the outbound message send is initiated. *Exactly once* delivery reliability is a hint, not a directive. When *exactly once* is specified, *exactly once* reliability is provided if possible. If *exactly once* is not possible, then *at least once* delivery semantics are attempted; if that is not possible, *best effort* delivery is performed.

*At least once* semantics means the message is delivered to the outbound from the inbound at least once, assuming a terminating error does not occur before the outbound message send is initiated. If fail over URLs are specified, *at least once* semantics are provided with respect to at least one of the URLs.

*Best effort* means that there is no reliable messaging and no elimination of duplicate messages—however, performance is optimized.

To override the quality of service attribute, you must set the `qualityOfService` in the outbound message context variable (`$outbound`). For more information, see “Message Context Schema” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

If the proxy inbound is JMS/XA and if the outbound transport for a publish action or a route action is JMS/XA, then the system ensures that the message is transferred *exactly once* from the input transport to the output transport without loss of the message or duplication of delivery. For the remainder of the transport protocols (email, FTP, file, HTTP, HTTPS, JMS/nonXA), when the proxy inbound is JMS/XA, the system ensures the message is transferred *at least once* from the input transport to the output transport without loss of the message. For more information on transport protocols, see “Adding a Proxy Service” and “Adding a Business Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

Delivery guarantees are based on JMS inbound retries. The `retrycount` and what to do if the count is exhausted can be configured in the WebLogic Server Administration Console (this cannot be configured in the AquaLogic Service Bus Console). By default, for queues created by AquaLogic Service Bus, the retry count is 1 and the message is discarded after the count is exhausted. For a list of the default JMS queues created by AquaLogic Service Bus, see the [BEA AquaLogic Service Bus Deployment Guide](#).

The guarantee of delivering the message at least once for HTTP(S) requires further explanation. Even for a case in which a target service responds (with a fault or HTTP status), it is assumed that the delivery completed. In other words, even though the target service returned an authentication error or page not found error (for example), the server was available and the service had an opportunity to process the message. However, if the proxy server or the target server crashes during message transfer, the target server is not running, or the response times out, the message is redelivered.

If you need to configure a business service or a proxy service that uses an unreliable transport like HTTP and you want more reliability, follow these guidelines:

1. For a proxy service using an unreliable transport (for example HTTP), split the service into two proxy services. The first one uses the unreliable transport whose only task is to quickly route to the second proxy service that uses a JMS/XA transport. Using this technique, the message that arrives is immediately persisted in the JMS queue by the first proxy, and the second proxy can reliably process it in the background without fear of losing the message and with a guarantee of an *at least once* semantics (or in some cases an *exactly once* semantics) with respect to the second proxy.
2. For a business service using an unreliable transport (for example HTTP), front-end it with a JMS/XA proxy service that other proxy services can route to. With this approach, the wrapper proxy service will redeliver the message to the business service for ensuring *at least once* semantics.

## Outbound Message Retries

In addition to configuring inbound retries for JMS messages, you can configure outbound retries and load balancing. Load balancing, failover, and retries work in concert to provide performance and high availability. For each message, the list of URLs you provide as failover URLs is automatically ordered based on the load balancing algorithm into a failover sequence. If the retry count is N, the entire sequence is retried N times before stopping. The system waits for the specified retry interval before commencing subsequent loops through the sequence. After completing the retry attempts, if there is still an error, the error handler pipeline for the route node is invoked. For more information on the error handler pipeline, see “Adding Pipeline Error Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

Note that for HTTP(S) transports, any HTTP status other than 200 or 202 is considered an error by AquaLogic Service Bus and must be retried. Because of this algorithm, it is possible that AquaLogic Service Bus retries errors like authentication failure that may never be rectified for that URL within the time period of interest. On the other hand, if AquaLogic Service Bus also fails over to a different URL for subsequent attempts to send a given message, the new URL may not give the error.

## Content Types, JMS Type, and Encoding

To support interoperability with heterogeneous endpoints, AquaLogic Service Bus allows you to control the content type used, the JMS type used, and the encoding used.

AquaLogic Service Bus does not make assumptions about what the external client or service needs, and uses the information configured for this purpose in the service definition. AquaLogic Service Bus derives the content type for outbound messages from the service type and interface. Content type is a part of the email and HTTP(S) protocols.

If the service type is:

- XML or SOAP (with or without a WSDL), the content type is text/XML.
- Messaging and the interface is MFL or binary, the content type is binary/octet-stream.
- Messaging and the interface is text, the content type is text/plain.
- Messaging and the interface is XML, the content type is text/XML.

You can override the content type in the outbound context variable (`$outbound`) for proxy services invoking a service, and in the inbound context variable (`$inbound`) for a proxy service response. For more information about the `$outbound` and `$inbound` context variables, see [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

Additionally, there is a JMS type, which can be byte or text. You configure the JMS type to use when you define the service in the AquaLogic Service Bus Console.

Encoding is also explicitly configured in the service definition for all outbound messages. For more information about the service definitions, see “Adding a Proxy Service” and “Adding a Business Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Asynchronous Request/Response

This section details the advantages that you gain from using asynchronous request/response messaging before briefly describing a user case scenario where it is advantageous to use asynchronous request/response messages.

Using asynchronous request/response messages has the following advantages:

- The request thread does not block waiting for the response. This removes a thread management issue that can occur when numerous blocking request/response invocations are made.
- The messaging is more reliable.

If you are using WebSphere MQ, asynchronous request/response messages may typically be the desired approach for interacting with some mainframes. You should note that the asynchronous service must echo the correlation ID. The correlation ID format used internally by AquaLogic

Service Bus is compatible with WebSphere MQ and works even if the target service is using MQ native interfaces. For more information, see [“JMS Correlation ID” on page 2-27](#).

Asynchronous request/response messages are handled by the outbound transport. That is, the message flow, except for the `$outbound` transport specific data, does not distinguish between JMS request/response and HTTP request/response.

A common use case where asynchronous request and response should be used is where the client invokes a proxy Web service via HTTP, but the backend system that is invoked by the proxy service uses JMS request/response.

## JMS Correlation ID

This section describes how you use the JMS correlation ID to link request and response messages.

You must use the JMS correlation ID to link request and response messages for business services that communicate with AquaLogic Service Bus using JMS. When you design the business service in Java, make sure that you use `getJMCCorrelationID` on the inbound message and `setJMCCorrelationID` on the outbound message before sending the JMS response to a queue or topic. For more information on configuring business services, see [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.

You can obtain the `JMCCorrelationID` when you receive a message using:

```
String getJMSCorrelationID()
```

The above method returns correlation ID values that provide specific message IDs or application specific string values.

To set the `JMSCorrelationID()` when you send a message:

```
void setJMSCorrelationID(String correlationID)
```

## About the JMS Transport URI Format

When using the JMS binding to configure a business service, the SOAP/JMS URI format you must provide in the AquaLogic Service Bus Console is:

```
jms://host:port/factoryJndiName/destJndiName
```

However, BEA WebLogic Workshop expects the following format:

```
://host:port/factoryJndiName/destJndiName?URI=/process/myprocess.jpdl
```

To overcome this problem, you must set the URI as a JMS property inside the message flow on the outbound variable (`$outbound`) before it is sent. For information about setting `$outbound`,

## Modeling Message Flow in AquaLogic Service Bus

see “Inbound and Outbound Variables” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

# Securing Inbound and Outbound Messages

This chapter provides the information that you need to secure messages when using BEA AquaLogic Service Bus. AquaLogic Service Bus makes use of the proven WebLogic Security Framework in WebLogic Server 9.0. Before reading this information, and to have a full understanding of security, BEA recommends that you read the [BEA WebLogic Server Security](#) documentation.

The intended audience for this information is Application Architects, Security Developers, Application Developers, Server Administrators, and Application Administrators. For a description of these roles, see “Document Audience” in *[Understanding WebLogic Security](#)*.

Configuration of AquaLogic Service Bus is done in the AquaLogic Service Bus Console, which is described in the *[AquaLogic Service Bus Console Online Help](#)*.

This chapter includes the following topics:

- [AquaLogic Service Bus Security FAQ](#)
- [About AquaLogic Service Bus Security](#)
- [WebLogic Server Prerequisites](#)
- [Transport-Level Security](#)
- [Message-Level Security](#)
- [Web Service Policy](#)
- [Access Control Security](#)
- [Securing AquaLogic Service Bus for a Production Environment](#)

## AquaLogic Service Bus Security FAQ

### What does AquaLogic Service Bus secure?

AquaLogic Service Bus is a messaging intermediary. As such, AquaLogic Service Bus utilizes the security features of WebLogic Server to ensure message confidentiality and integrity (message-level security); secure connections between WebLogic Server, service clients, and business services (transport-level security); and authentication and authorization (access control).

### 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 security in WebLogic Server as described in “[WebLogic Server Prerequisites](#)” on page 3-5 and “[Web Service Policy](#)” on page 3-21.

### What WebLogic security providers does AquaLogic Service Bus support?

AquaLogic Service Bus supports the security providers included with WebLogic Server, such as the WebLogic Authentication Provider, Identity Assertion Provider, Authorization Provider, and Credential Mapping Provider. However, the Security Assertion Markup Language (SAML) Identity Assertion provider is not supported in this release. It will be supported in a future release.

For more information, see “WebLogic Security Providers” in the [WebLogic Security Service Architecture](#) in *Understanding WebLogic Security*.

### Does AquaLogic Service Bus support third-party security providers?

Third-party security providers, such as Netegrity, Oracle-Oblis, and RSA have not been tested and therefore are not supported in this release of AquaLogic Service Bus. Additionally, AquaLogic Service Bus does not support third-party token handlers, such as Kerberos tokens, since they have not been tested.

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

The client identity is available to the AquaLogic Service Bus message flow in the message context. However, AquaLogic Service Bus does not propagate the client identity to the target service. This capability will be added in a future release. For more information about message identity, see “[Security for JMS, Email, FTP, and File Transport](#)” on page 3-13.

### What kind of Web Service security policies and credentials does AquaLogic Service Bus support?

AquaLogic Service Bus supports integrity and confidentiality policies through the use of digital signatures and encryption. It also supports message origin authentication through username and password tokens, X.509 certificate chains, and trusted X.509 certificates. AquaLogic Service



Bus is built on the WebLogic Security Framework. This framework implements the OASIS Web Service Security Standard, the Username Token Profile, and the X.509 Certificate Token Profile. Back-end services, such as clients and proxy services are configured with optional Web Service security policies, which are modeled after the Web Services Policy Framework (WS-Policy) specification developed by BEA, IBM, Microsoft, SAP, Sonic Software, and VeriSign. Because the WS-Policy specification has not been fully standardized, AquaLogic Service Bus supports a WebLogic Server-proprietary format. For more information, see [“Web Service Policy” on page 3-21](#).

### **Is single sign-on supported in AquaLogic Service Bus?**

Single sign-on is not relevant to securing messages. However, in a message intermediary, such as AquaLogic Service Bus, the related concept of identity propagation is relevant, but as previously mentioned, identity propagation is not supported in this release. For the AquaLogic Service Bus Console and WebLogic Server, single sign-on is supported. For more information, see “Single Sign-On” in [Security Fundamentals](#) in *Understanding WebLogic Security*.

### **What types of security should I configure?**

AquaLogic Service Bus uses the WebLogic Security Framework to configure the following types of security:

**Access Control**—Access control security involves granting an entity permissions and rights to perform certain actions on a resource. AquaLogic Service Bus supports authorization using the default WebLogic Authorization Provider. In role-based authorization, security policies define the roles that are authorized to access the resource. The AquaLogic Service Bus Console and MBean access are also secured with built-in role-based access-control security policies. For more information see, [“Access Control Security” on page 3-28](#).

**Transport-Level Security**—With transport-level security, you secure the connection over which messages are transported. HTTPS connections are secured with SSL. JMS connections can be optionally secured with T3S (T3 over SSL). SSL provides point-to-point security, it does not protect the message when intermediaries exist in the message path. FTP, email, and file transports can also be protected. For more information, see [“Transport-Level Security” on page 3-8](#).

**Message-Level Security**—Message-level security includes the security benefits of SSL, but with additional flexibility and features. Message-level security can be end-to-end, which means that a SOAP message is secure even when the transmission involves one or more intermediaries. The SOAP message itself is digitally signed and encrypted over the connection. And finally, message-level security allows you to specify that only parts of the message are signed or encrypted. Note also that SSL only works for synchronous communication, whereas message-level security can be used with asynchronous transports such as JMS. Currently, AquaLogic

Service Bus supports message-level security over HTTP, HTTPS, and JMS; other transports may be added in the future. For more information, see [“Message-Level Security” on page 3-16](#).

### **Are security errors monitored?**

Yes, for more information, see [“Service Monitoring Details” on page 5-11](#).

### **Can I configure security for MBeans?**

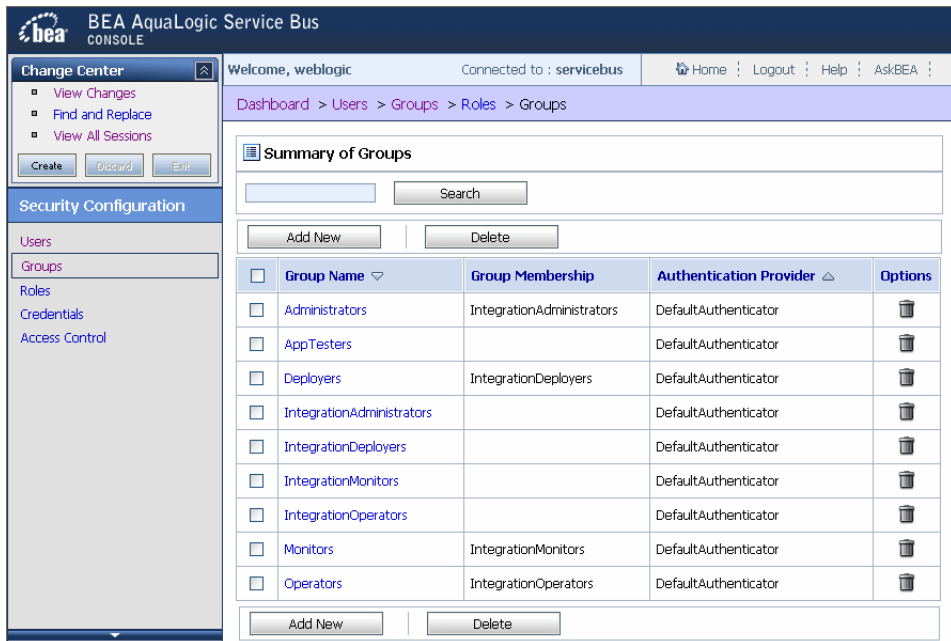
WebLogic Server enforces MBean access control. In this release of AquaLogic Service Bus, a client must be in a WebLogic Server administrator role to invoke AquaLogic Service Bus MBeans. MBean access control is not configurable in this release.

## **About AquaLogic Service Bus Security**

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.

You configure AquaLogic Service Bus security in the AquaLogic Service Bus Console (see [Figure 3-1](#)). This console provides predefined rules that make it simple to secure messages; use secure transport protocols; manage users, groups, and roles; and view and add credentials. Before configuring AquaLogic Service Bus security, you need to configure some aspects of WebLogic Server security (see [“WebLogic Server Prerequisites” on page 3-5](#)).

Figure 3-1 AquaLogic Service Bus Console

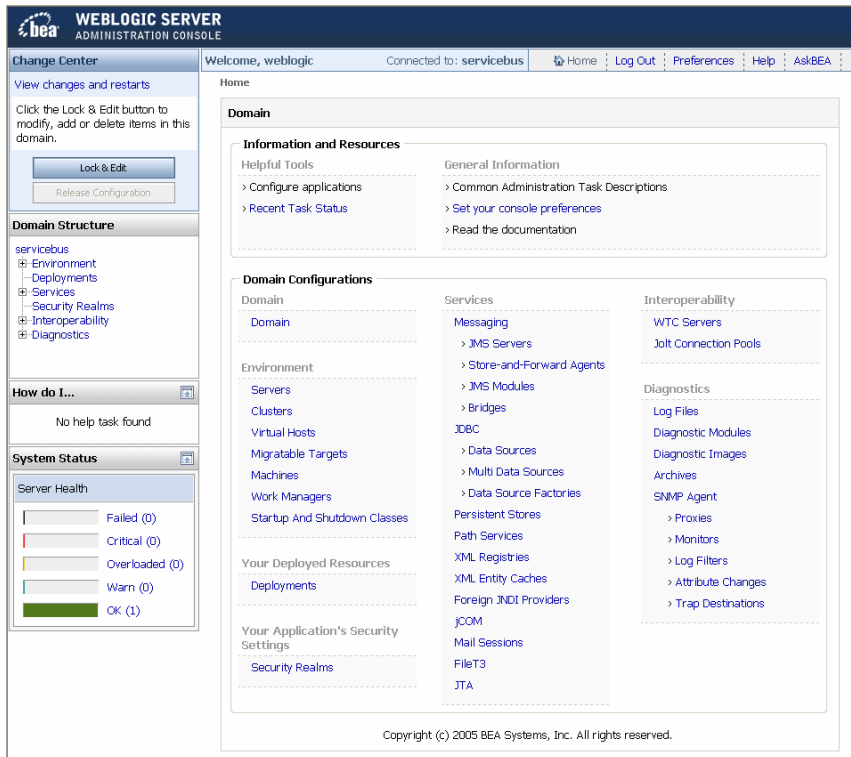


To access the AquaLogic Service Bus Console, see “Starting the BEA AquaLogic Service Bus Console” in the [Introduction](#) of the *AquaLogic Service Bus Console Online Help*.

## WebLogic Server Prerequisites

AquaLogic Service Bus leverages the WebLogic Security Framework. To take full advantage of all the security features in AquaLogic Service Bus, you must perform some security configuration in WebLogic Server before configuring security in AquaLogic Service Bus. What you need to configure in WebLogic Server depends on your business needs. To make these WebLogic Server configurations, use the WebLogic Server Administration Console. For more information, see [Securing WebLogic Server](#).

Figure 3-2 WebLogic Server Administration Console



The following lists the possible security properties that you may need to configure in WebLogic Server before configuring security in AquaLogic Service Bus. You can use the WebLogic Server Administration Console to make these changes.

- PKI Credential Mapper.** A PKI (Public Key Infrastructure) Credential Mapper is a security provider that must be configured with a keystore (with the location of the keystore relative to the domain root), keystore password, keystore type (optional), and keystore provider (optional). This keystore can be the same as the server's identity keystore or a different one. If you define a proxy service provider, you must configure a PKI credential mapper in your security realm. By default the realm configuration does not have a PKI mapper. For more information, see "Configuring a PKI Credential Mapping Provider" in [Configuring WebLogic Security Providers](#) in *Securing WebLogic Server*.

**Warning:** After adding or deleting a security provider, you must reboot the server for the security changes to take effect.

- Keystores.** Configure each WebLogic Server 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). There must be a keystore for holding the proxy service providers private-keys and certificates and a separate keystore for holding the trusted roots (trusted CA certificates). For information about configuring keystores in WebLogic Server, see [Create Keystore used by SOAP Message Digital Signatures](#) and [Create Keystore used by SOAP Message Encryption](#) in the *WebLogic Server Administration Console Online Help*.
- SSL.** If using SSL, you must configure the WebLogic Server with an SSL port, keystores for storing the server's SSL private key, digital certificates, trusted CA certificates, and the username-password to the server's SSL private key. In addition, if client certificate authentication is required, you must enable two-way SSL. When configuring two-way SSL, you must choose between two modes: *Client Certificate Requested* or *Client Certificates Requested and Enforced*. BEA recommends that you choose *Client Certificate Requested and Enforced*. For more information, see "Secure Sockets Layer (SSL)" in [Security Fundamentals](#) in *Understanding WebLogic Security*.
- Deployment.** When deploying AquaLogic Service Bus in a production environment, you need to ensure that Host Name Verification is enabled. For information about how to do this, see "Using Host Name Verification" in [Configuring SSL](#) in *Securing WebLogic Server*.
- Password Digest.** If the WebLogic Server password digest is used with username-password tokens, you must enable the password-digest for the appropriate authentication providers. This instructs the provider to store the password in encrypted form, instead of storing a hash of the password. For more information, see [Use a Password Digest in SOAP Messages](#) in the *WebLogic Server Administration Console Online Help*.
- Digest Authentication.** If password digest authentication is required with username password tokens, the `wsse:PasswordDigest` must be one of the active token types of the appropriate identity assertion provider(s). For more information, see [weblogic.security.spi.Interface IdentityAsserter](#) in the *WebLogic Server 9.0 API Reference*.
- X.509 tokens.** If X.509 tokens are used, X.509 must be one of the active token types for the appropriate identity assertion providers. In addition, you must configure a user-name mapper. X.509 tokens are required when configuring HTTPS proxies with CLIENT-CERT authentication (two-way SSL). For more information, see "Identity Assertion and Tokens" under "Authentication" in [Security Fundamentals](#) in *Understanding WebLogic Security*.
- Digital Signature.** If there is a requirement to check digital signature certificates against the certificate registry, you will have to configure a Certificate Lookup and Validation (CLV) provider with certificate registry enabled. For more information, see "Certificate Lookup and Validation" under "Identity and Trust" in [Security Fundamentals](#) in *Understanding WebLogic Security*.

- **Auditing.** If auditing is required, you must configure an auditor provider. For more information, see “Configuring a WebLogic Auditing Provider” in [Configuring WebLogic Security Providers](#) in *Securing WebLogic Server*.

**Note:** AquaLogic Service Bus supports auditing of Web Services Security (WS-Security), but does not support auditing of administrative security actions. To enable WS-Security auditing, start the server with the following system property:

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

- **Domain-Wide Web Service Security Configuration.** WS-Security is configured through `WebserviceSecurityMBeans`. Two instances of this MBean are automatically configured when you create an AquaLogic Service Bus domain using the BEA WebLogic Configuration Wizard.

```
- __SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__
```

```
- __SERVICE_BUS_OUTBOUND_WEB_SERVICE_SECURITY_MBEAN__
```

You can customize the configuration of these MBeans. For example, you can enable X.509-based WS-Security authentication (X.509 Token profile), for either inbound or outbound messages. As another example, you can enable the use of password digests with username and password tokens, as described in “Web Services Security Username Token Profile 1.0”, which is available at the following URL:

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

## Transport-Level Security

Transport-level security ensures that the connection is secure. You can configure transport security between client applications and AquaLogic Service Bus proxy services and between AquaLogic Service Bus and business services.

Securing the connection includes authenticating the identity of the users by various means. If any part of authentication fails, the overall authentication will fail, and the message will be rejected. Whether or not authentication succeeds or fails, WebLogic Server generates an audit event (if audit providers are configured in the security realm). Each audit provider can be configured to filter specific event types. For example, a filter may specify that the audit provider write only failed authentication events, not successful ones.

The following types of transport security are supported:

- [HTTPS Transport-Level Security](#)
- [HTTP Transport-Level Security](#)

- [Security for JMS, Email, FTP, and File Transport](#)
- [Transport-Level Security in Message Flow and WS-Callout](#)

## HTTPS Transport-Level Security

AquaLogic Service Bus relies on WebLogic Server for server-side SSL support, including session management, client certificate validation and authentication, trust management, and server SSL key and certificate configuration. Currently, AquaLogic Service Bus supports only the default WebLogic Server (SSL) secure network channel. AquaLogic Service Bus supports both inbound and outbound HTTPS proxy endpoints, as described in the following sections:

- [Inbound HTTPS Transport-Level Security](#)
- [Outbound HTTPS Transport-Level Security](#)

### Inbound HTTPS Transport-Level Security

Inbound transport-level security applies to the connection between client applications and AquaLogic Service Bus proxy services.

You configure transport-level security for the proxy endpoints using the AquaLogic Service Bus Console while configuring proxy services. For information on how to do this, see [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

The AquaLogic Service Bus Console allows you to configure HTTPS proxy endpoints for the following security levels:

- None—one-way SSL
- Basic—one-way SSL and user-password client authentication
- Client Certificate—two-way SSL, both client and server authenticate during the SSL handshake

#### None—Inbound HTTPS

When you specify HTTPS *without* BASIC or CLIENT CERT authentication, you secure your messages with one-way SSL. In one-way SSL, the client initiates the connection and WebLogic Server sends its certificate to the client. In other words, the client authenticates WebLogic Server. For information about configuring SSL certificates in WebLogic Server, see [“WebLogic Server Prerequisites” on page 3-5](#).

## Basic—Inbound HTTPS

When you specify HTTPS with BASIC authentication, authentication takes place over an encrypted channel so passwords are secure. After the one-way SSL connection is established, the client authenticates to WebLogic Server with a username and password. Trust is established by validating the username and password using the authentication providers configured in the WebLogic Server security realm. The client must send its username and password on the HTTP request header.

**Warning:** When creating an HTTPS proxy-service endpoint that requires BASIC authentication, a transport-authorization policy is not automatically associated with the inbound endpoint URI. To enforce BASIC authentication, you must define a transport-authorization policy for the endpoint.

As mentioned in the warning, if a transport-authorization policy (security policy) for the inbound endpoint is not associated with the endpoint URL, authentication will not occur—the server will accept HTTPS requests without the username and password header and will not challenge the client to provide the username and password. Moreover, authentication will not take place if the client has preemptively included the BASIC username and password header in the request—the server will accept requests with invalid usernames or invalid passwords. After you configure a transport-authorization policy on the proxy service endpoint, the client will be forced to properly authenticate and the server will enforce this access-control policy. For information about security policies, see [“Access Control Security Policies” on page 3-34](#).

## Client Certificates—Inbound HTTPS

Client Certificate authentication provides the highest level of transport security. In addition to the client authenticating WebLogic Server, WebLogic Server will authenticate the client application during the SSL handshake. This is sometimes called two-way SSL. For more information about SSL, see “Secure Sockets Layer (SSL)” in [Security Fundamentals](#) in *Understanding WebLogic Security*.

Although a full description of CLIENT CERT authentication for inbound HTTPS is out of scope in this document, the essentials are as follows: The SSL engine validates the client certificate, which includes making sure that the client holds the private key (via the SSL protocol), establishing a chain of trust from the client certificate to one of the trusted CAs (Certificate Authority) in the trust keystore, checking the certificate expiration, and so on. Additional validation can be performed by configuring WebLogic Server SSL to invoke the Certificate Lookup and Validation Framework during the SSL handshake. After trust in the certificate has been established, the certificate is passed to the identity assertion providers to extract the client identity.

Identity assertion providers are another component of the WebLogic Security Framework. Identity asserters can be configured to extract a field in the certificate to use as the client identity, typically the



CN (common name) or E (email) of the `SubjectDistinguishedName` in the certificate. After extracting the field from the certificate, the extracted client identity is compared to the registered users in the authentication provider. If the client identity matches, the authentication provider collects all groups to which the user belongs. The end result is a signed Java Authentication and Authorization Service (JAAS) Subject with one principal that holds the client identity and one principal for each group to which the user belongs.

For more information about identity assertion providers, see [Security Fundamentals](#) in *Understanding WebLogic Security*.

## Outbound HTTPS Transport-Level Security

Outbound transport security applies to the connections between AquaLogic Service Bus proxy services and business services.

You configure transport-level security for the outbound endpoints while creating or editing a business service on the Transport Configuration page. For information on how to do this, see [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.

The AquaLogic Service Bus Console allows you to configure HTTPS outbound endpoints for the following security levels:

- None—one-way SSL
- Basic—one-way SSL and user-password client authentication
- Client Certificate—two-way SSL, both client and server authenticate during the SSL handshake

In None, Basic, and Client Certificate, the remote server certificate is validated during the SSL handshake. Hostname verification, if enabled, checks the `SubjectDistinguishedName` in the server certificate against the business service URL. For more information, see “Hostname Verification” under “Secure Sockets Layer (SSL)” in [Security Fundamentals](#) in *Understanding WebLogic Security*.

**Note:** Hostname verification should be enabled in a production environment.

### None—Outbound HTTPS

When you specify HTTPS *without* BASIC or CLIENT CERT authentication, you secure your messages with one-way SSL. In one-way SSL, the proxy service initiates the connection, but does not authenticate itself to the business service.

## Basic—Outbound HTTPS

When you specify HTTPS with BASIC authentication, authentication takes place over an encrypted channel so that passwords are secure. After the one-way SSL connection is established, the proxy service authenticates itself to the business service with a username and password. The proxy service uses the username and password associated with the service account that is defined for the business service. You specify the service account on the business service HTTPS Transport Configuration page and associate the username and password with the service account in the Credentials section of the Security Configuration module. For more information on credentials and service accounts, see [“Credentials” on page 3-31](#).

## Client Certificate—Outbound HTTPS

Client Certificate authentication provides the highest level of transport security. In this case, two-way SSL is established. During the SSL handshake, the proxy service authenticates itself to the business service with an SSL certificate. The SSL private-key and corresponding certificate that the proxy service uses to authenticate to the business service is supplied by the proxy service provider defined for that proxy service. This means that before specifying CLIENT CERT authentication, you must configure a proxy service provider and associate an SSL client key-pair with that proxy service provider. For more information, see [“Proxy Service Providers” on page 3-32](#).

## HTTP Transport-Level Security

The AquaLogic Service Bus Console allows you to configure AquaLogic Service Bus to require BASIC authentication for inbound and outbound endpoints. In inbound BASIC authentication, the client authenticates to WebLogic Server with a username and password. In outbound BASIC authentication, the proxy service authenticates itself to the business service using a username and password. Unlike HTTPS transport-level security, when using HTTP the business service does not authenticate itself to the proxy service.

**Warning:** Although the AquaLogic Service Bus Console allows you to specify BASIC authentication for HTTP connections, this is strongly discouraged because the password is sent in clear text. However, it is safe to send passwords over HTTPS, as HTTPS provides an encrypted channel.

If you do use BASIC Authentication for HTTP outbound endpoints, you must specify a Service Account. This service account maps the username and password that AquaLogic Service Bus uses to connect to the business service. For more information, see [“Service Accounts” on page 3-32](#).

Also if you use BASIC Authentication for HTTP inbound endpoints, you must associate a transport-authorization policy with the proxy service URI. For more information, see [“Basic—Inbound HTTPS” on page 3-10](#).

## Security for JMS, Email, FTP, and File Transport

This section provides information about protecting JMS, email, FTP, and file transport.

### JMS Transport-Level Security

You can configure AquaLogic Service Bus to provide transport security over JMS for inbound messages to a proxy service and outbound messages from a proxy service. The connection to JMS servers is secured using the T3S protocol (T3 over SSL). While this 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 the JMS server for sending or receiving JMS messages.
- The ability to specify the credentials (username and password) that AquaLogic Service Bus proxy services use to authenticate while establishing the connection to a JMS server and/or while looking up JMS destinations in the JNDI tree.

For a description of the T3 protocol, see [Using WebLogic RMI with T3 Protocol](#) in *Programming WebLogic RMI*.

### Inbound JMS Transport-Level Security

As previously mentioned, for inbound transport, you can designate that the connection to the JMS server uses the T3S protocol. This is done while creating or editing a proxy service by selecting the Use SSL check box on the Transport Configuration page. For information on how to do this, see [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

If the JMS administrator has restricted access to a JMS connection pool, you need to configure your proxy service to authenticate when connecting to the JMS server, as follows:

1. To supply the username and password to the JMS server, create a (JMS) service account in the Project Explorer module in the AquaLogic Service Bus Console.
2. While creating or editing a proxy service, designate the JMS service account on the JMS Transport Configuration page.

3. Specify the username and password for the JMS service account using the Credentials section of the Security Configuration module. If the JMS server is in a remote WebLogic Server domain, trust must be established between the domains.

For information on how to configure service accounts, see [Service Accounts](#) in the *AquaLogic Service Bus Console Online Help*.

### Outbound JMS Transport-Level Security

For outbound messages, you can designate that the connection to the JMS server uses the T3S protocol. This is done while creating or editing a business service by selecting the Use SSL check box on the Transport Configuration page. For specific information on how to do this, see [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.

AquaLogic Service Bus supports access control for both the JMS connection pool and the JNDI entry for the JMS destination (queue or topic). You can configure the access-control policies for these two resources independently. When access control to the JMS connection pool has been configured, AquaLogic Service Bus must authenticate while establishing the connection to the JMS server. When access control to the JNDI entry has been configured, AquaLogic Service Bus must authenticate during the JNDI lookup. You must specify which service account to use for authentication when accessing either resource.

AquaLogic Service Bus can communicate with the local JMS server or a foreign JMS server.

For information on configuring access control to a WebLogic JMS Server and the JNDI entry, see [Configuring JMS System Resources](#) in *Configuring and Managing WebLogic JMS* and [Securing WebLogic Resources](#).

To configure AquaLogic Service Bus to authenticate while establishing the connection to the JMS server, take the following steps:

1. To supply the username and password to the JMS server, create a service account for the JMS server in the Project Explorer module of the AquaLogic Service Bus Console.
2. Specify the username and password for the JMS service accounts using the Credentials section of the Security Configuration module.
3. While creating or editing the business service, designate this service account as the JMS service account on the JMS Transport Configuration page, as shown in [Figure 3-3](#).

To configure AquaLogic Service Bus to authenticate while looking up the JNDI entry for the JMS queue or topic, take the following steps:

1. To supply the username and password for the JNDI entry, create a service account for the JNDI entry in the Project Explorer module of the AquaLogic Service Bus Console.
2. Specify the username and password for the JNDI service accounts using the Credentials section of the Security Configuration module.
3. While creating or editing the business service, designate this service account as the JNDI service account on the JMS Transport Configuration page, as shown in the following figure.

**Figure 3-3 JMS Transport Configuration Page**

Edit a Business Service - JMS Transport Configuration (Path - default)	
Destination Type	<input checked="" type="radio"/> Queue <input type="radio"/> Topic
Use SSL	<input type="checkbox"/>
Message Type	<input checked="" type="radio"/> Bytes <input type="radio"/> Text
Expiration	<input type="text" value="0"/>
Is Response Required	<input type="checkbox"/>
Response URI	<input type="text"/>
Response Timeout	<input type="text" value="0"/>
Unit Of Order	<input type="text"/>
JNDI service account	<input type="text" value="Your JNDI Service Account"/> <input type="button" value="Browse"/>
Request encoding	<input type="text" value="utf-8"/>
Response encoding	<input type="text" value="utf-8"/>
JMS service account	<input type="text" value="Your JMS Service Account"/> <input type="button" value="Browse"/>
Dispatch policy	<input type="text" value="default"/> ▼

For specific information on how to configure service accounts, see [Service Accounts](#) in the *AquaLogic Service Bus Console Online Help*.

## Email and FTP Transport-Level Security

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 *AquaLogic Service Bus Console Online Help*.

## 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 in Message Flow and WS-Callout

Transport-level security is available from within the proxy service pipeline from the following:

- **Message Flow**—in a proxy service, the processing of messages is driven by metadata specified in the *message flow* definition. This definition can use the client identity to apply a transformation to a message or determine which pipeline branch the message flow should follow. If you need to specify authenticated transport-level user information in the message flow, you can do this using the `security` element in the `inbound` context variables. For information on how to use the `security` element, see “Inbound and Outbound Variables” in [Message Context](#), and for information on configuring message flow, see “Message Flow” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
- **Web Service Callout**—in a message flow, you use a Web Service callout to look up information in a registered business service. You can apply transport-level security to the connection using HTTP or HTTPS.

## Message-Level Security

This section includes information on the following topics:

- [About Message-Level Security](#)
- [Inbound Web Services Security](#)
- [Outbound Web Services Security](#)

## About Message-Level Security

Message-level security specifies whether the SOAP messages between a client application and a Web service should be digitally signed or encrypted or both. It is also used to specify Username Token

authentication and X.509 certificate authentication. Signing insures message integrity, that is, it ensures that the message wasn't tampered with. However, signing does not prevent the message from being read. Encryption prevents the message from being read by unauthorized persons.

Message-level security provides a level of flexibility not present in transport-level security. It can ensure that SOAP messages are secure even when the transmission involves one or more intermediaries and because the security measures are built into the message, the message can be protected point-to-point or end-to-end. Additionally, you can specify that only parts of the message be signed or encrypted. This ability is useful for routing messages in AquaLogic Service Bus. Message-level security provides confidentiality of the protected information where needed and along the entire message path.

Message-level security is configured using security policy statements, as specified by the Web Service Policy (WS-Policy). WS-Policy statements are bound to a business or proxy service's WSDLs. These policies may be referenced XML documents from within the WSDL or included as part of the WSDL. A WSDL may import other WSDLs containing policy attachments, either inlined or external. For more information about WS-Policy, see [“Web Service Policy” on page 3-21](#).

When Web services security (WS-Security) is applied to a message, a new SOAP header is added to the message envelope. The new header includes the XML-DSIG digital signatures, security tokens, and other constructs. These security tokens can be used for the following:

- Sender authentication
- Key wrapping—which is carrying a randomly generated symmetric encryption key encrypted with the recipient's public key
- Carrying the signature verification certificate
- Including the sender's encryption public certificate—which provides the means for the recipient to encrypt the response.

When the recipient consumes the secured envelope, the cryptographic operations are performed in reverse order and the security header is removed. The recipient then verifies that the message conforms to its policy. For example, the recipient confirms that the required message parts were signed and/or encrypted and that the required tokens are present with the required claims.

## Inbound Web Services Security

This section includes information on the following topics:

- [About Inbound Web Services Security](#)
- [Client Request and Proxy Service Response](#)

- [Configuring Inbound Web Services Security](#)

## About Inbound Web Services Security

Inbound WS-Security applies to messages between client applications and AquaLogic Service Bus proxy services. It applies to both the request and the response between the client application and AquaLogic Service Bus.

As previously mentioned, you specify the details for inbound message-level security using WS-Policy statements, which are inlined within or referenced through WSDLs. AquaLogic Service Bus provides two types of inbound message security: active intermediary and pass-through. In the active intermediary scenario, the proxy service is configured with WS-Policy statements. In the pass-through scenario, the proxy service is configured with WS-Policy statements, but processing is disabled. In the active intermediary scenario, the WS-Policy statement is publicized to clients (from the WSDL) and the proxy service processes the header and enforces the policy. In the pass-through scenario, the WS-Policy statement is publicized to clients but not enforced by the AquaLogic Service Bus proxy; the client application enforces the policy.

## Client Request and Proxy Service Response

When a client makes a request to a proxy service and the proxy service responds to the request, two security scenarios are available:

- **Active-Intermediary**—in this scenario, the client applies WS-Security to the request and response messages. The proxy service processes the message header and enforces the security policy on the messages.
- **Pass-Through**—in this scenario, the client applies WS-Security to the request and response messages. The proxy service 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. Note that the business service must be configured with WS-Policy security statements. The secured response message is passed untouched to the client.

## Configuring Inbound Web Services Security

You configure inbound message-level security while creating or editing an AquaLogic Service Bus proxy service. Use the following steps as a guideline. The particular steps you need to perform depend on whether you are configuring an active-intermediary or pass-through scenario.

- Assign a proxy service provider to the proxy service.

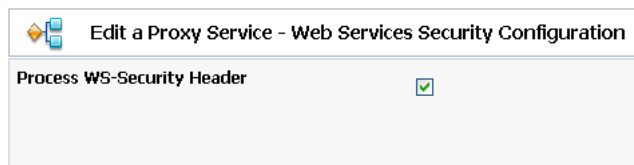


A proxy service provider is required when the inbound message to a proxy service is using WS-Security encryption; the proxy service provider supplies the key pair to the proxy service for decrypting inbound messages.

**Note:** To be able to assign a proxy service provider to a proxy service, you first need to define the proxy service provider.

- Map any key-pair credentials (for encryption) to the proxy service provider assigned to the proxy service on the Create New Credential page in the Security Configuration module. For information on how to do this, see “Adding a Credential” in [Security Configuration](#) in the *AquaLogic Service Bus Console Online Help*.
- Import a WSDL with WS-Policy security statements, either inlined or referenced. The references to external policies must be resolved. For information about unresolved WSDL references, see “Resolving Unresolved WSDL References” in [WSDLs](#) in the *AquaLogic Service Bus Console Online Help*.
- If you are configuring an active-intermediary scenario, you must select the Process WS Security Header check box on the Web Service Security Configuration page, as shown in [Figure 3-4](#). (This page is available only when WS-Policy statements are inlined or referenced from the WSDL.) For information on how to do this, see [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

**Figure 3-4 Process Web Services Security Header**



- If you are configuring a pass-through scenario, do not select the Process WS Security Header check box on the Web Service Security Configuration page, as shown in the previous figure.

## Outbound Web Services Security

This section contains information on the following topics:

- [About Outbound Web Services Security](#)
- [Proxy Service Request and Business Service Response](#)
- [Configuring Outbound Web Services Security](#)

## About 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. Outbound WS-Security applies only to SOAP-HTTP or SOAP-JMS business services. You specify the details for outbound message-level security using WS-Policy statements attached (inlined or referenced) from the WSDL.

**Note:** For SOAP-JMS, WS-Security is supported only for one-way messages. It is not supported for request and response messaging.

As with inbound Web service security, outbound WS-Security has both an active intermediary scenario and the pass-through scenario.

## Proxy Service Request and Business Service Response

When a proxy service makes a request to a business service and the business service responds to the request, two security scenarios are available:

- **Active-Intermediary**—in this scenario, the proxy service applies WS-Security to the request and response messages. The business service processes the message header and enforces WS-Policy security.
- **Pass-Through**—in this scenario, the client applies WS-Security to the request and response messages. The proxy service passes the secured message untouched to a business service. After the business service receives the message, it processes the security header and acts on the request. The secured response message is passed untouched to the client.

## Configuring Outbound Web Services Security

You configure outbound message-level security while creating or editing a business service. Use the following steps as a guideline. The particular steps you need to perform depend on whether you are configuring an active-intermediary or pass-through scenario.

- Import a WSDL with WS-Policy statements attached (inlined or referenced) from the business service. If the security policies are referenced, they must be resolved. If the business service is a WebLogic Server 9.0 Web service or if an AquaLogic Service Bus proxy service is running on another AquaLogic Service Bus installation, you can get its WSDL from a running server by pointing a Web browser to `<service-url>?WSDL`. For information about resolving WSDL references, see “Resolving Unresolved WSDL References” in [WSDLs](#) in the *AquaLogic Service Bus Console Online Help*.

- Define the business service in AquaLogic Service Bus using the WSDL as a template. If the WS-Policy of the business service requires UsernameToken authentication, you must specify a service account. Proxy services that route to this business service will get the username and password for WSS Username Token authentication from the service account.
- In the proxy service, configure the route node and operations on the business service.
- Set up any credentials (signing key pair and X.509 key pair) needed for the proxy service and target service in the proxy server provider.
- If necessary, set the `doOutboundWSS` flag in the message context variables' `security` element:
  - If the `doOutboundWSS` flag is explicitly set to `true`, the proxy service will act on the security policy (active intermediary scenario).
  - If the `doOutboundWSS` flag is explicitly set to `false`, the proxy service will not act on the message even if the proxy service is configured with a WS-Policy statement (pass-through scenario).
  - If the `doOutboundWSS` flag is *not* explicitly set to `true` or `false` and if the target service has a WS-Policy statement, the proxy service will apply the policy to the message.

For information on configuring the `security` element and the `doOutboundWSS` flag, see “Inbound and Outbound Variables” in [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.

**Note:** When the WS-Policy of a business service requires messages to be encrypted, you must make sure that the policy of the business service with the encryption certificate embedded is inlined in the WSDL. You must refer to this WSDL when defining the business service in the AquaLogic Service Bus Console. Otherwise, AquaLogic Service Bus will not be able to retrieve the public key to use for encrypting the messages to the business service.

## Web Service Policy

This section contains information on the following topics:

- [About Web Service Policy](#)
- [Web Services Policy Attachment](#)
- [Effective Policy](#)
- [Out-of-the-Box WS-Policy Statements](#)

## About Web Service Policy

Web Services Policy (WS-Policy) is an extensible XML-based framework that extends the configuration of a Web service with domain specific assertions and specifies the requirements, expectations, and capabilities of Web services. In AquaLogic Service Bus, WS-Policy is used for configuration of message-level security in proxy and business services using security policy statements. Because the specification has not been fully standardized, AquaLogic Service Bus supports a WebLogic Server-proprietary format. For more information, see [Configuring Security](#) in *Programming Web Services for WebLogic Server*.

WS-Policy statements ensure message integrity, confidentiality, and message origin authentication by specifying signing, encryption, application of security algorithms, and authentication mechanisms. A WS-Policy statement may include both security and reliable messaging assertions. At this time, AquaLogic Service Bus supports security assertions, but not reliable messaging assertions.

WS-Policy statements are XML documents, which may be inlined or referenced from WSDLs. A WSDL may import other WSDLs containing policy attachments, either inlined or referenced. You can associate one or more WS-Policy statements to different WSDL constructs as described later in this chapter.

In AquaLogic Service Bus, WS-Policies are required to have a `wsu:Id` attribute from the following namespace:

```
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 repository. Note that this attribute is optional in the WS-Policy schema.

[Listing 3-1](#) shows a WS-Policy with a security policy assertion that specifies encryption of the SOAP body.

### Listing 3-1 Sample WS-Policy

---

```
<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="encryptWithX509Token">

  <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>wsp:GetBody(.)</wssp:MessageParts>
    </wssp:Target>
    <wssp:KeyInfo>
      <wssp:SecurityToken TokenType=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"/>
    </wssp:KeyInfo>
  </wssp:Confidentiality>
</wsp:Policy>

```

---

For information on using WS-Policy in the AquaLogic Service Bus Console, see the following in *AquaLogic Service Bus Console Online Help*:

- [WSDLs](#)
- [WS-Policies](#)
- [Business Services](#)
- [Proxy Services](#)

## Web Services Policy Attachment

WS-Policy Attachment, defines the mechanisms for assigning policies to Web services. WebLogic Server 9.0, and subsequently, AquaLogic Service Bus, implements only WSDL policy attachment. Policies statements may be inlined in a WSDL document by adding a `<wsp:Policy>` element as a child of the `<wsdl:definition>` element. An XML fragment identifier is used to reference inlined policies.

Referenced WS-Policy statements are associated with a Web service by adding one or more of the following policy URI attributes or policy reference elements to WSDL elements (which type depends on what the WSDL schema allows):

- **PolicyURIs**—a global attribute whose value is a list of URIs. Each URI is a single policy reference. Use for WSDL elements that allow only attribute extensibility.
- **PolicyReference**—a global element with a URI attribute, which also includes a digest and digest algorithm. The URI is a reference to a policy. Use for WSDL elements that allow only element extensibility.

Using `PolicyURIs` or `PolicyReference`, you can reference one or more policies. The references may be local to the WSDL document (fragment URIs) or external.

WS-Policy Attachment also defines a `<wsp:UsingPolicy/>` element that must appear as a child to the `<wsdl:definitions>` element whenever a WSDL has policy attachments. To ensure that proxy and business services are capable of processing the policy attachments, this element can be marked as a mandatory extension (`wsdl:required="true"`) in the WSDL.

**Warning:** If the `UsingPolicy` tag is missing from the WSDL, AquaLogic Service Bus ignores any WS-Policy present in the WSDL.

[Listing 3-2](#) shows a WSDL with two inlined policies. One inlined policy, `policy1`, is referenced from the input part of operation `doFoo` on `portType Sample` using the `policyURIs` syntax with a fragment identifier. The other inlined policy, `policy2`, is referenced from operation `doFoo` in binding `SampleBinding` using the nested `PolicyReference` syntax.

### Listing 3-2 WSDL with Policy References to 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>

```

---

## Effective Policy

You can associate WS-Policies statements with the different policy subjects. A policy subject is an entity, such as service, endpoint, operation, or message, with which a policy can be associated. 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 `wsdl: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. In other words, a policy scope is the collection of policy subjects to which a policy applies. For example, the effective policy of the input message of a binding operation is the merge of the following:

- all policies attached to the input message of the binding operation
- all policies attached to the binding operation
- all policies attached to the binding
- all policies attached to the input message of the port-type operation
- all policies attached to the port-type operation

- all policies attached to the port-type
- all policies attached to the service

**Note:** When a proxy service is defined from a binding, any WS-Policy attached to the service or port is ignored.

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

**Figure 3-5 Effective Policy**

OPERATION	EFFECTIVE REQUEST/RESPONSE POLICY
doFoo	<pre>&lt;wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"&gt; &lt;ExactlyOne xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"&gt;   &lt;All&gt;     &lt;wssp:Integrity xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part" xmlns:wssp="http://www.bea.com/wls90/security/policy" xmlns:wsu="http://www.w3.org/2000/09/xmldsig#rsa-sha1"&gt;       &lt;wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/&gt;       &lt;wssp:CanonicalizationAlgorithm URI="http://www.w3.org/2001/10/xml-exc-c14n#"/&gt;       &lt;wssp:Target&gt;         &lt;wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/&gt;         &lt;wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part"&gt;wls:System-Headers()&lt;/wssp:MessageParts&gt;       &lt;/wssp:Target&gt;       &lt;wssp:Target&gt;         &lt;wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/&gt;         &lt;wssp:MessageParts Dialect="http://www.bea.com/wls90/security/policy/wsee#part"&gt;wls:Security-Header(wsu:Timestamp)&lt;/wssp:MessageParts&gt;       &lt;/wssp:Target&gt;       &lt;wssp:Target&gt;         &lt;wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/&gt;         &lt;wssp:MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part"&gt;wsp:Body()&lt;/wssp:MessageParts&gt;       &lt;/wssp:Target&gt;     &lt;/wssp:Integrity&gt;     &lt;wssp:MessageAge xmlns:wssp="http://www.bea.com/wls90/security/policy"/&gt;   &lt;/All&gt; &lt;/ExactlyOne&gt; &lt;/wsp:Policy&gt;</pre>
	<pre>&lt;wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"&gt; &lt;ExactlyOne xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"&gt;   &lt;All&gt;     &lt;wssp:Integrity xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part" xmlns:wssp="http://www.bea.com/wls90/security/policy" xmlns:wsu="http://www.w3.org/2000/09/xmldsig#rsa-sha1"&gt;       &lt;wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/&gt;</pre>

## Out-of-the-Box WS-Policy Statements

WebLogic Server includes three out-of-the-box WS-Policy statements. Most of the time a combination of one or more of these policies will address your requirements. However, you can write your own WS-Policies, import them to AquaLogic Service Bus, and refer to them from the WSDL.

To use the out-of-the-box policies, reference them from any WSDL using URIs, such as `policyURIs="policy:Auth.xml"`. The following policies are included with WebLogic Server.

- `Auth.xml`—specifies that the client application invoking the Web service must authenticate itself. The policy does not specify what type of security tokens are accepted. This depends on the server configuration, specifically the `WebServiceSecurityMBean` instances. For more information, see [“WebLogic Server Prerequisites” on page 3-5](#).
- `Encrypt.xml`—the SOAP body must be encrypted 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.



- `Sign.xml`—this policy ensures message integrity. It requires the client 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:

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

The namespace prefixes correspond to the namespaces in the following table:

Prefix	Namespace
wsm	<code>http://schemas.xmlsoap.org/ws/2005/02/rm</code>
wsa	<code>http://schemas.xmlsoap.org/ws/2004/08/addressing</code>
wsu	<code>http://schemas.xmlsoap.org/ws/2002/07/utility</code>
wsax	<code>http://schemas.xmlsoap.org/ws/2004/01/addressingx</code>

The out-of-the-box policies are located in the `BEA_HOME/weblogic90/server/lib/weblogic.jar`, where `BEA_HOME` is the directory in which your BEA products are installed. For more information on these policies, see “Use of WS-Policy Files for Message-Level Security Configuration” in [Configuring Security](#) in *Programming Web Services for WebLogic Server*.

[Listing 3-3](#) shows a WSDL with references to two of the out-of-the-box policies.

### Listing 3-3 WSDL with Policy References to Out-Of-The-Box 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="#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="policy:Sign" />
                <wsp:PolicyReference URI="policy:Auth" />
            </wsp:Policy>
            ...
        </operation>
    </binding>
    ...
</definitions>
```

## Access Control Security

This section contains information on the following topics:

- [About Access Control Security](#)

- [Users](#)
- [Groups](#)
- [Security Roles](#)
- [Credentials](#)

## About Access Control Security

Access control determines who has access to the resources in AquaLogic Service Bus. 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. When a user attempts to access a 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.

AquaLogic Service Bus Console and MBean access are secured with built-in role-based access-control security policies. These security policies are not WS-Policy statements. Recall that in this release, MBean access control is not configurable. Message flow can also be controlled by configuring security policies on proxy services. For more information, see [Access Control Security Policies](#).

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. You configure security policies in the WebLogic Server Administration Console.

**Warning:** 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 *AquaLogic Service Bus Console Online Help*.

## Users

Users are entities that can be authenticated in a security realm. A user can be a person, such as an application end-user, or a software entity, such as a client application, or other instances of WebLogic Server. As a result of authentication, a user is assigned an identity, or principal. Each user is given a

unique identity within the security realm. Users may be placed into groups that are associated with security roles, or be directly associated with security roles.

## Groups

To make users easier to manage, users can be grouped. Groups are logically ordered sets of users, usually having something in common. Managing groups is more efficient than managing large numbers of users individually. For example, an administrator can specify permissions for multiple users at one time by placing the users in a group, assigning the group to a security role, and then associating the security role with a WebLogic resource via a security policy. All user and group names must be unique within a security realm.

AquaLogic Service Bus includes the following predefined groups:

- **Administrators**—have complete access to all AquaLogic Service Bus objects and functions.
- **Deployers**—have read access to all objects. Can create, delete, edit, import or export resources, services, proxy service providers, or projects.
- **IntegrationAdministrators**—have complete access to all AquaLogic Service Bus objects and functions, except for defining security policies and editing users, group, and roles.
- **IntegrationDeployers**—have read access to all objects. Can create, delete, edit, import or export resources, services, proxy service providers, or projects.
- **IntegrationMonitors**—have read access to all objects. Can export any resource, service, proxy service provider, or project.
- **IntegrationOperators**—have read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.
- **Monitors**—have read access to all objects. Can export any resource, service, proxy service provider, or project.
- **Operators**—have read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.

## Security Roles

As previously mentioned, AquaLogic Service Bus supports role-based authorization using the WebLogic Security Framework. Authorization involves granting an entity permissions and rights to perform certain actions on a resource. In role-based authorization, security policies define the roles that are authorized to access the resource.

The difference between groups and security roles is that a group is a static identity assigned by an administrator, while membership in a security role is dynamically calculated based on data such as username, group membership, or the time of day. Security roles can be granted to individual users or to groups.

AquaLogic Service Bus includes built-in roles that are associated with certain administrative and monitoring privileges as follows:

- **Integration Administrator**—have complete access to all AquaLogic Service Bus objects and functions, except for defining security policies and editing users, group, and roles.
- **Integration Operator**—have read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.
- **Integration Monitor**—have read access to all objects. Can export any resource, service, proxy service provider, or project.
- **Integration Deployer**—have read access to all objects. Can create, delete, edit, import or export resources, services, proxy service providers, or projects.

For information on how to configure users and groups, see [Security Configuration](#) in the *AquaLogic Service Bus Console Online Help*.

For more information about security roles, see [Users, Groups, and Security Roles](#), in *Securing WebLogic Resources*.

## Credentials

You can configure AquaLogic Service Bus with the credentials it needs to securely interact with clients and business services. AquaLogic Service Bus credentials are built on top of the WebLogic Security Framework. To set up credentials, you use the Credentials section of the Security Configuration module in the AquaLogic Service Bus Console. For information on using the console to configure credentials, see [Security Configuration](#) in the *AquaLogic Service Bus Console Online Help*. AquaLogic Service Bus supports the following types of credentials:

- Username and password
- Key pairs

This section includes information on the following topics:

- [Service Accounts](#)
- [Proxy Service Providers](#)

- [Configuring Credentials](#)
- [Access Control Security Policies](#)

### Service Accounts

A service account is an alias resource for a username and password. AquaLogic Service Bus uses service accounts to provide authentication when connecting to a business service or server. For example, when configuring FTP transport-level security for a business service, you may need to provide a username and password to authenticate to the FTP server.

Service accounts are used when configuring transport protocols for business services. Before configuring your business services, you have to define a service account. You can use the same service account for multiple purposes. After you define a service account, you can specify the associated username and password to the service account using the Credentials section of the Security Configuration module in the AquaLogic Service Bus Console. For more information, see [“Configuring Credentials” on page 3-33](#).

For information on how to create a service account, see [Service Accounts](#) in the *AquaLogic Service Bus Console Online Help*.

### Proxy Service Providers

To configure security for a proxy service, you must designate a proxy service provider. Proxy service providers provide credentials for inbound and outbound messages from a proxy service. You must configure a PKI credential mapper in your security realm when using a proxy service provider. For more information, see [“Configuring a PKI Credential Mapping Provider” in \*Configuring WebLogic Security Providers\* in \*Securing WebLogic Server\*](#).

**Warning:** After adding or deleting a security provider, such as a PKI credential mapper, you must reboot the server for the security changes to take effect.

The proxy service obtains the needed PKI credentials from the proxy service provider. For example, to open an HTTPS connection with client-certificate authentication, the designated proxy service provider supplies the key-pair that the proxy service uses for SSL authentication. Multiple proxy services can use the same proxy service provider. You can assign different PKI credentials, such as private-key and certificate pairs, to a proxy service provider for different purposes. Proxy service providers supply the following types of security:

- **SSL Client Authentication**—used for outbound transport-level security. For a description of CLIENT CERT authentication, see [“Outbound HTTPS Transport-Level Security” on page 3-11](#).

- **Digital Signature**—provides outbound message integrity. This key pair is used with WS-Security when a proxy service is required by WS-Policy statements to sign one or more parts of a SOAP envelope.
- **Encryption**—provides inbound message confidentiality. This key-pair is used with WS-Security when a proxy service is required by WS-Policy statements to decrypt one or more parts of a SOAP envelope.
- **X.509 authentication**—this key pair is used for outbound WS-Security when a business service requires X.509 authentication.

Before configuring security for a proxy service, you first need to create a proxy service provider. This allows you to designate the proxy service provider while configuring the proxy service. After you activate the proxy service in the AquaLogic Service Bus Console, you can associate PKI credentials to the proxy service provider using the Credentials section of the Security Configuration module. For more information, see [“Configuring Credentials” on page 3-33](#).

For more information on how to create a proxy service provider, see [Proxy Service Providers](#) in the *AquaLogic Service Bus Console Online Help*. For information about credential mapping providers, see Security Providers in [Understanding WebLogic Security](#) and [“WebLogic Server Prerequisites” on page 3-5](#).

## Configuring Credentials

Credentials are persisted in security providers and not in the repository governed by the AquaLogic Service Bus sessions. This means that credentials are independent of the session in which the associated service account or proxy service provider are created. Be sure to follow these guidelines:

- If you want to configure AquaLogic Service Bus to access credentials, use the following order:
  - a. Create the service account or proxy service provider.
  - b. Activate the session. This makes the service account or proxy service provider visible in the Credentials section.
  - c. After the session is activated, define the credential for the service account or proxy service provider.
- If you want to delete, rename, or move a service account or proxy service provider:
  - a. Before activating a session, delete the credential from the credentials section of the AquaLogic Service Bus Console.

- b. After the credential is deleted, activate a session to delete, rename, or move the service account or proxy service provider.

For information on how to perform the steps described in this section, see [Service Accounts](#), [Proxy Service Providers](#), and [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

### Access Control Security Policies

An (access control) security 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. Security 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 security policies, see [Security Fundamentals](#) in *Understanding WebLogic Security* and [Components of a Security Policy: Policy Conditions, Expressions, and Statements](#) in *Securing WebLogic Resources*.

You configure security policies in the WebLogic Server Administration Console. For more information, see [Security Policies](#) in *Securing WebLogic Resources* and [Manage Security Policies](#) in the *WebLogic Server Administration Console Online Help*.

Security policies are persisted in security providers and not in the repository governed by the AquaLogic Service Bus sessions. Subsequently, security policies are independent of the session in which the associated proxy service is created. If you make changes to your proxy services, be sure to follow these guidelines:

#### **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 or its operations.
3. Move or rename the proxy.



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 change the proxy service URL:**

1. Create a session if you have not already done so.
2. Delete any transport security policy assigned to the proxy URL.
3. Edit the proxy URL.
4. Active the session.
5. Re-assign the transport security policy 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.

## Securing AquaLogic Service Bus for 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 *AquaLogic Service Bus Console Online Help*.
- 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
```

## Securing Inbound and Outbound Messages

- 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.

# Change Management and Resource Organization

BEA AquaLogic Service Bus provides a number of capabilities to manage changes and organize large numbers of resources in the repository. The resources in the AquaLogic Service Bus repository include WSDLs, Schemas, XQueries, XSLTs, MFLs, WS-Policies, Business Services, and Proxy Services.

This section includes the following topics:

- [Repository Users](#)
- [Projects and Folders](#)
- [Sessions](#)
- [Concurrent Modifications](#)
- [Tracking Configuration Changes](#)
- [Tracking Dependencies](#)
- [Semantic Integrity](#)
- [Undoing Modifications to Resources and Session Activations](#)
- [Importing and Exporting Configurations](#)
- [Scripting Support](#)

## Repository Users

AquaLogic Service Bus is focused on supporting a set of trusted IT department specialists who manage the resources and services in the repository on behalf of the organizations they represent. All such users are defined as integration administrators or integration deployers and have full permissions to modify all the resources in the repository.

Integration monitor users have full read access to the repository but cannot modify any resources. Typically, they are users who search or browse for resources or services.

Integration operator users have full read access to the repository and can only change the operational characteristics of the services.

AquaLogic Service Bus is not focused on supporting large numbers of end users managing their resources with fine grained security, as would be required in a shared enterprise repository and services directory or in trading partner management systems.

For more information about AquaLogic Service Bus users and roles, see [Security Configuration](#) in the *AquaLogic Service Bus Console Online Help*.

## Projects and Folders

The resources in the AquaLogic Service Bus repository can be organized into separate projects. The projects can contain folders, which in turn, can contain other folders. A typical use case is one in which corporate-wide standard resources (for example, schemas or abstract WSDLs) are located in one project, while department-level (local) resources are placed into separate projects per department. Additionally, each type of resource can be placed into separate folders in a project. Typically, a small team or an individual manages the resources in a project.

Resources can be moved between projects or folders and can be renamed. Resources that are located in one project can reference and use resources that are defined in other projects.

Dependencies are preserved when resources are renamed and moved. All references to a renamed or moved resource are automatically adjusted. For more information about referenced resources, see [“Tracking Dependencies” on page 4-4](#).

For more information, see [Project Explorer](#) in the *AquaLogic Service Bus Console Online Help*.

## Sessions

Before you make modifications to resources in AquaLogic Service Bus, you must create a session. All modifications are made using the AquaLogic Service Bus Console in a given session. A session can be considered a sandbox environment in which changes are kept private to the user

making those changes. In other words, they are not visible to other concurrent users making modifications. Note that modifications made in a session are not deployed in the server until the session is activated, so it is not possible to execute the server with the changes in the session before activation. You can have only one session active at any time and should only log into the AquaLogic Service Bus Console through one browser.

To compare a resource modified in a given session against the resource that is already deployed to run time, you can temporarily exit the session, view the deployed resource, then reenter the session and view the changed resource. All resources are visible in the session. The view of all resources in a session is called the session view—it is a merged view of the unmodified deployed resources and the resources modified in the current session. Therefore, the session view at any point in time shows the configuration state if the session is activated at that point in time. The view of resources outside any session is the view of the deployed resources.

All individual session modifications, individual session activations, and undo operations for a session are performed transactionally to prevent data loss in the event of a failure. Sessions are persistent and long running—the restart of a server does not result in the loss of active sessions. This means that you can modify the configuration in a single session over a period of days (during which time the server can be stopped and restarted) if necessary. Each user has their own session, and can work in it independently without the need to lock other users out of the system. You cannot activate a session if another user is already in the process of activating their session. If another user is activating a session when you try to activate your session, the Activate button will be disabled and you will have to wait until the other session is activated before you can activate your session. In certain circumstances, the Activate button may not be disabled if you did not refresh the page or if you are directly using MBeans. In this case, you will time out after a short while.

Administrators have permissions to access other user's sessions and view ongoing changes, make updates in those sessions, or discard them.

For more information, see [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

## Concurrent Modifications

Sessions use an optimistic scheme for conflicts. When you activate a session, the changes you made to resources in that session become visible immediately in other sessions. If you deploy a changed resource that is open in another user's active session, the other user's session receives a message in the Change Center indicating that the deployed resource has changed in the run time since the user started modifications. The user of the active session can then:

- Discard the changes to the resource in the current session. That is, refresh the resource in the session with the newly deployed resource.
- Activate the current session, which results in the resource in the run time being overwritten with the current session's changes. This is the default behavior.

For more information, see [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

## Tracking Configuration Changes

The system keeps a log of all users who activated a session along with any resource modified by the session and when it was modified. This provides the enterprise with auditing and tracking facilities in addition to a history of changes made to a particular resource or project. The log is visible in the Change Center in the AquaLogic Service Bus Console.

## Tracking Dependencies

A crucial part of managing a large number of resources is establishing and exploring dependencies between resources. For example, it is useful to identify the WSDL that a service implements, or the XQueries used by a message flow configuration. AquaLogic Service Bus provides this capability by automatically tracking the references between resources and creating a graph of the dependencies. In both session views and deployed views, the AquaLogic Service Bus Console displays for a given resource:

- The resources that it references
- The resources that it is referenced by

Also, for each project and folder, the AquaLogic Service Bus Console displays other resources outside the project or folder that reference resources in the selected project or folder. The AquaLogic Service Bus Console also displays the resources that a given project or folder references. This aids dependency tracking—you can easily navigate the dependency graph in the AquaLogic Service Bus Console by clicking on the names of the referenced resources.

You can use this functionality to identify the dependencies between departmental projects or between departmental projects and corporate-wide shared projects in the repository.

Dependencies are preserved when resources are renamed and moved. All references to a renamed or moved resource are automatically adjusted.

## Semantic Integrity

AquaLogic Service Bus protects the integrity of all resources in the session view. You can view a list of all current validation errors for all resources in the session view by clicking the View Conflicts link in the Change Center. Changes to a referenced resource can cause validation errors in any resources that reference it.

AquaLogic Service Bus allows you to create resources with most semantic errors. However, all such errors must be fixed before a session can be committed.

There are certain classes of validation errors that are never allowed. If you attempt to update a resource that has one of these disallowed validation errors, your update will fail. For example, where your configuration requires an XQuery, you cannot enter arbitrary text in place of the XQuery. If you try to do this, your update fails. The XQuery and XPath editors in the AquaLogic Service Bus Console provide a facility to validate your expressions. Click Validate to validate your XQuery and XPath expressions at design time. This reduces the possibility of run-time errors as a result of invalid configurations.

## Undoing Modifications to Resources and Session Activations

You can undo tasks that you have performed in your AquaLogic Service Bus configuration during your current session, and you can undo session activations outside of a session.

### Undoing Modifications to Resources

If you are working in a session, you can view a list of the modifications you have made in the session by accessing View Changes in the Change Center. You can undo specific tasks in the Change Center. An undo operation can result in objects becoming semantically invalid. For example, if a WSDL operation name change is undone, the proxy service routing to that operation on the service that uses that WSDL is semantically invalid. These validation errors are displayed immediately in the Change Center when you click the View Conflicts link.

Although you can undo tasks in any order (provided that individual undo operations result in valid data), the resulting configuration may be different depending on the order of undo. The undo operation sets the value of the resource to the value it had before the change to that resource. If the task being undone was one that created an object, there is no previous state to which an object can be returned—in other words, no object existed before this task was performed. Effectively, the undo operation deletes the new object from the session. In this case, errors occur for the objects that reference the one being deleted. You can view such errors on the View Conflicts page in the Change Center.

## Undoing Session Activations

When you are not working in a session, you can view a list of session activations by accessing View Changes in the Change Center. You can undo a session activation in the Change Center. When you undo a session, the session activation is undone and all the operations performed in the session are lost. The system does not allow you to undo a session activation if an error in the run time configuration would result from the undo operation. For example, if you attempt to undo a deployment that removes an object that is being referenced by another object, that undo operation is disallowed. For more information, see “Undoing a Task” in [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

**Note:** Once an undo operation succeeds, you cannot redo it.

For more information, see [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

## Importing and Exporting Configurations

An important part of large scale development is the ability to develop, test, stage, and deploy resources to a production system. AquaLogic Service Bus provides import and export features, and support for the global change of environment-specific attributes for resources. This functionality minimizes the expertise, time, and resources needed to achieve various deployment scenarios.

In a simple scenario, you can propagate your configuration from one instance of AquaLogic Service Bus to another instance by exporting all, or a subset of, the resources currently deployed in an AquaLogic Service Bus domain. The configuration you want to export is saved in a JAR file, which you can then import into a session on another AquaLogic Service Bus domain.

There are no restrictions on what can be exported. You can choose to export one or more projects, or select resources from one or more projects. The AquaLogic Service Bus Console also allows you to export a resource and all the other resources on which that resource depends, using the dependency tracking feature. You must be working outside of a session to export configurations. Only configurations that have been activated (that is, deployed to run time) can be exported.

You must be working in a session to import a configuration JAR file. You can choose to import only a subset of the exported data, and change the values of certain configuration data. You first open the JAR file, and then work on the configuration data to customize it.

You can perform many updates and import multiple JAR files in a single session. In this way, you can tailor the imported resources for the new domain before activating them.



Support for environment values is an important feature related to the import and export facilities. AquaLogic Service Bus allows you to find and replace certain values in resources in a global manner. AquaLogic Service Bus defines two types of pre-defined environment values:

- Service URIs
- File/directory paths

Using the import functionality in concert with the find and replace feature, you can import a JAR file, find all the URLs containing a specific string (for example, localhost:7001), and change it to another value (say, productionhost:7002). The find and replace feature is provided to facilitate changing many similar values in a convenient way. It is not meant to replace a more careful tuning of configuration that may be required by complex deployment scenarios.

For more information, see [System Administration](#) in the *AquaLogic Service Bus Console Online Help*.

## Scripting Support

You can perform all AquaLogic Service Bus configuration and deployment using the AquaLogic Service Bus Console. You can also use the WebLogic Server Scripting Tool (WLST) to automate deployment tasks.

For information, see [Using the AquaLogic Service Bus Deployment API](#) in the *BEA AquaLogic Service Bus Deployment Guide*.

For information about WLST, see [WLST Command and Variable Reference](#) in the *WebLogic Scripting Tool*.



# Monitoring

BEA AquaLogic Service Bus provides the capability to monitor and collect run-time information for both systems operations and business auditing purposes. AquaLogic Service Bus aggregates run-time statistics that you can view on a customizable Dashboard. The Dashboard allows you to monitor the health of the system and alerts you to problems in your messaging services. With this information, you can quickly and easily isolate and diagnose problems as they occur.

This chapter includes the following topics:

- [Monitoring Scenarios](#)
- [About Monitoring](#)
- [Service Summary](#)
- [Server Summary](#)
- [Alert Summary](#)
- [Alert Rules](#)

## Monitoring Scenarios

The following describes some of the ways in which you can use AquaLogic Service Bus to check system operations and monitor messages.

### **Operational Health**

The Dashboard page in the AquaLogic Service Bus Console provides the ability to immediately view the state of all servers and monitored services. The Dashboard displays

two pie charts, a table, and several links. The Service Summary pie chart shows the percentage of alerts according to their severity for all services that have alerts defined and monitoring enabled for the last 30 minutes. The Server Summary pie chart shows the current status of every server in the AquaLogic Service Bus domain. Additionally, from the Server Summary panel, you can drill-down and view the domain logs, which are grouped according to severity.

In addition to the pie charts, these Summaries include a list of the most active services and critical servers. The most active services are those that have the most number of alerts in descending order up to ten services. The most critical server list displays the ten most critical servers. This display is based on the health state of the running servers, as defined by the WebLogic Diagnostic Service. For more information about the WebLogic Diagnostic Service, see [Understanding the WebLogic Diagnostic Framework](#).

From each of the summaries, you can drill-down into more detail by clicking a specific area on a pie chart or by clicking one of the links on the page.

The default Alert Summary table shows the severity of the alert, when the alert occurred, the name of the corresponding service, and what alert rule was violated. Alerts are displayed by severity. You can customize, search, and scroll through this table.

### Alert Monitoring

When you log into the AquaLogic Service Bus Console, you see a list of alerts on the Dashboard. Each row of the table displays the information that you have configured, such as the severity, timestamp, and associated service. You notice that numerous alerts have been generated since your last viewing. To find the problem, you filter the alerts and discover that the Service Level Agreement (SLA) violation is due to errors produced by the Post-Trade Processing proxy service. SLAs are agreements that define the precise level of service expected by AquaLogic Service Bus business and proxy services.

Alternatively, attention to the problem involves an alert rule's ability to send messages in the event of a SLA violation. In this case, you are notified by email of the alert rule violation. After receiving the emails, you look into the problem and discover that the errors are produced by the Post-Trade Processing proxy service.

To narrow the problem down, you can use the reporting module. This scenario is continued in [“Message Tracking” on page 6-2](#).

### Statistics Monitoring

Suppose that you want to see how many messages in a particular service have processed successfully and how many have failed. To access this information, from the Dashboard, you access the Service Monitoring Summary page and filter the display for the relevant service. Besides displaying the number of messages that have successfully processed or

failed, you can also see which project the service belongs to, the average execution time of message processing, and the number of alerts associated with the service for the aggregation interval specified for that service.

Clicking the name of the service brings you to that service's Service Monitoring Details page. This page provides additional information, such as the success-failure ratio, the number of messages that have failed because of security or validation errors, and the number of messages associated with proxy service components (pipelines and route nodes).

### Verifying Service Level Agreements

You are notified by email of a large number of execution-time SLA violations from the Trade Execution proxy service. To track down this problem, you log into the AquaLogic Service Bus Console. From the Dashboard, you drill into the service associated with the alerts and see that a pipeline operation that invokes an Avitek Web Service is unacceptably slow. After successfully renegotiating service-level characteristics with Avitek, you configure alert metrics to track Avitek's compliance with the agreement. Your company uses these results as the basis of ongoing discussions with Avitek regarding their performance.

## About Monitoring

This section contains information on the following topics:

- [Aggregation Interval](#)
- [Monitoring Architecture](#)
- [Monitoring Services](#)
- [Refresh Rate of Monitored Information](#)
- [Dashboard](#)

## Aggregation Interval

In AquaLogic Service Bus, the monitoring subsystem collects statistical information, such as message-count and execution time, over an aggregation interval. The aggregation interval is the time period over which data points for a statistic are collected and then displayed in the AquaLogic Service Bus Console.

To illustrate how the aggregation interval works, suppose that you have configured a Purchasing Order proxy service that has monitoring enabled with an aggregation interval of 10 minutes.

When a user sends the first message through the proxy service, monitoring is started. During the first ten minutes, the Service Summary page displays the partially computed data. At this time the system does not have 10 minutes of data. After the first 10 minutes of data aggregation, the system always displays the last 10 minutes of data. For example, at the 14th minute, the Dashboard displays minutes 4 through 14. If no messages are processed after the 15th minute, on the 25th minute, the Service displays zero messages. For more information about how aggregation interval affects the display of monitored information, see [“Alert Rules” on page 5-26](#).

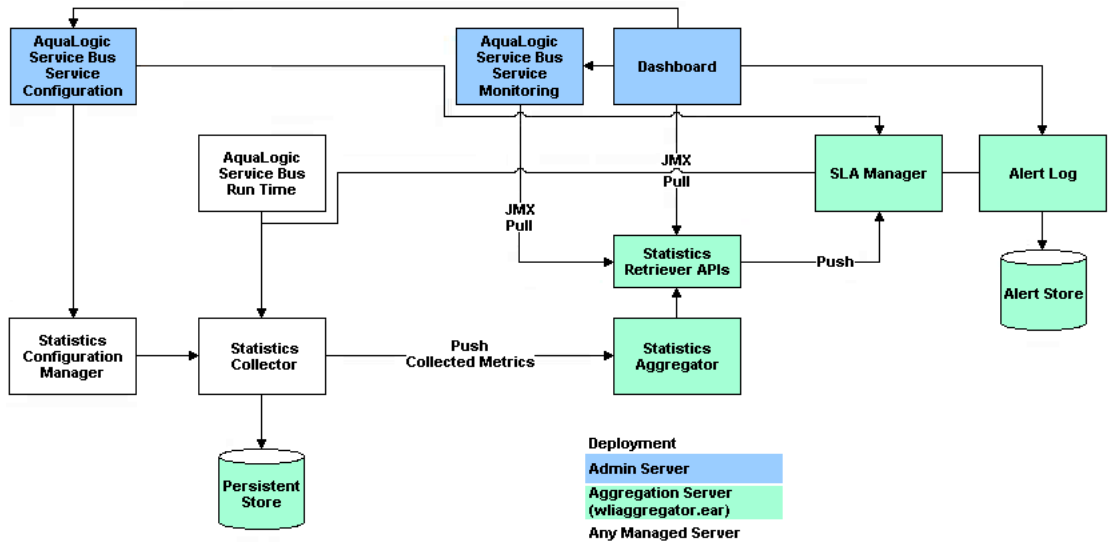
You must explicitly enable monitoring for any business or proxy service that you create; monitoring is disabled by default. After you have enabled monitoring and set the aggregation interval for your individual services, you can enable or disable monitoring for all those services from the Monitoring Configuration page in the System Administration module. For more information, see [“Monitoring Services” on page 5-6](#).

Alerts are automated responses to Service Level Agreements (SLAs) violations or occurrences, which are displayed on the Dashboard. You define alert rules to specify unacceptable service performance according to your business and performance requirements. Each alert rule allows you to specify the aggregation interval for that rule when configuring the alert rule. This aggregation interval is not affected by the aggregation interval set for the service. Alert rules also allow you to send an email notification or post a message to a JMS queue or topic about the violation.

## Monitoring Architecture

The following diagram shows the architecture of AquaLogic Service Bus monitoring.

Figure 5-1 Monitoring Architecture



The Statistics Configuration Manager stores and manages the statistics configuration for each operational resource. An operational resource is defined as the unit for which statistical information can be collected by the monitoring subsystem. An operational resource includes a proxy service, service operations, and pipelines. The Statistics Configuration Manager is notified about changes in the service definition, such as adding, updating, or deleting a pipeline.

Each managed server in a cluster hosts a Statistics Collector. The Statistics Collector collects statistics on operational resources as directed by the Statistics Configuration Manager. The collector also keeps samples history within the aggregation interval for the collected statistics. At every system-defined checkpoint interval, the collector stores the snapshot of current statistics into a persistent store for recovery purposes and sends the information to the Aggregator.

One of the managed servers in a cluster, called the *Aggregating Server* or *Aggregator*, is designated as the aggregator for cluster-wide statistics. At system-defined checkpoint intervals, each managed server in the cluster sends a checkpoint snapshot of its contributions to the Aggregator. The Aggregator then combines this information to offer cluster-wide statistics to its clients through Retriever APIs. The clients of Aggregator are the Dashboard, SLA Manager, and Service Monitoring modules.

To contribute a data point to the system, an operational resource in the system, such as a proxy service pipeline run time, calls a method on the Statistics Collector, and identifies itself, the statistic, and the data point.

The Dashboard shows the overall health related information of AquaLogic Service Bus. It provides an overview of the state of the system organized by server, services, and alerts.

After monitoring is enabled, the Service Monitoring Summary page in the AquaLogic Service Bus Console provides a view of the statistics collected for each service. It also provides information about the alerts generated due to SLA violations.

As previously mentioned, an SLA is an agreement that defines the precise level of service expected from business and proxy services in AquaLogic Service Bus. The SLA Manager, with the help of the AquaLogic Service Configuration module, allows users to configure SLA rule conditions and actions. The SLA Manager monitors SLA violations with the help of data provided by the Aggregator and sends notifications as configured in the alert rule actions. The SLA Manager is always deployed with the Aggregator and resides on only one managed server in cluster. The SLA Manager gives alerts to the Alert Log to store in the Alert Store.

## Monitoring Services

When you create a business or proxy service, monitoring is disabled by default for that service. Enable monitoring as follows:

- To enable monitoring for an individual service, select the Enable Monitoring check box on the Manage Monitoring page. Then set the aggregation interval for the service by selecting the interval times from the hour and minute drop-lists. For information on how to do this, see “Viewing the Dashboard Statistics” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.
- To enable monitoring for all services, select the Enable Monitoring check box on the Monitoring Configuration page. For information on how to do this, see “Enabling Monitoring” in [System Administration](#) in the *AquaLogic Service Bus Console Online Help*.

**Note:** The Monitoring Enabled option permits you to enable or disable monitoring of all services that have individually been enabled for monitoring. If monitoring for a particular service has *not* been enabled, you must first enable it and set the aggregation interval on the Manage Monitoring page for that service.

When creating alert rules, you must enable monitoring before you create the rule. For more information, see “Alert Rules” on page 5-26 and “Create an Alert Rule” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

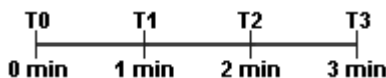


## Refresh Rate of Monitored Information

At run time, the default refresh rate for the Dashboard page is one minute. However, it may take up to three minutes for the information to be displayed on the Dashboard. This delay happens because of the time gaps between when the messages are processed by the proxy service, when the metrics are collected, and the refresh rate of the Dashboard. The system works as follows:

1. Every minute the data collector sends the current snapshot to the aggregator.
2. Every 60 seconds, the aggregator merges all the documents it has received from the managed servers within the last minute.
3. The AquaLogic Service Bus Console refreshes every minute; that is, it runs a query on the aggregated document and then displays the results.

**Figure 5-2 Aggregation Time Line**



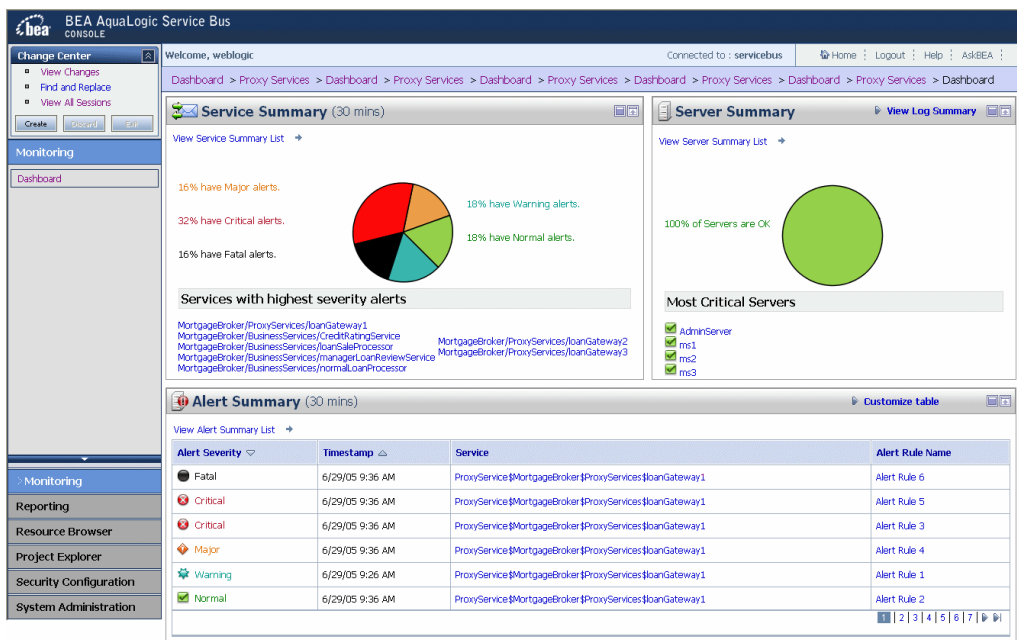
For example, a proxy service starts sending data in T1, as shown in [Figure 5-2](#). At T2—that is, the second minute—the collector sends the data to the aggregator. However, if an aggregation cycle has just occurred, the aggregator does not merge this data until the next aggregation cycle, which occurs after one minute, or a maximum of two minutes from the previous aggregation cycle. When the data is merged, it is now available for the AquaLogic Service Bus Console. Since the console refreshes every minute, if the refresh cycle has just passed, then the data is not displayed on the console until the third minute. Therefore, three minutes is the maximum delay.

You change the Dashboard polling interval in the System Administration module in the AquaLogic Service Bus Console. For information on how to do this, see “Setting the Dashboard Polling Interval Refresh Rate” in [System Administration](#) in the *AquaLogic Service Bus Console Online Help*.

## Dashboard

When you log onto the AquaLogic Service Bus Console, the Dashboard is automatically displayed. The Dashboard shows the monitoring information for the last 30 minutes. It provides an overview of the state of the system organized by server, services, and alerts, as shown in the following figure.

Figure 5-3 AquaLogic Service Bus Dashboard



As shown in the previous figure the Dashboard displays the following information:

- Services Summary—if alerts have been configured, summarizes the alert status for both proxy and business services. Alerts notify you of service performance based on rules you create.
- Servers Summary—displays the status of the servers.
- Alerts Summary—if alerts have been configured, displays which alert rules have been triggered.

From the Dashboard, you can drill-down into the system and easily find specific information, such as the average execution time of a service, the date and time an alert occurred, or length of time a server has been running.

You configure the Dashboard and monitoring in the AquaLogic Service Bus Console, which is described in the [Monitoring](#) and [System Administration](#) sections of the *AquaLogic Service Bus Console Online Help*.

## Service Summary

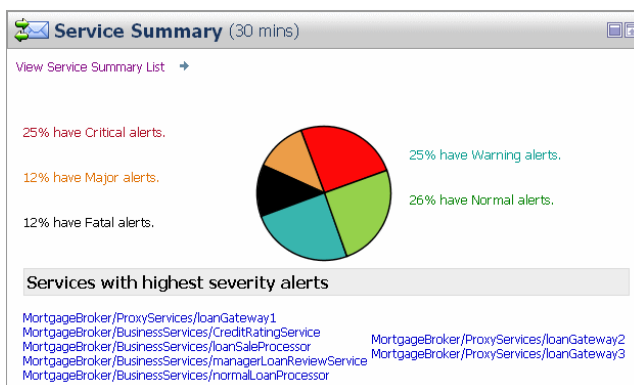
This section contains information on the following topics:

- [About the Service Summary](#)
- [Service Monitoring Summary](#)
- [Service Monitoring Details](#)

## About the Service Summary

The Service Summary panel provides an overview of the state of the services. The Service Summary pie chart shows the percentage of alerts according to their severity for all services that have alerts defined and monitoring enabled for the last 30 minutes. The severity level of alerts is user configurable and has no absolute meaning. The services having the highest severity alerts are listed beneath the pie chart, as shown in the following figure. Up to ten services can be listed in descending order. The order of severity is Fatal, Critical, Major, Minor, Warning, and Normal.

**Figure 5-4 Services Summary Pane**



From the Service Summary panel, you can access more information about alerts by clicking the following:

- A specific area on a pie chart—displays the Service Summary page.
- The name of a service under Services With Highest Severity Alerts—displays the Service Monitoring Details page for that service.

- View Service Summary List—displays the Service Monitoring Summary page. To help you locate specific services, you can filter the services by different criteria.

Each of these pages is fully described in the sections that follow.


**Warning:** When a service is renamed or relocated, its statistical data is lost.


For information on how to access detailed alert information, see “Viewing the Dashboard Statistics” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

## Service Monitoring Summary

The Service Monitoring Summary page provides a table of services, as shown in the following figure. The data displayed in the table is a moving statistic of the data collected by each service. The statistics displayed in each row is specified by the aggregation interval shown in the Aggregation Interval column. For example, if the aggregation interval of a particular service is 30 minutes, that service’s row displays the data collected in the last 30 minutes.

**Figure 5-5 Service Monitoring Summary Page**

 **Service Monitoring Summary**

 **Search** Name:  Path:

☐ Has Alerts  
☐ Has Errors  
☐ Invoked by proxy:

Name ▾	Path ▲	Aggregation Interval ▲	Avg. Execution Time (msecs) ▲	Message Count ▲	Error Count ▲	Alert Count ▲
<a href="#">CreditRatingService</a>	MortgageBroker/BusinessServices/CreditRatingService	0 Hour(s) and 5 Minutes	100	1490	0	16
<a href="#">loanGateway1</a>	MortgageBroker/ProxyServices/loanGateway1	0 Hour(s) and 5 Minutes	45	1716	6	25
<a href="#">loanGateway2</a>	MortgageBroker/ProxyServices/loanGateway2	0 Hour(s) and 5 Minutes	112	1754	2	3
<a href="#">loanGateway3</a>	MortgageBroker/ProxyServices/loanGateway3	0 Hour(s) and 5 Minutes	171	2253	1	0
<a href="#">loanSaleProcessor</a>	MortgageBroker/BusinessServices/loanSaleProcessor	0 Hour(s) and 5 Minutes	250	1757	3	25
<a href="#">managerLoanReviewService</a>	MortgageBroker/BusinessServices/managerLoanReviewService	0 Hour(s) and 5 Minutes	175	1115	2	11
<a href="#">normalLoanProcessor</a>	MortgageBroker/BusinessServices/normalLoanProcessor	0 Hour(s) and 5 Minutes	52	1562	0	1

As shown in the top section of the preceding figure, you can filter the display of information using the following criteria:

- Name and Path—the name of the proxy or business service, plus the project folder in which it resides.
- Has Alerts—by services that have alert messages.
- Has Errors—by services that have failed messages.

- Invoked by proxy—the name of the proxy service.

The Service Monitor Summary table displays the following information:

- Name—the name of the proxy or business service. The name is a link to the Service Monitoring Details page. See [“Service Monitoring Details” on page 5-11](#).
- Path—the project folder in which the service resides.
- Aggregation Interval—the time period over which data points for specific statistics are collected and then displayed for the service.
- Average Execution Time—the average time it takes the service to process a message.
- Message Count—the total number of messages processed by the service.
- Error Count—the number of messages that have failed.
- Alert Counts—the number of alerts raised by alert rule occurrences and violations.

## Service Monitoring Details

The Service Monitoring Details page provides you with detailed information about a specific service, as shown in the following figure.

**Figure 5-6 Service Monitoring Details Page**

Service Monitoring Details			
Alert Status : Normal at 4:49:45 PM June 29, 2005			
Aggregation Interval : 0 Hour(s) and 5 Minutes			
Alerts for last Aggregation interval : 2 <a href="#">Alert History</a>			
Location Path : MortgageBroker/BusinessServices			
Display Metrics For : <span>Cluster</span>			
Operations		Performance	
Operations	Message Count	Avg. Execution Time (msecs)	
processLoanApp	2153	153	
			Overall Avg. Execution Time (msecs): 256
			Total Number of Messages: 2253
			Messages With Errors: 13
			Success Rate: 99.4%
			Security: 2
			Validation: 3

The displayed details have the following definitions:

- Service Monitoring Details
  - Alert Status—the current alert status.

- Aggregation Interval—the time period over which data points for specific statistics are collected and then displayed for the service.
- Alerts for Last Aggregation Interval—the total number of alerts associated with this service within the last aggregation interval.
- Alert History—a link to the Customized System Alerts History page. See [“System Alerts History” on page 5-22](#).
- Location Path—the project and folder where the service resides.
- Display Metrics For—displays the metrics for a server. For a single node, only Cluster is displayed.
- Operations
  - Operations—the tasks performed by a pipeline or route node in the message flow associated with the service.
  - Message Count—the number of messages associated with each operation.
  - Average Execution Time—the average time the operation takes to execute messages.
- Performance
  - Overall Average Execution Time—the overall average time that the service takes to execute messages.
  - Total Number of Messages—the total number of messages, including failed messages.
  - Messages With Errors—the number of messages that failed. In two-way messaging, if the response message fails, but the request message was processed, only the failed response message is counted.
  - Success Rate—the ratio of successfully processed messages to failed messages.
  - Security—the number of messages that failed due to security reasons, such as authentication errors, security policy violations, or authorization errors.
  - Validation—the number of messages that failed when a validate action compared one or more parts of a message against an XSD schema or WSDL resource.
- Flow Components
  - Component Name—the name of pipeline or node in the message flow.
  - Message Count—the number of messages associated with each component.
  - Error Count—the number of failed messages associated with each component.

- Average Execution Time—the average time the component takes to execute a message.

## Server Summary

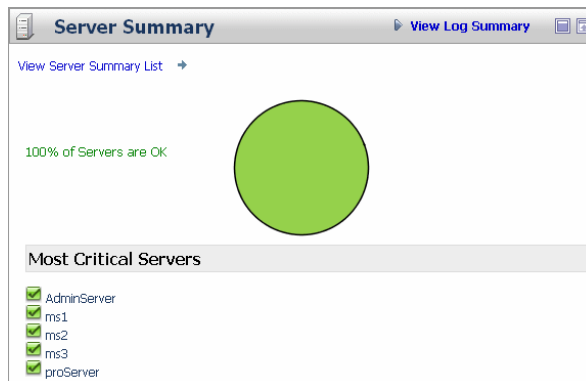
This section contains information on the following topics:

- [About the Server Summary](#)
- [Log Summary](#)
- [Server Summary](#)
- [Server Details](#)

## About the Server Summary

The Server Summary panel provides an overview of the state of the servers. The pie chart shows the status of each server in the domain. The status for each server is derived from the WebLogic Diagnostic Service (see [Understanding the WebLogic Diagnostic Framework](#)). The ten most critical servers are displayed, as shown in [Figure 5-7](#).

**Figure 5-7 Server Summary Pane**



The displayed statuses have the following meanings:

- Fatal—the server has failed and must be restarted.
- Critical—server failure pending; something must be done immediately to prevent failure. For more details, check the server logs and the corresponding `RuntimeMBean`.

- Warning—the server could have problems in the future. For more details, check the server logs and the corresponding `RuntimeMBean`.
- Ok—the server is functioning without any problems.
- Overloaded—the server has more work assigned to it than the configured threshold; it might refuse more work.

## Log Summary

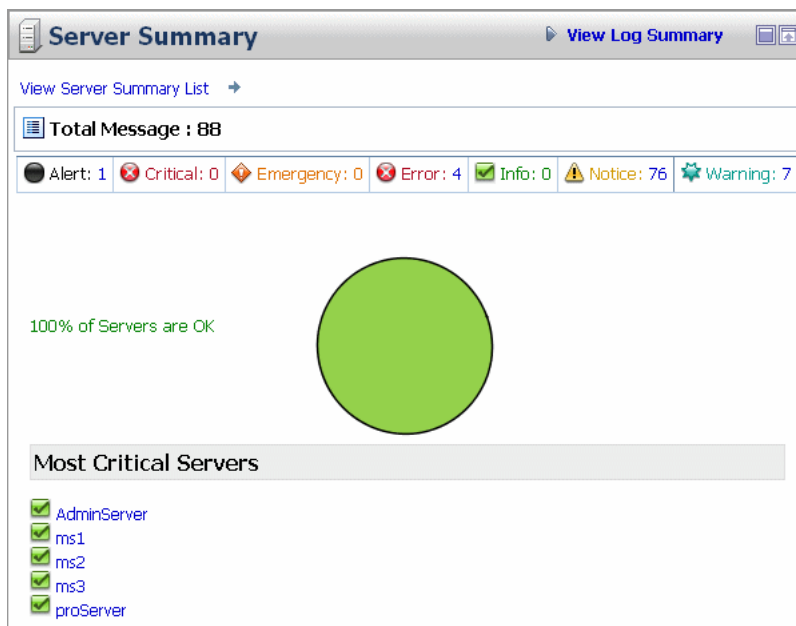
The AquaLogic Service Bus Console allows you to view the WebLogic Server domain log. The domain log file provides a central location from which to view the overall status of the domain. Each server instance forwards a subset of its messages to a domain-wide log file. By default, servers forward only messages of severity level NOTICE or higher. You can modify the set of messages that are forwarded. For more information, see [Understanding WebLogic Logging Services](#) in *Configuring Log Files and Filtering Log Messages*.

If you configure the logging action in a pipeline, the log is forwarded to the server log. Unless you configure WebLogic Server to forward these messages to the domain log, you cannot view this log from AquaLogic Service Bus Console. For information in how to do this, see [Create Log Filters](#) in the *WebLogic Server Administration Console Online Help*.

To see the number of messages currently raised by the system, click the View Log Summary link in the Server Summary panel. A table is displayed that contains the number of messages grouped by severity, as shown in the following figure.



Figure 5-8 Log Summary



The displayed message statuses have the following meanings:

- **Alert**—a particular service is in an unusable state while other parts of the system continue to function. Automatic recovery is not possible; the immediate attention of the administrator is needed to resolve the problem.
- **Critical**—a system or service error has occurred. The system can recover but there might be a momentary loss or permanent degradation of service.
- **Emergency**—the server is in an unusable state. This severity indicates a severe system failure.
- **Error**—a user error has occurred. The system or application can handle the error with no interruption. Limited degradation of service may occur.
- **Info**—reports normal operations; a low-level informational message.
- **Notice**—an informational message with a higher level of importance than Info messages.
- **Warning**—a suspicious operation or configuration has occurred. However, normal operations may not be affected.

This display is based on the health state of the running servers, as defined by the WebLogic Diagnostic Service. For more information about the WebLogic Diagnostic Service, see [Understanding the WebLogic Diagnostic Framework](#).

To view the domain log for a particular type of message, click the number corresponding with the type of message. The following figure shows an example of a domain log file displayed in the AquaLogic Service Bus Console.

Figure 5-9 Domain Log File Entries

Dashboard > Project Explorer > Proxy Services > Dashboard

This page shows you the latest contents of the domain log file.

Domain Log File Entries

View

Showing 1 - 8 of 8 Previous | Next

Date	Subsystem	Severity	Message ID	Message
<div><div></div><div>Jun 13, 2005 10:05:07 AM MDT</div></div>	BEA-AlertManager	Error	BEA-394000	Exception on executeAction, com.bea.wli.sb.transports.TransportException: com.bea.wli.sb.transports.TransportException: javax.mail.SendFailedException: Sending failed; nested exception is: javax.mail.MessagingException: Unknown SMTP host: host; nested exception is: java.net.UnknownHostException: host; com.bea.wli.sb.transports.TransportException: com.bea.wli.sb.transports.TransportException: javax.mail.SendFailedException: Sending failed; nested exception is: javax.mail.MessagingException: Unknown SMTP host: host; nested exception is: java.net.UnknownHostException: host; at com.bea.wli.sb.transports.email.EmailTransportProvider.sendMessageAsync(EmailTransportProvider.java:87) at jrockit.reflect.VirtualNativeMethodInvoker.invoke(Ljava.lang.Object;[Ljava.lang.Object;)[Ljava.lang.Object;(Unknown Source) at jrockit.reflect.InitialMethodInvoker.invoke(Ljava.lang.Object;[Ljava.lang.Object;)[Ljava.lang.Object;(Unknown Source) at java.lang.reflect.Method.invoke(Ljava.lang.Object;[Ljava.lang.Object;)[Ljava.lang.Object;(Unknown Source) at com.bea.wli.sb.transports.TransportManagerImpl\$1.invoke(TransportManagerImpl.java:1195) at \$Proxy9.sendMessageAsync(Lcom.bea.wli.sb.transports.TransportMessageContext;Lcom.bea.wli.sb.transports.TransportSendListener;)[V(Unknown Source) at com.bea.wli.sb.transports.TransportManagerImpl.sendMessageWithoutService(TransportManagerImpl.java:542) at com.bea.wli.sb.transports.TransportManagerImpl.sendMessageAsync(TransportManagerImpl.java:468) at com.bea.wli.monitoring.alert.action.emailAlert.EmailActionProvider.executeAction(EmailActionProvider.java:186) at com.bea.wli.monitoring.alert.AlertManager_invokeActions(AlertManager.java:593) at com.bea.wli.monitoring.alert.AlertManager.evaluateSingleRule(AlertManager.java:538) at

The following information is displayed:

- Date—the date and time the entry was logged in a format that is specific to the local time zone and format.
- Subsystem—the WebLogic Server subsystem that was the source of the message, such as the EJB container or Java Messaging Service.
- Severity—indicates the degree of impact or seriousness of the event.
- Message ID—the unique six-digit identification for the message.
- Message—a description of the event or condition.

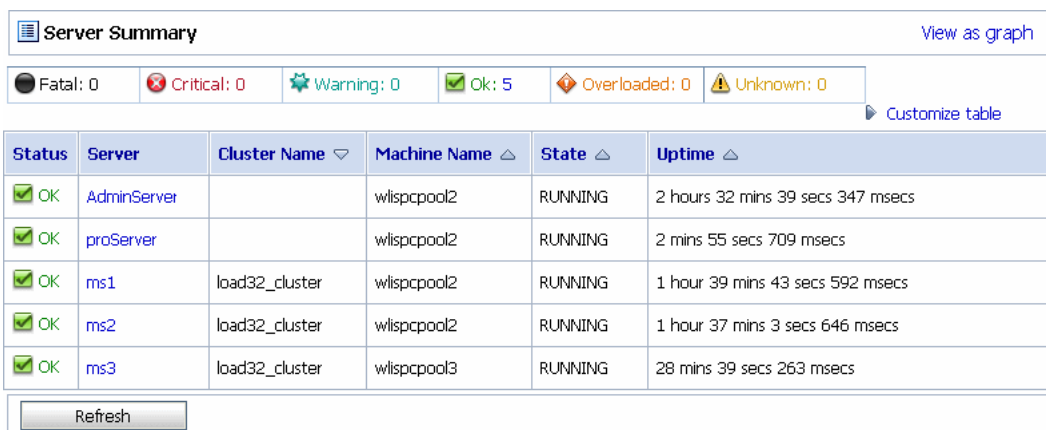
For more information, see “Message Attributes” in [Understanding WebLogic Logging Services in Configuring Log Files and Filtering Log Messages](#).

To display details of a single log file on the page, select the radio button for the appropriate log, then click the View button.

## Server Summary

The Server Summary page provides a customizable table of servers, as shown in the following figure.

**Figure 5-10 Server Summary Page**



As shown in the top section of the preceding figure, the Server Summary Page displays the number of messages currently raised by the system. For information about the meaning of each type of status message, see [“Log Summary” on page 5-14](#).

The server table displays the following information:

- **Status**—the status of the server:
  - **Fatal**—the server has failed and must be restarted.
  - **Critical**—server failure pending; something must be done immediately to prevent failure. For more details, check the server logs and the corresponding `RuntimeMBean`.
  - **Warning**—the server could have problems in the future. For more details, check the server logs and the corresponding `RuntimeMBean`.
  - **OK**—the server is functioning without any problems.
  - **Overloaded**—the server has more work assigned to it than its configured threshold; it might refuse more work.
- **Server**—the name of the server. The name is a link to the View Server Details page. See [“Server Details” on page 5-18](#).

- Cluster Name—if the server is associated with a cluster, the name of the cluster.
- Machine Name—the name of the computer associated with the server.
- State—the state of the server:
  - RUNNING
  - FAILED
  - SHUTDOWN
- Uptime—the length of time this server has been running.

To view this information in the table as a pie or bar chart, click View as a Graph.

To filter the display of servers, click Customize Table above the server table. The available filtering is shown in the following figure.

**Figure 5-11 Server Summary Table Filter**

The screenshot shows the 'Server Summary Table Filter' dialog box. It contains several sections: a list of filterable fields (Severity, Server, Cluster Name, Machine Name, State) each with a checkbox and a dropdown menu set to 'All'; a 'Columns Display' section with 'Available' and 'Chosen' columns and arrows for moving items between them; and a bottom section with 'Number of rows displayed per page' (set to 20) and 'Maximum Results Returned' (set to 10). 'Apply' and 'Reset' buttons are at the bottom.

Server Summary Table Filter	
<input type="checkbox"/> Severity	All
<input type="checkbox"/> Server	All
<input type="checkbox"/> Cluster Name	All
<input type="checkbox"/> Machine Name	All
<input type="checkbox"/> State	All
Columns Display	<div>Available</div> <div>Chosen</div>
	<div>Number of rows displayed per page: 20</div> <div>Maximum Results Returned: 10</div>
<div>Apply</div> <div>Reset</div>	

For information about how to use the Server Summary Table Filter, see “Customize Your View of the Server Summary” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

# Server Details

You can access the View Server Details page by clicking the name of a server under Most Critical Servers or by clicking the name of a server in the Servers Summary page.

The View Server Details page enables you to view more server monitoring details, as shown in the following figure.

Figure 5-12 Server Details Page—General Tab

Dashboard		
<a href="#">General</a>   <a href="#">Channels</a>   <a href="#">Performance</a>   <a href="#">Threads</a>   <a href="#">Timers</a>   <a href="#">Workload</a>   <a href="#">Security</a>   <a href="#">JMS</a>   <a href="#">JTA</a>		
This page provides general runtime information about this server.		
<b>State:</b>	RUNNING	<a href="#">core.server.servermonitoringgeneral.state.label.inlinhelp</a> <a href="#">More Info...</a>
<b>ActivationTime:</b>	Wed Jun 15 10:55:47 MDT 2005	<a href="#">core.server.servermonitoringgeneral.activationtime.label.inlinhelp</a> <a href="#">More Info...</a>
<a href="#">Advanced</a>		
<b>Weblogic Version:</b>	WebLogic Server 9.0 Mon Jun 13 20:43:42 PDT 2005 585258 - internal build by sjbuild on client qs.build.auntie	<a href="#">core.server.servermonitoringgeneral.weblogicversion.label.inlinhelp</a> <a href="#">More Info...</a>
<b>Java Vendor:</b>	BEA Systems, Inc.	<a href="#">core.server.servermonitoringgeneral.javavendor.label.inlinhelp</a> <a href="#">More Info...</a>
<b>Java Version:</b>	1.5.0_03	<a href="#">core.server.servermonitoringgeneral.javaversion.label.inlinhelp</a> <a href="#">More Info...</a>
<b>OSName:</b>	Windows XP	<a href="#">core.server.servermonitoringgeneral.osname.label.inlinhelp</a> <a href="#">More Info...</a>
<b>OSVersion:</b>	5.1	<a href="#">core.server.servermonitoringgeneral.osversion.label.inlinhelp</a> <a href="#">More Info...</a>
<b>JACC Enabled</b>	false	<a href="#">core.server.servermonitoringgeneral.jaccenabled.label.inlinhelp</a> <a href="#">More Info...</a>

The information displayed on this page is a subset of the Monitoring tab in the AquaLogic Service Bus Console Server Settings page. The details available are:

- **General**—provides general run-time information about the server. Click Advanced to display more information, such as WebLogic Server version or operating system name.
- **Channels**—displays monitoring information about each channel.
- **Performance**—displays performance information about the server.
- **Threads**—displays current run-time characteristics and statistics for the server's active executable queues.
- **Timers**—displays information about the timer in use by the server.
- **Workload**—displays statistics for work managers, constraints, and policies configured for the server.
- **Security**—allows you to monitor user-lockout management statistics for the server.
- **JMS**—allows you to monitor JMS information about the server.

- JTA—displays the summary of all transaction information for all resource types on the server.

For more information, see the [WebLogic Server Administration Console Online Help](#).

## Alert Summary

This section contains information on the following topics:

- [About the Alert Summary](#)
- [System Alerts History](#)
- [Rule Details](#)
- [View Alert Rule Details](#)

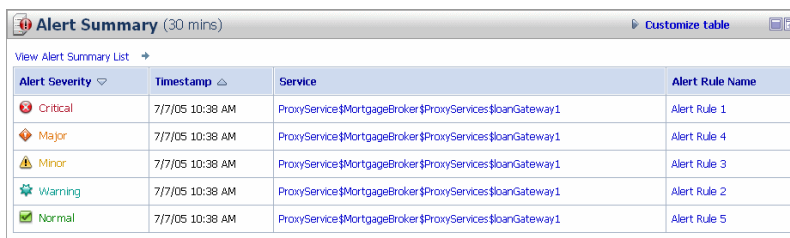
## About the Alert Summary

The Alert Summary panel contains a customizable table displaying information about violations or occurrences of events in the system. These violations and occurrences are based on SLAs. AquaLogic Service Bus provides various SLA monitors that you can configure to monitor proxy and business services. Some examples of SLA monitors are maximum execution time and authorization failure. You configure these monitors by creating alert rules. When a rule evaluates to true, it raises an alert. Additionally, you configure an alert rule to send an email or post a message on a JMS queue or topic.

**Note:** When you configure an alert rule to post a message to a JMS destination, you must create a JMS connection factory and a queue or topic, and target them to the appropriate JMS server in the WebLogic Server Administration Console. For information on how to do this, see “Configuring a JMS Connection Factory” and “JMS Resource Naming Rules for Domain Interoperability” in [Configuring JMS System Resources](#) in *Configuring and Managing WebLogic JMS*.

The AquaLogic Service Bus Console provides several ways to view and find alerts, such as by severity and by service. You can also view alerts graphically. For information on how to do this, see “Listing and Locating Alerts” and “Viewing a Chart of Alerts” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

The following figure shows the Alert Summary panel:

**Figure 5-13 Alert Summary Panel**


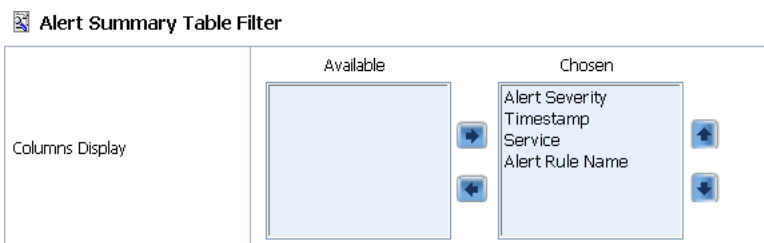
Alert Severity	Timestamp	Service	Alert Rule Name
Critical	7/7/05 10:38 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1	Alert Rule 1
Major	7/7/05 10:38 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1	Alert Rule 4
Minor	7/7/05 10:38 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1	Alert Rule 3
Warning	7/7/05 10:38 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1	Alert Rule 2
Normal	7/7/05 10:38 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1	Alert Rule 5

The Alert Summary panel shows alerts for the last 30 minutes. It contains the following types of information:

- **Alert Severity**—the user-defined severity of the alert. The Severity is a link to the Alert Rule Details page. See [“Rule Details” on page 5-23](#).
- **Timestamp**—the date and time that the alert occurred.
- **Alert Rule Name**—the name assigned to the alert. The name is a link to the View Alert Rule Details page. See [“View Alert Rule Details” on page 5-24](#).
- **Service/Project Name**—the name of the service and project associated with the alert. The name is a link to the Service Monitoring Details page. See [“Service Monitoring Details” on page 5-11](#).

To view a complete list of alerts, click [View Alert Summary List](#). See [“System Alerts History” on page 5-22](#).

To customize the information displayed in the Alert Summary Panel, click [Customize Table](#) above the summary table. The available filtering is shown in the following figure.

**Figure 5-14 Alert Summary Table Filter**


**Alert Summary Table Filter**

Columns Display

Available

Chosen

Alert Severity

Timestamp

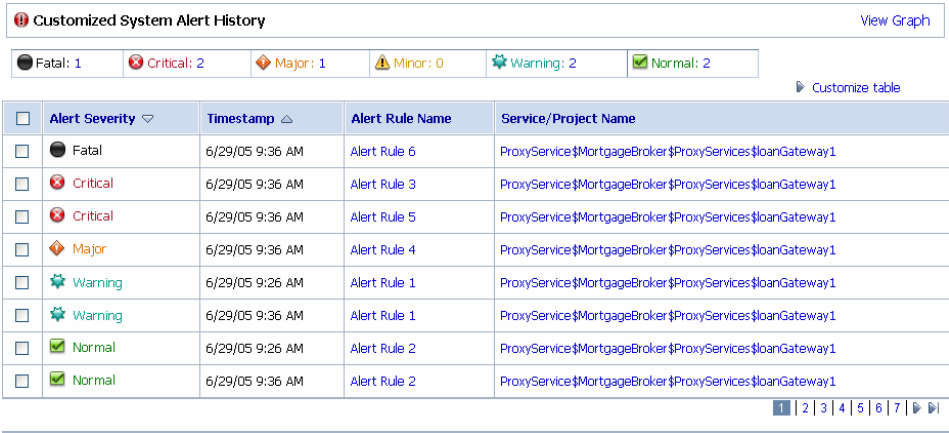
Service

Alert Rule Name

# System Alerts History

To access the Customized System Alerts History page, in the Alert Summary panel, click the View Alert Summary List. The Customized System Alerts History page enables you to view all the alerts by paging through the table (Figure 5-15) or by filtering the display of the alerts (Figure 5-16).

Figure 5-15 Customized System Alerts History



The table shown in the preceding figure is customizable and provides the following information:

- Alert Severity—the severity level of alerts is user configurable and has no absolute meaning.
- Timestamp—the date and time that the alert occurred.
- Alert Rule Name—the name assigned to the alert. The name is a link to the View Alert Rule Details page. See “View Alert Rule Details” on page 5-24.
- Service/Project Name—the name of the service and project associated with the alert. The name is a link to the Service Monitoring Details page. See “Service Monitoring Details” on page 5-11.

To view a pie or bar chart of the alerts, click View Graph in the table.

To search for a specific alert, you can filter the display of alerts by clicking Customize Table in the Customized System Alerts History table. The available filtering is shown in the following figure.



Figure 5-16 System Alerts Table Filter

System Alerts Table Filter

Time

June

15

2005

5

38

PM

June

15

2005

5

38

PM

0

days

0

hours

00

mins

☐Severity

All

☐Service

All

☐Alert Rule Name

All

Columns Display

Available

Chosen

Alert Severity

Timestamp

Alert Rule Name

Service/Project Name

Number of rows displayed per page

10

Maximum Results Returned

Show All

For information about how to use the Alerts Table Filter, see “Customize Your View of Alerts” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

## Rule Details

The Rule Details page displays complete information about the alert and allows you to add an annotation to the alert, as shown in the following figure.

Figure 5-17 Rules Details Page

Alert Rule 2 Details

OK

Cancel

Delete

Alert Name	Alert Rule 2
Description	
Timestamp	Wed Jun 29 16:47:35 MDT 2005
Severity	Normal
Alert	uuid:37cb61f8f3397d86-5ffcef17:104c8984e71-704a
Rule Id	2597702875768776266-5ffcef17:104c8984e71-7fd4
Alert Rule Name	<a href="#">Alert Rule 2</a>
Rule Path	
Service	<a href="#">ProxyService\$MortgageBroker\$ProxyServices\$loanGateway1</a>
Annotation	<div></div>

OK

Cancel

Delete

The following information is displayed:

- Alert Name—the name assigned to the alert.
- Description—a description of the alert.
- Timestamp—the date and time the alert occurred.
- Severity—the user-defined severity of the alert.
- Alert—the unique identification for this alert.
- Rule ID—the unique identification for the alert rule associated with this alert.
- Alert Rule Name—the name of the alert rule. The name is a link to the View Alert Rule Details page. See [“View Alert Rule Details” on page 5-24](#).
- Rule Path—the path for the alert rule.
- Service—the name of the service associated with the alert. The name is a link to the Service Monitoring Details page. See [“Service Monitoring Details” on page 5-11](#).
- Annotation—use this field to add notes to the alert.

You access this page from the Dashboard by clicking Alert Severity in the Alert Summary table. This page also allows you to delete the alert.

## View Alert Rule Details

The View Alert Rule Details page displays complete information about the alert, as shown in the following figure.

**Figure 5-18 View Alert Rule Details Page**

View Alert Rule Details - Alert Rule 9	
General Configuration <span>Edit</span>	
Rule Name:	Alert Rule 9
Rule Description:	
Start Time (HH:MM):	09:00
End time (HH:MM):	23:00
Rule Expiration Date (MM/DD/YY):	
Rule Enabled:	true
Alert Severity:	normal
Alert Frequency:	every-time
Stop Processing More Rules:	false
Include Log In Management Data Set:	true
Include Log In Reporting Data Set:	false
Conditions <span>Edit</span>	
Condition Expression :	Aggregation Interval :0 Hour(s) and 10 Minutes average Response Time > 300
Action Parameters <span>Edit</span>	
Send an alert via e-mail	mailto:admin@avitek.com

The following information is displayed:

- General Configuration
  - Rule Name—the name assigned to the alert rule.
  - Description—a description of the rule.
  - Start Time (HH:MM)—specifies the starting time during which the rule is active on each day prior to the expiration date.
  - End Time (HH:MM)—specifies the ending time during which the rule is active on each day prior to the expiration date.
  - Rule Expiration Date (MM/DD/YY)—the expiration date of the rule. The rule expires at 12.01am on the specified date. If you do not specify a date, the rule never expires.
  - Rule Enabled—indicates whether the rule is enabled or not.
  - Alert Severity—the user-defined severity of the alert.
  - Alert Frequency—indicates whether the actions (email or JMS destination) designated in the alert rule are executed every time the alert rule evaluates to true or are executed the first time the rule evaluates to true.

- Stop Processing More Rules—when multiple rules associated with a service exist, this flag indicates whether subsequent rules associated with the service must be evaluated if the current rule evaluates to true.
- Include Log in Management Data Set—indicates whether a log of the alert is included in the management data set. These alert logs are visible on the Dashboard in the Alert Summary table.
- Include Log in Reporting Data Set—indicates whether a log of the alert is included in the reporting data set. Viewing the reporting data set requires developing a Reporting Provider to fetch and display these logs. For more information, see [“Reporting Framework” on page 6-3](#).
- Conditions
  - Condition Expression—displays the condition that triggers the alert rule.
- Action Parameters
  - Send an alert via e-mail
  - Send an alert to a JMS Destination

For information about how to define alert rules, see “Create an Alert Rule” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

## Alert Rules

This section includes information on the following topics:

- [About Alert Rules](#)
- [Some Uses for Alerts](#)
- [Understanding Alert Rules](#)

## About Alert Rules

As mentioned earlier, alerts are automated responses to SLAs violations or occurrences, which are displayed on the Dashboard. You define alert rules to specify unacceptable service performance according to your business and performance requirements. Each alert rule allows you to specify the aggregation interval for that rule when configuring the alert rule. The alert aggregation interval is not affected by the aggregation interval set for the service.

Rules are executed once every aggregation interval. On the New Alert Rule page, if you set the Alert Frequency to Every Time, the rule's actions are executed every time the alert rule evaluates to true. If you set the Alert Frequency to Once Until Conditions Clear, the rule's actions are executed the first time the rule evaluates to true, and no more alerts are generated until the condition resets itself and evaluates to true again.

In the case where the Alert Frequency is set to Every Time, the number of times an alert rule is fired depends on the aggregation interval and the sample interval associated with that rule. For example, if the aggregation interval is set to 5 minutes, the sample interval is 1 minute. Rules are evaluated each time 5 samples of data are available. Therefore, the rule is evaluated for the first time approximately 5 minutes after it is created and every minute thereafter.

In the case where the Alert Frequency is set to Once Until Conditions Clear, after an alert is fired the first time in an aggregation interval, it is not fired again in the same aggregation interval.

Creating an alert rule involves three parts:

- **General Configuration**—defines the name, duration, severity, frequency, logging, and other general behavior.
- **Define Conditions**—defines one or more conditions that trigger the alert rule. Additionally, you defined the aggregation interval for the condition on this page.
- **Define Actions**—defines whether to use an email or JMS message for notification that the rule was triggered.

**Note:** Rules can only be created for services that are enabled for monitoring.

Detailed information about creating an alert rule is located in “Create an Alert Rule” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.

## Some Uses for Alerts

The following are some uses for alerts:

- Monitoring and email notification of WS-Security errors.
- Monitoring the number of messages passing through a particular pipeline.
- Email notification when the average execution time exceeds 5 seconds during stock exchange hours.

## Understanding Alert Rules

The information in this section is presented in question-answer format.

**Question 1:** I created a service with an alert rule that has the following condition expression:

```
Aggregation Interval: 0 Hours(s) and 1 Minutes  
Message Count = 0
```

It's been 10 minutes and I have not received any alerts.

**Answer:** Monitoring statistic collection for each statistical attribute, such as message count and error count, associated with a service begins when a change in the value of that statistic occurs. Data collection for the Message Count attributes begins when the first message is processed by the service and the Message Count attribute is incremented. Similarly, collection of data for the Error Count statistic starts only when the service encounters its first error and the Error Count attribute is incremented. If the service is idle, no monitoring information is collected for that service and subsequently no alert rules are triggered. After the first message is processed, monitoring data for that service is continually collected even if the service does not receive any further requests. Check to see if the service has received any requests.

**Question 2:** I defined a new alert rule with an aggregation interval that did not exist before and that rule does not seem to fire at all. All other rules created prior to this one are working correctly.

**Answer:** The cause is the same as in Question 1; the service needs to process at least one request after a rule with a new aggregation interval is created to trigger the alert rule. The other rules defined with different aggregation interval values are not affected by the alert rule.

**Question 3:** I restarted the server and none of my services have processed any requests. Why do I see alerts being generated?

**Answer:** Once the Monitoring subsystem has started collecting data for services, killing and restarting a server does not abort the collection process. The data collected is persisted and statistic collection picks up from where it left off.

**Question 4:** I have an alert rule with the following definition:

```
Aggregation Interval: 0 Hours(s) and 5 Minutes  
Success Rate < 80%
```

The Service Monitoring Summary page shows the following values:

Message Count: 4

Error Count: 1

Why am I being alerted in this case? Shouldn't the success rate be 80% in this case?

**Answer:** No, the message count value displayed is the total of all messages processed by the service, including the ones that generated an error. Subsequently, in this case, the success rate is 75%.

**Question 5:** I created a service with an aggregation interval of 10 minutes that sends a JMS message. I could see the message on the Service Monitoring Summary page, but some time later the message count for my service shows as zero.

**Answer:** The Service Monitoring Summary page displays a moving statistic. In this case, it shows the message count in the last 10 minutes. Because no messages were processed by the system in the last 10 minutes, the message count is displayed as zero.

**Question 6:** I changed the aggregation interval of a service from 10 minutes to 5 minutes. The Service Monitoring Summary page shows all statistics as zero. One of the alerts in this server was configured to a statistical element with a 2 minute aggregation interval, which did not fire the next minute.

**Answer:** Changing the aggregation interval for a service removes the statistical information for all the services and alerts associated with that service. The alert initializes again and fires after the next aggregation interval expiry.

**Question 7:** I have a business service with multiple endpoints with an alert rule defined as `Failover-count > 0`. When one of the endpoints goes down, the alert is triggered. However, when a service has only one endpoint, the `Failover-count` is not incremented for this service. Instead, an error is generated.

**Answer:** Set the Retry count to a number greater than zero. For information about setting the Retry count, see “Adding a Business Service” in [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.

**Question 8:** I see that an alert is generated on the Dashboard but the value for the Alerts for last Aggregation Interval field on the Service Monitoring Details page displays zero.

**Answer:** Alert rules are evaluated after the completion of the interval, which happens after a checkpoint completion. If a rule evaluates to true, the rule’s actions are triggered, a log is generated, and the interval-count statistic attribute (Alerts for Last Aggregation Interval) is incremented. The updated value of this counter is processed in the next checkpoint, 60 seconds later. The Monitoring Details page displays the updated count approximately one minute after the alert is generated.

**Question 9:** How does the active time for rules that span midnight work?

**Answer:** Consider the case where the active time for a rule is specified as 22:00 to 09:00.

## Monitoring

On a given date, say June 7, the rule will be active and inactive as follows:

June 6, 10:00 P.M. to June 7, 9:00 A.M. – Active

June 7, 9:01 A.M. to June 7, 9:59 P.M. – Inactive

June 7, 10:00 P.M. to June 8, 9:00 A.M. – Active



# Reporting

BEA AquaLogic Service Bus provides the capability to deliver message data and alerts to one or more reporting providers. Message data can be captured not only from the body of the message but any other variables associated with the message, such as header or inbound variables. Alert data contains information about Service Level Agreement (SLA) violations or occurrences that you can configure to monitor proxy services. You can use the message or alert data delivered to the reporting provider for functions such as tracking messages or regulatory auditing.

AquaLogic Service Bus includes a JMS Reporting Provider for message reporting. The Reporting module in the AquaLogic Service Bus Console displays the information captured from this reporting provider. If you do not wish to use the out-of-the-box reporting provider, you can untarget it and create your own reporting provider using the Reporting Service Provider Interface (SPI). If you configure your own reporting provider for messages, no information is displayed in the AquaLogic Service Bus Console and you will need to create your own user interface. If you wish to capture SLA data, you will need to create a reporting provider for alerts.

This chapter contains information on the following topics

- [Reporting Scenarios](#)
- [Reporting Framework](#)
- [JMS Reporting Provider](#)
- [How to Enable Message Reporting](#)
- [Removing, Stopping, or Untargeting a Reporting Provider](#)

## Reporting Scenarios

The following scenarios describe some of the ways in which you can use AquaLogic Service Bus to track messages:

### Message Tracking

In [“Alert Monitoring” on page 5-2](#), a scenario was described where the number of alerts had increased significantly. In that scenario, you filtered the alerts and discovered that the Service Level Agreement (SLA) violations were due to errors produced by the Post-Trade Processing proxy service. Reporting helps you find the source of the errors more easily.

To continue the scenario, you proceed to the AquaLogic Service Bus Console, where you filter messages to display the Post-Trade Processing proxy service. Drilling into some of the messages, you discover that the pipeline errors are due to message transformation errors and that the mangled messages are coming out of the portal associated with the proxy service. To solve the problem, a developer adds a new transformation to the pipeline for all messages originating from that portal site.

### Search for a Particular Message

Customer Service calls Operations with a complaint that the customer who submitted trade 1234 did not receive a trade confirmation. You access the AquaLogic Service Bus Console and search for the trade number. The search shows that two messages were processed in the request pipelines, but no response messages. You ask Customer Service to contact the customer and assure the customer that the trade was successfully processed.

### Logging for Regulatory Auditing

At the end of the month, the Mortgage Processing team must provide their compliance department with information about the mortgages they have processed. To fulfill this requirement, the Application Development team updates the applicable proxy service pipelines to capture the relevant message information. Specifically, data is extracted from the messages for the customer ID and name, mortgage ID, mortgage amount, property address, and the date the loan application was submitted.

### Alert Reporting Provider

By configuring a reporting provider for alerts you can receive an alert notification outside of the AquaLogic Service Bus Console and process the alert according to your business needs. For example, you could develop an alert reporting provider that utilizes the reporting stream for alerts and then display the alerts on a custom console, such as HP OpenView, or Tivoli.

# Reporting Framework

AquaLogic Service Bus contains an extensible framework for creating one or more reporting providers for messages or alerts.

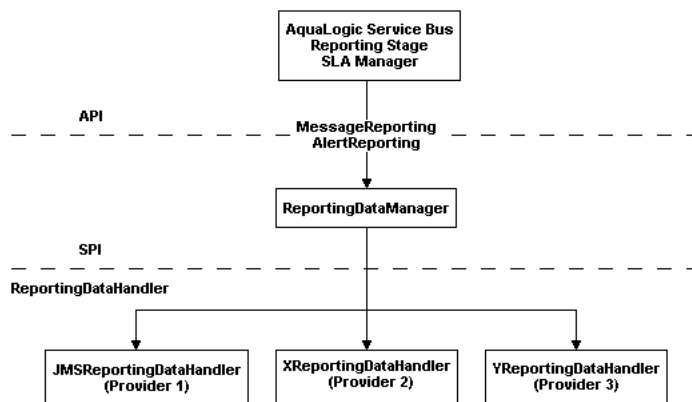
To enable message reporting you must first create a Report action in the message flow for the proxy service. The Report action allows you to extract information from each message and write it to the AquaLogic Service Bus Reporting Data Stream. You do not need to configure a report action for alert reporting. Alert data is always available in the Reporting Data Stream. For more information, see [“How to Enable Message Reporting” on page 6-5](#).

All the information you need to create your own reporting provider is located in `com.bea.wli.reporting` within the [Javadoc for AquaLogic Service Bus](#). The Javadoc provides information about what you need to do to implement a reporting provider, including how to package it, where it goes, how to deploy it, and the order of deployment. The location of the reporting schema (`MessageReporting.xsd`) is

`BEA_HOME/weblogic90/servicebus/lib/sb-schemas.jar`, where `BEA_HOME` is the location where you installed your BEA products.

The following figure shows the reporting framework.

**Figure 6-1 Reporting Framework**



As shown in the previous figure, both report messages and alerts are exported to reporting data streams. In the Report stage, information is extracted by the Report action from each message and written to the Reporting Data Stream with metadata that adheres to `MessageReporting.xsd`. Similarly, the SLA Manager uses Reporting Data Manager APIs to write to the Alert Reporting Stream with metadata that adheres to the `AlertReporting.xsd`. If you want to develop a

reporting provider for alerts or your own message reporting provider, you need to implement one interface called `ReportingDataHandler` and use one class called `ReportingDataManager`.

The `ReportingDataHandler` Interface takes the reporting or alert data stream and processes it. It can process and/or store this stream in a relational database, file, JMS queue, and so on. Depending on which stream you want to use, you need to implement the appropriate handle methods to process the data stream:

- **Message Reporting Stream**—the report action of AquaLogic Service Bus run time uses the following two handle methods to write to the Message Reporting Stream:

```
handle(com.bea.xml.XmlObject metadata, String s)
```

```
handle(com.bea.xml.XmlObject metadata, com.bea.xml.XmlObject data)
```

- **Alert Reporting Stream**—the Alert Manager uses the following handle method to write to the Alert Reporting Stream:

```
handle(com.bea.xml.XmlObject metadata, com.bea.xml.XmlObject data)
```

The `ReportingDataManager` is a server-local singleton object that keeps a registry of reporting providers. Reporting providers implement the `ReportingDataHandler` interface. The `ReportingDataManager` provides operations that add and remove reporting data handlers and exports reporting data stream using various handle operations.

## JMS Reporting Provider

The out-of-the-box JMS Reporting Provider provides a pluggable architecture to capture the reporting information from each message via a Report action. All messages across the cluster are aggregated and stored in the JMS Reporting Provider Data Store in a database specific format. When you use the out-of-the-box JMS Reporting Provider, the Reporting module in the AquaLogic Service Bus Console displays information from the JMS Reporting Provider Data Store.

**Note:** The JMS Reporting Provider is automatically configured when you create an AquaLogic Service Bus domain. If you do not wish to use this reporting provider, you must untarget it. For more information, see [“Removing, Stopping, or Untargeting a Reporting Provider” on page 6-14](#).

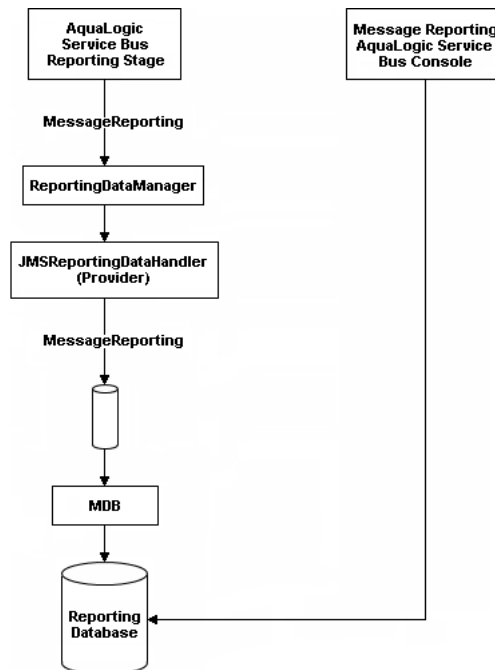
This section contains information on the following topics:

- [About the JMS Reporting Provider](#)
- [Using the Reporting Module](#)

## About the JMS Reporting Provider

The JMS Reporting Provider consists of a producer and a consumer, which are decoupled to improve scalability. The producer is a JMS producer and the MDB (Message Driven Bean) acts as the JMS consumer, as shown in the following diagram.

**Figure 6-2 JMS Reporting Provider**



The Reporting stage contains the Report actions that collect the reporting information and dispatch the reporting stream to JMS Reporting Provider through various `handle` operations in the `ReportingDataManager`. The `JMSReportingDataHandler` is the JMS producer of the reporting provider. The `JMSReportingDataHandler` takes the reporting stream and logs the information to a JMS queue. The `MDB` listens to the JMS reporting queue, which processes the message asynchronously and stores the data in the JMS Reporting Provider Data Store.

## How to Enable Message Reporting

To receive report messages from either the out-of-the-box JMS Reporting Provider or your reporting provider you must first create a Report action in the message flow for the proxy service.

The Report action allows you to extract information from each message and write it to the AquaLogic Service Bus Reporting Data Stream. In the Report action, you specify the information you want to extract from the message and add to the AquaLogic Service Bus Reporting Data Stream.

You do not need to configure a report action for alert reporting. Alert data is always available in the Reporting Data Stream.

When configuring a Report action, you use key values to extract key identifiers from the message. You can configure multiple keys. Information can be captured not only from the body of the message but any other variable associated with the message, such as header or inbound variables. For more information about message variables, see [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.


You can use any XML elements as a key:


```
<?xml version="1.0" encoding="utf-8"?>
<poIncoming>
  <areacode>408</areacode>
  <item-quantity>100</item-quantity>
  <item-code>ABC</item-code>
  <item-description>Medicine</item-description>
</poIncoming>
```

For example, you can specify the key as the `itemcode`, the value as `../item-code` (an xpath expression), and the variable as message body (`body`), as shown in the following figure.


**Figure 6-3 Key Name and Value**

Request Actions:

 **Report** `$body` with search keys:

Key Name	Key Value
 <code>itemcode</code>	<code>&lt;../item-code&gt;</code> in variable <code>body</code>

Response Actions:

 [Add an Action](#)

If you are using the out-of-the-box JMS Reporting provider, the keys and associated values are displayed in the Report Index column of the Summary of Messages table. If you configure multiple keys, the key-value pairs are displayed in Report Index Column with each key-value separated by a semicolon, as shown in the following figure.

**Figure 6-4 Keys and Associated Values Display**

Summary of Messages <span style="float: right;">Search</span>		
Report Index	DB TimeStamp	Inbound Service
Customer ID=EA-3822883	6/30/05 8:11 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway
Customer Name=John Smith	6/30/05 8:11 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway
Property Address=2315 North St, San Jose, CA 95131	6/30/05 8:11 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway
Date=6/20/05	6/30/05 8:11 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway
Mortgage Amount=\$778,900,Interest Rate=05.00%	6/30/05 8:11 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway

For information on how to create a Report action or on how to view the Summary of Messages page, see the following in the *AquaLogic Service Bus Console Online Help*:

- “Adding an Action” in [Proxy Services](#)
- “Listing and Locating Messages” in [Reporting](#)

## Using the Reporting Module

The reporting module in the AquaLogic Service Bus Console displays the information collected by the JMS Reporting Provider Data Store. The first page of the Reporting module, called the Summary of Messages, displays a table containing the extracted information and other information, such as the time the message was written to the database and the service with which the message is associated. You can customize the display of information on this page by filtering and sorting the data. Additionally, you can drill down to view detailed information about specific messages, including error information.

The Reporting module provides a purge functionality to help you manage your message data. You can purge all of the messages from the reporting datastore or base the purge on a range of time.

The JMS Reporting Provider Data Store requires a database. An evaluation version of the PointBase database is installed with WebLogic Server. You can use PointBase for a development environment but not for production. AquaLogic Service Bus also supports databases from other vendors. Be sure to apply standard database administration practices to the database hosting the JMS Reporting Provider Data Store. For more information, see [“Configuring a Database for the JMS Reporting Provider Store”](#) on page 6-13.

All information about how to use the reporting module is located in the *AquaLogic Service Bus Console Online Help*.





This section includes information on the following topics:

- [Summary of Messages](#)
- [View Message Details](#)
- [Purging Messages](#)

## Summary of Messages

When you click Reporting in the navigation panel, the Summary of Messages page is displayed. This page contains a table that provides a list of report messages sorted by the time the message was written in the database.

**Figure 6-5 Summary of Messages**

Summary of Messages <span>Search</span>			
Report Index 	DB TimeStamp 	Inbound Service 	Error Code 
errorCode=BEA-382000	6/15/05 5:05 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway3	BEA-382000
errorCode=BEA-382000	6/15/05 5:05 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway3	BEA-382000
errorCode=BEA-382000	6/15/05 5:01 PM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway3	BEA-382000
errorCode=BEA-382000	6/15/05 11:08 AM	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway3	BEA-382000

If the messages are not filtered, the Summary of Messages table displays up to 100 of the latest messages based on the database timestamp. If you filter the messages, up to 1000 messages are displayed.

**Note:** After you filter the message, the filter remains in effect until you update it.

The table shown in the preceding figure provides the following information:

- **Report Index**—displays the key-value pairs extracted from the message context variables or the message payload. For more information, see [“About the JMS Reporting Provider” on page 6-5](#).
- **DB TimeStamp**—the timestamp of the database when the message was registered.
- **Inbound Service**—the inbound service associated with the message. The service is a link to the View Proxy Service Details page.
- **Error Code**—the error code associated with this message if it exists. For more information about error codes, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

To search for specific messages, you can filter the display of messages by clicking Search in the Summary of Messages Table. The available filtering is shown in the following figure.



**Figure 6-6 Summary of Messages Search**

**Summary of Messages** [Close Search](#)

☐ Start Date: June 16 2005 4 53 PM  
☐ End Date: June 16 2005 4 53 PM  
☐ For the Last: 0 days 0 hours 00 mins

Inbound Service Name:   
 Error Code:   
 Report Index:


As shown in the previous figure, you can filter report messages for a specified period of time, by the name of a service, by error code, and by report index. After you filter the messages, the title of the page changes to Summary of Filtered Messages. For information on how to use the Summary of Messages filter, see “Listing and Locating Messages” in [Reporting](#) in the *AquaLogic Service Bus Console Online Help*.

To view more information about a report message, click the name of the message in the Report Index column. The View Message Details page is displayed.

## View Message Details

The View Message Details page displays complete information about the report messages, as shown in the following figure.

Figure 6-7 Report Message Detail Page

 View Message uuid:c91ab9e973f25cc1:17db5bc9:10480ef1244:-7fd9 Details

OK

General Configuration	
Message ID	uuid:c91ab9e973f25cc1:17db5bc9:10480ef1244:-7fd9
Database Timestamp	Wednesday, June 15, 2005 5:05:00 PM MDT
Time at point of Logging	Wednesday, June 15, 2005 5:05:00 PM MDT
Server Name	xbusServer
State	ERROR
Node Name	PipelinePairNode1
Pipeline Name	PipelinePairNode1_request
Stage Name	validate loan application
Inbound Service	
Name	ProxyService\$MortgageBroker\$ProxyServices\$loanGateway3
URI	/loan/gateway3
Operation	processLoanApp
Outbound Service	
Name	
URI	
Operation	
Report Index	
Report Index Text	errorCode=BEA-382000
Fault	
Error Code	BEA-382000
Reason	Decimal fractional digits (1) of value '10.1' does not match fractionDigits facet (0) for xs:int: <xml-fragment xmlns:java="java:normal.client" xmlns:m="http://example.org" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" />
Detail	<con:stack-trace xmlns:con="http://www.bea.com/wli/sb/context" xmlns:rep="http://www.bea.com/wli/reporting">com.bea.wli.sb.pipeline.PipelineException: Decimal fractional digits (1) of value '10.1' does not match fractionDigits facet (0) for xs:int:&lt;xml-fragment xmlns:java="java:normal.client" xmlns:m="http://example.org" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" /> ... </con:stack-trace>
Report Body	
Detail	<a href="#">Detail</a>

The page shows the following information:

- General Configuration
  - Message ID—the unique identification for this message.
  - Database Timestamp—the timestamp of the database when the message was registered.
  - Time at point of Logging—the date and time, on the server machine, that the message was reported.
  - Server name—the name of the server from which this message was generated.

- State—state of the pipeline from which this message was generated, as follows:
  - REQUEST—indicates that the reporting action was executed in a request pipeline.
  - RESPONSE—indicates that the reporting action was executed in a response pipeline.
  - ERROR—the action was running in the service-level error handler.
- Node Name—the pipeline node from which this message was generated.
- Pipeline Name—the pipeline from which this message was generated.
- Stage Name—the stage from which this message was generated.
- Inbound Service
  - Name—the inbound proxy service associated with this message. An inbound proxy service exchanges messages with client applications. The name is a link to the View Proxy Service Details page. For more information about this page, see “Viewing and Changing Proxy Services” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
  - URI—the URI associated with the proxy service.
  - Operation—the inbound operation associated with this message. Operations are the tasks performed by a pipeline or route node in the message flow associated with the service.
- Outbound Service
  - Name—the outbound business service associated with this message. An outbound business service exchanges messages with an AquaLogic Service Bus proxy service. The name is a link to the View Business Service Details page. For more information about this page, see “Viewing and Changing Business Services” in [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.
  - URI—the URI to the outbound business service end point.
  - Operation—name of the operation invoked on the outbound service. Operations are the tasks performed by a pipeline or route node in the message flow associated with the service.
- Report Index
  - Report Text Index—displays the key-value pairs extracted by a Report Action from the message context variables or the message payload. For more information, see [“About the JMS Reporting Provider” on page 6-5](#).

- Fault
  - Error Code—the code associated with the message, if it exists. For more information, see “Error Messages and Handling” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
  - Reason—the reason for the error code.
  - Detail—The fault details associated with the error code. These details, if present, are typically a stack trace of where a particular fault occurred, which may be truncated due to a size limitation in the database. The limit is 2048 characters.
- Report Body
  - Detail—opens a browser window that displays the report body. You use an XQuery expression in a Report Action to capture the report body text. For more information, see “Adding an Action” and “Editing an XQuery Expression” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Purging Messages

You can purge all of the messages from the reporting datastore or base the purge on a range of time. Message purging is an asynchronous process that occurs in the background of the AquaLogic Service Bus Console. This feature prevents the Summary of Messages page in the AquaLogic Service Bus Console from being locked up while the purge occurs.

**Figure 6-8 Purging Messages Page**

**Set Reporting Data Store Purge Policy**

☒ Purge All Messages

☐ Purge From

Purge From: January 1 2005 5 07 PM

Purge To: June 30 2005 5 07 PM

Submit

The length of time it takes a purge to complete depends on how many messages are in the purge. The deletion of messages is slowed if you search for reporting messages during the purge process. Additionally, the Summary of Messages page may display incorrect data as some data may not yet be purged.

Because the purge process is asynchronous and occurs in the background, the AquaLogic Service Bus Console does not display any messages to indicate that a purge is in process. However, if another user attempts to start a purge when a purge is already taking place, the following message is displayed:

**A Purge job is already running. Please try later.**

## Configuring a Database for the JMS Reporting Provider Store

AquaLogic Service Bus requires a database for the JMS Reporting Provider Data Store. The PointBase database that is installed with WebLogic Server is for evaluation purposes only and not intended for a production environment. Non-evaluation development or other use of the PointBase Server requires that you obtain a separate PointBase license directly from DataMirror.

In a production environment you must use one of the supported databases. For the latest information about supported databases, see “Supported Databases and Drivers” in [Supported Configurations for WebLogic Platform](#) in *Supported Configurations for AquaLogic Service Bus*.

This section contains information on the following topics:

- [Configuring a Database in a Development Environment](#)
- [Configuring a Database for Production](#)

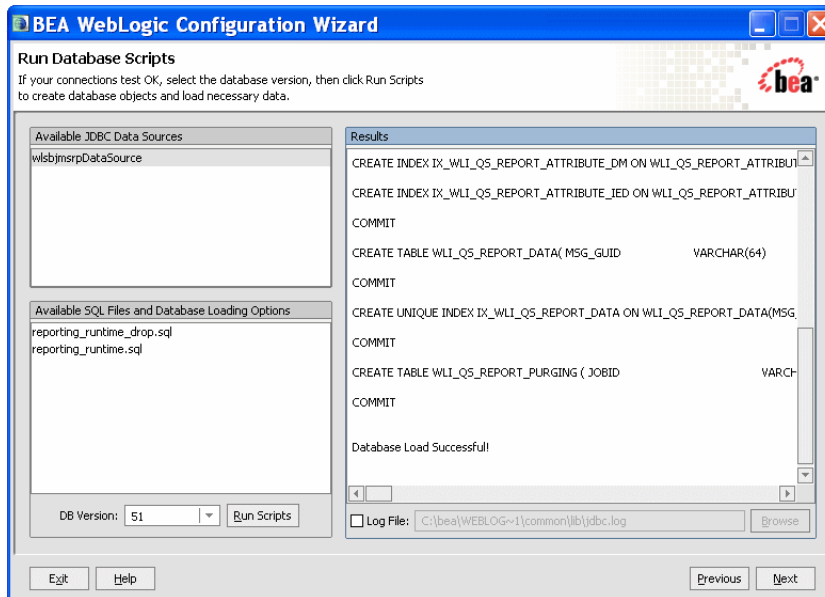
### Configuring a Database in a Development Environment

When you create an AquaLogic Service Bus domain, the Configuration Wizard does not create database tables automatically. In a development environment, the out-of-the-box JMS Reporting Provider checks whether the tables exist for the specified database at run time. If the tables do *not* exist, the Reporting Provider creates them; if they do exist, the Reporting Provider uses them.

**Note:** If you are using Pointbase, you do not need to specify a database in the Configuration Wizard.

You can specify which database is used by the JMS Reporting Provider in one of the following ways:

- Run the reporting SQL scripts in your AquaLogic Service Bus domain. The scripts are located in `BEA_HOME/weblogic90/integration/common/dbscripts`, where `BEA_HOME` represents the location in which you installed your WebLogic products.
- When you create your domain in the Configuration Wizard, customize the JDBC Settings on the Run Database Scripts page (see [Figure 6-9](#)). For more information, see [Creating WebLogic Domains Using the Configuration Wizard](#).

**Figure 6-9 Run Database Scripts in the Configuration Wizard**

## Configuring a Database for Production

Complete information about configuring a database for production is located in the *BEA AquaLogic Service Bus Deployment Guide* in the following chapters:

- [Configuring a Single-Server Deployment](#)
- [Configuring a Cluster Deployment](#)

## Removing, Stopping, or Untargeting a Reporting Provider

As previously mentioned, the out-of-the-box JMS Reporting Provider is automatically configured when you create an AquaLogic Service Bus domain. If you do not wish to use this reporting provider or any reporting provider, you must untarget it.

**Caution:** If no reporting provider exists, you can still define a Report action. However, no data will be written.

The following sections provide information on how to stop or untarget any reporting provider:

- [Stopping a Reporting Provider when the Server is Running](#)

- [Untargeting a Reporting Provider when the Server is Running](#)
- [Untargeting the JMS Reporting Provider—Server Not Running](#)

## Stopping a Reporting Provider when the Server is Running

If you wish to stop a reporting provider when the server is running in the AquaLogic Service Bus domain, take the following steps:

1. Start the WebLogic Server Administration Console. For more information, see “Starting the Administration Console” in [Overview of the Administration Console](#) in *Introduction to WebLogic Server and WebLogic Express*.
2. After logging into the WebLogic Server Administration Console, in the Domain Structure, click Deployments. The Summary of Deployments page is displayed.
3. In the Deployments table, select the check box next to the reporting provider you wish to stop.

**Figure 6-10 Stopping a Reporting Provider**

Control | **Monitoring**

This page displays a list of J2EE Applications and standalone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

**Deployments**

Install Update Delete Start Stop

Showing 1 - 1 of 1 Previous Next

<input type="checkbox"/>	Name	State	Type	Deployment Order
<input type="checkbox"/>	creditLoanJWSBasicEjb	Active	EJB	100
<input type="checkbox"/>	Email Transport Provider	Active	EJB	152
<input type="checkbox"/>	examplesWebApp	Active	Web Application	100
<input type="checkbox"/>	File Transport Provider	Active	Web Application	153
<input type="checkbox"/>	Ftp Transport Provider	Active	EJB	151
<input checked="" type="checkbox"/>	JMS Reporting Provider	Active	Enterprise Application	125
<input type="checkbox"/>	largeLoanJWSBasicEjb	Active	EJB	100

4. Click stop and after the list is displayed, choose the appropriate command.
5. After the Stop Application Assistant page is displayed, click Yes. The Deployments table shows that the state of the reporting provider is now Prepared.

## Untargeting a Reporting Provider when the Server is Running

If you wish to untarget a reporting provider when the server is running in the AquaLogic Service Bus domain, take the following steps:

1. Start the WebLogic Server Administration Console. For more information, see “Starting the Administration Console” in [Overview of the Administration Console](#) in *Introduction to WebLogic Server and WebLogic Express*.
2. After logging into the WebLogic Server Administration Console, in the Change Center, click Lock & Edit.
3. From the left panel, under Domain Structure, click Deployments. The Summary of Deployments page is displayed.
4. In the Deployments table, click the reporting provider you wish to untarget. The Settings page for the Reporting Provider is displayed.
5. Click the Targets tab.
6. Clear the appropriate check box.

**Figure 6-11 Untargeting a Reporting Provider**



7. Click Save. A message is displayed indicating that the settings have been successfully updated.
8. After you untarget the reporting provider, untarget the data source used by the reporting provider, as follows:

**Note:** This step is only required for reporting providers that use their own data sources. If you are untargeting the out-of-the-box JMS Reporting Provider, you must perform this step.

- a. In the left panel, under Domain Structure, select **Services→JDBC→Data Sources**.



- b. In the Summary of JDBC Data Source page, click the name of the data source you wish to untarget. The Settings page for the data source is displayed.
- c. Click the Targets tab.
- d. Clear the appropriate check box.
- e. Click Save. A message is displayed indicating that the settings have been successfully updated.
- f. To activate the changes, in the Change Center, click Activate Changes.

## Untargeting the JMS Reporting Provider—Server Not Running

If the server is not running in the AquaLogic Service Bus domain, you can use the WebLogic Scripting Tool (WLST) to remove the JMS Reporting Provider from the AquaLogic Service Bus domain. For more information about WLST, see [WebLogic Scripting Tool](#) in the WebLogic Server documentation.

To untarget a reporting provider, complete the following steps:

1. If you have not already set up your environment to use WLST, see “Main Steps for Using WLST” in [Using the WebLogic Scripting Tool](#) in *WebLogic Scripting Tool*.
2. Open a UNIX shell or command window.

3. Invoke WLST Offline.

```
C:>java com.bea.plateng.domain.script.jython.WLST_offline
```

4. Read the domain that was created using the Configuration Wizard. For example:

```
wls:/offline>readDomain("C:/bea/user_projects/domains/base_domain")
```

5. Untarget the reporting provider data source. For example:

```
wls:/offline/base_domain>unassign("JdbcSystemResource", "wlsbjmsrpDataSource", "Target", "AdminServer")
```

6. Untarget the reporting provider application. For example:

```
wls:/offline/base_domain>unassign("AppDeployment", "JMS Reporting Provider", "Target", "AdminServer")
```

7. Update the domain:

```
wls:/offline/base_domain>updateDomain()
```

8. Close the domain:

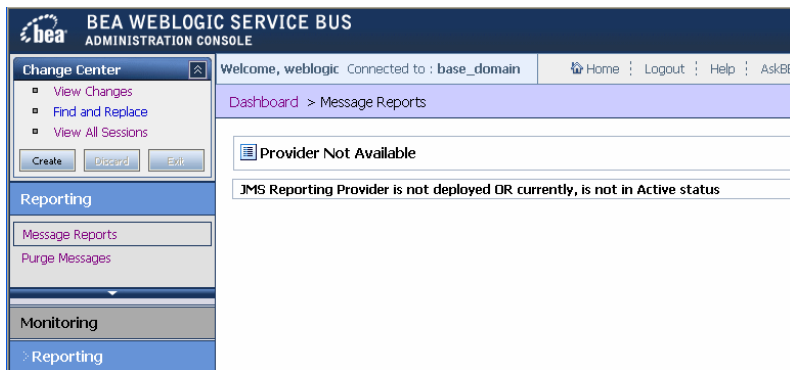
```
wls:/offline/base_domain>closeDomain()
```

9. Exit from the WLST command prompt:

```
wls:/offline>exit()
```

After the out-of-the-box reporting provider is untargeted, the Reporting module in the AquaLogic Service Bus Console will indicate that the reporting provider is not deployed, as shown in the following figure.

**Figure 6-12 Reporting Provider Not Deployed**



**Note:** In a cluster, the JMS Reporting Provider is targeted to Cluster. Therefore in a cluster, to view and purge messages, you must configure at least one managed server to run with the Administration server. If no managed servers are running, AquaLogic Service Bus Console displays the message shown in the previous figure.

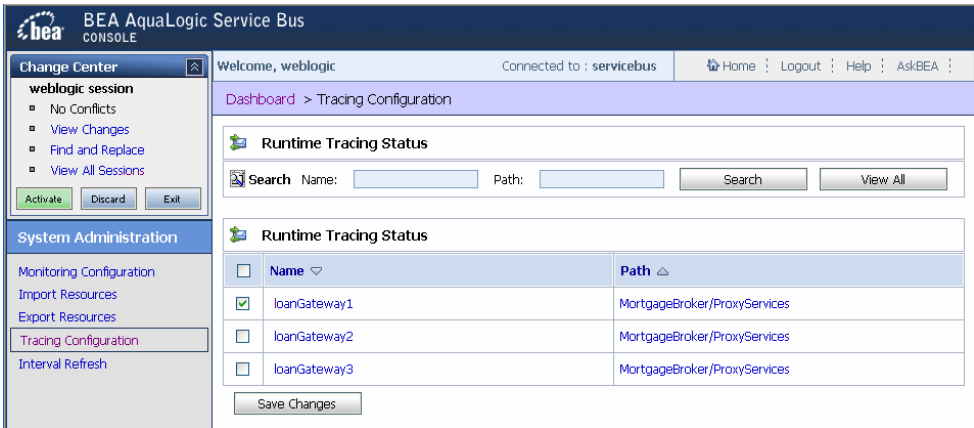
# Tracing

BEA AquaLogic Service Bus provides the capability to trace messages without having to shut down the server. This feature is useful in both a development and production environment. Tracing allows administrators, support engineers, and systems engineers to troubleshoot and diagnose a message flow in one or more proxy services.

For example, if one of your proxy services is failing and you want to find out at which stage the problem exists, you can enable tracing for that proxy service. After tracing is enabled, the system logs various details extracted from the message flow such as stage name, name of the pipeline, and route node name. The entire message context is also printed including headers and message body. In case of a fault in the message flow, additional details such as error code and reason are logged. Tracing occurs at the beginning and end of each component in the message flow, which includes stages, pipelines, and nodes (actions are not traced individually).

You enable tracing in the System Administration module of the AquaLogic Service Bus Console, as shown in the following figure.

Figure 7-1 Tracing Configuration



As shown in the preceding figure, the Tracing Configuration page displays the tracing status of the proxy services. If the check box adjacent to the name of the proxy service is selected, tracing is enabled for that service. The Runtime Tracing Status table displays the following information:

- Name—the name of the proxy service. The name is a link to the View Proxy Service Details page.
- Path—the project name and the name of the folder in which the proxy service resides. It is a link to the Project Details or Folder Details page.

Information about the pages referenced from the Runtime Tracing Status table is available in the *AquaLogic Service Bus Console Online Help*, as follows:

- View Proxy Service Details page—“Viewing and Changing Proxy Services” in [Proxy Services](#)
- Project Details—“Viewing Project Details” in [Project Explorer](#)
- Folder Details—“Viewing Folder Details” in [Project Explorer](#)

For information on how to use the AquaLogic Service Bus Console to enable tracing, see “Enabling Runtime Tracing Status of Proxy Services” in [System Administration](#) in the *AquaLogic Service Bus Console Online Help*.

**Note:** Remember to activate the session to start logging. Once the session has been activated, the trace setting is persisted along with the other details of the proxy service configuration.

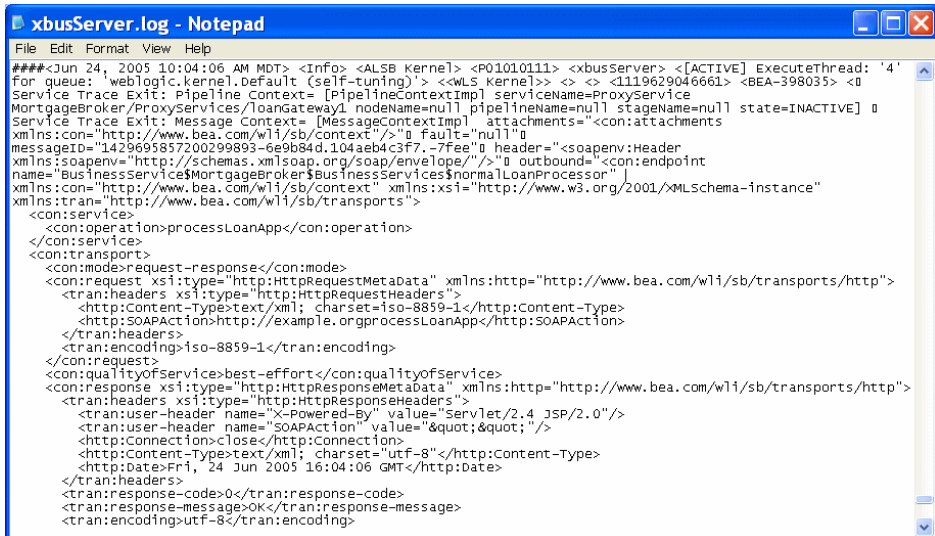
The tracing information is placed in the server directory logs. For example, in the AquaLogic Service Bus Examples, the tracing information is logged in the following directory.

`BEA_HOME\weblogic90\samples\domains\servicebus\servers\xbusServer\logs\xbusServer.log`

In the preceding paragraph, `BEA_HOME` is the directory in which you installed your BEA products.

The following figure shows a sample of the tracing log.

**Figure 7-2 Tracing Log Example**



```
####Jun 24, 2005 10:04:06 AM MDT> <Info> <ALSB Kernel> <P01010111> <xbusServer> <[ACTIVE] ExecuteThread: '4'
for queue: weblogic.kernel.Default (self-tuning)> <<WLS Kernel>> <> <<119629046661> <BEA-398035> <0
Service Trace Exit: Pipeline Context= [PipelineContextImpl serviceName=ProxyService
MortgageBroker/ProxyServices/loangateway1 nodeName=null pipelineName=null stageName=null state=INACTIVE] 0
Service Trace Exit: Message Context= [MessageContextImpl attachments="<con:attachments
xmlns:con="http://www.bea.com/wli/sb/context"/>"0 fault="null"0
messageID="1429695857200299893-6e9b84d.104aeb4c3f7.-7fee"0 header="<soapenv:Header
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">"0 outbound="<con:endpoint
name="BusinessService$MortgageBroker$BusinessServices$normalLoanProcessor"
xmlns:con="http://www.bea.com/wli/sb/context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tran="http://www.bea.com/wli/sb/transport">
<con:service>
<con:operation>processLoanApp</con:operation>
</con:service>
</con:service>
<con:transport>
<con:mode>request-response</con:mode>
<con:request xsi:type="http:HttpRequestMetadata" xmlns:http="http://www.bea.com/wli/sb/transport/http">
<tran:headers xsi:type="http:HttpRequestHeaders">
<http:Content-Type>text/xml; charset=iso-8859-1</http:Content-Type>
<http:SOAPAction>http://example.org/processLoanApp</http:SOAPAction>
</tran:headers>
<tran:encoding>iso-8859-1</tran:encoding>
</con:request>
<con:qualityofService>best-effort</con:qualityofService>
<con:response xsi:type="http:HttpResponseMetadata" xmlns:http="http://www.bea.com/wli/sb/transport/http">
<tran:headers xsi:type="http:HttpResponseHeaders">
<tran:user-header name="X-Powered-By" value="Servlet/2.4 JSP/2.0"/>
<tran:user-header name="SOAPAction" value="&quot;&quot;"/>
<http:Connection>close</http:Connection>
<http:Content-Type>text/xml; charset=utf-8</http:Content-Type>
<http:Date>Fri, 24 Jun 2005 16:04:06 GMT</http:Date>
</tran:headers>
<tran:response-code>0</tran:response-code>
<tran:response-message>OK</tran:response-message>
<tran:encoding>utf-8</tran:encoding>
```

## Tracing

# Tuning AquaLogic Service Bus

This appendix provides AquaLogic Service Bus tuning tips.

- Whenever possible, set the logging level to *warning*. You set the logging level in the WebLogic Server Administration Console. For more information, see [Servers: Logging: General](#) in the *WebLogic Server Administration Console Online Help*. The following code displays the output server `config.xml` file when the logging level is set to warning. For more information on logging, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *Log* action.

```
<server>
  <name>AdminServer</name>
  <log>
    <file-min-size>5000</file-min-size>
    <log-file-severity>Warning</log-file-severity>
    <log-file-filter xsi:nil="true"></log-file-filter>
    <stdout-severity>Off</stdout-severity>
    <stdout-filter xsi:nil="true"></stdout-filter>
    <domain-log-broadcast-severity>Error</domain-log-broadcast-severity>
    <domain-log-broadcast-filter
xsi:nil="true"></domain-log-broadcast-filter>
    <memory-buffer-severity>Error</memory-buffer-severity>
    <memory-buffer-filter xsi:nil="true"></memory-buffer-filter>
  </log>
```

</server>

- Group JMS queues on different JMS servers based on message loads. Different JMS servers use different file stores, which you can distribute to separate disk volumes. For more information, “Adding a Business Service” in [Business Services](#) in the *AquaLogic Service Bus Console Online Help* and pay particular attention to the *JMS* configuration information.
- If you are using an Oracle database as a JMS persistent store, it is recommended that you use a 10g database and ensure that it has sufficient JDBC connections. Create a JDBC store on a separate schema to use a separate tablespace.
- If you do not require monitoring for a proxy or business service, disable the monitoring capability. For more information, see “Overview of Monitoring” in [Monitoring](#) in the *AquaLogic Service Bus Console Online Help*.
- If possible, set the routing data in the JMS message properties. AquaLogic Service Bus does not deserialize message content until the content is explicitly accessed in the pipeline. For example, if the content is an XML document, XML parsing does not happen until an XQuery or XSLT operation happens in the pipeline. For more information about working with the message context in the message flow, see [Message Context](#) in the *AquaLogic Service Bus Console Online Help*.
- If you need to extract some of the inbound header elements for processing, you should specify that AquaLogic Service Bus retrieves specific header elements instead of all the elements.
- Where possible, use the insert action instead of the assign action. The insert action uses “in-place” modification semantics making it more performant compared to the assign action. For information on configuring actions, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.
- Use AquaLogic Service Bus clustering and WebSphere MQ clustering to achieve scalability.
- If a front end application invokes AquaLogic Service Bus synchronously, AquaLogic Service Bus can use the sync-async feature to communicate with WebSphere MQ synchronously. On the WebSphere MQ side, a request and a response queue is set up. AquaLogic Service Bus sends a request to the request queue and waits for a response from the response queue. To achieve improved performance, you can use a dedicate work manager for the response Message Driven Bean. You configure the dedicate work manager in the WebLogic Server Administration Console. For more information, see [Work Manager](#) in the *WebLogic Server Administration Console Online Help*. The following code displays the output server `config.xml` file after the dedicate work manager is configured.



```
<self-tuning>
  <min-threads-constraint>
    <name>minThreadsConstraint</name>
    <target>AdminServer</target>
    <count>20</count>
  </min-threads-constraint>
  <work-manager>
    <name>MQWorkManager</name>
    <target>AdminServer</target>
    <min-threads-constraint> minThreadsConstraint
  </min-threads-constraint>
    <ignore-stuck-threads>false</ignore-stuck-threads>
  </work-manager>
</self-tuning>
```



# Debugging AquaLogic Service Bus

This section provides information about enabling debugging for different modules in AquaLogic Service Bus. You can enable and disable debugging by modifying the corresponding entries in the `wldebug.xml` file, which is located in the root directory of the AquaLogic Service Bus domain. If the `wldebug.xml` file is not in the root directory or if it has been deleted, it is created again without any contents when the server starts. The following listing provides an example of the contents of the `wldebug.xml` file with debugging disabled for all modules (all entries set to `false`).

## Listing B-1 `wldebug.xml` File

---

```
<?xml version='1.0' encoding='UTF-8'?>
<java:wli-debug-logger xmlns:java="java:com.bea.wli.debug">
  <n1:Name xmlns:n1="java:weblogic.diagnostics.debug">wldebug</n1:Name>
  <java:wli-management-debug>false</java:wli-management-debug>
  <java:wli-monitoring-debug>false</java:wli-monitoring-debug>
  <java:wli-management-dashboard-debug>false</java:wli-management-dashboard-debug>
  <java:wli-config-debug>false</java:wli-config-debug>
  <java:wli-config-transaction-debug>false</java:wli-config-transaction-debug>
  <java:wli-config-deployment-debug>false</java:wli-config-deployment-debug>
  <java:wli-config-component-debug>false</java:wli-config-component-debug>
  <java:wli-sb-transport-debug>false</java:wli-sb-transport-debug>
  <java:wli-sb-pipeline-debug>false</java:wli-sb-pipeline-debug>
```

```
<java:wli-alert-manager-debug>>false</java:wli-alert-manager-debug>

<java:wli-jms-reporting-provider-debug>>false</java:wli-jms-reporting-provider-debug>

<java:wli-monitoring-aggregator-debug>>false</java:wli-monitoring-aggregator-debug>

< java:wli-credential-debug >>false</java:wli-credential-debug >

< java:wli-management-common-debug >>false</java:wli-management-common-debug >

</java:wli-debug-logger>
```

---

Although debugging should be disabled during normal AquaLogic Service Bus operation, you may find it helpful to turn on certain debug flags while you are developing your solution and experimenting with it for the first time. For example, you may want to turn on the alert debugging flag when you are developing alerts and would like to investigate how the alert engine works.

Some of the available debug flags are:

- `wli-config-debug`—Provides information on general aspects of AquaLogic Service Bus configuration.
- `wli-config-deployment-debug`— Provides debug information on session creation, activation, and distribution of configuration in a cluster.
- `wli-config-transaction-debug`—Provides low level debug information about changes made to in-memory data structures and files. This alert flag also generates server startup recovery logs.
- `wli-config-component-debug`—Provides low level debug information about create, update, delete, and import operations.
- `wli-sb-transport-debug`—Provides transport related debug information, including transport headers, which is printed per-message.
- `wli-sb-pipeline-debug`—Prints errors that are generated within the pipeline.
- `wli-alert-manager-debug`—Prints an evaluation of alerts.

All other debug flags are self explanatory.

For all flags, debug information is logged to the server log at

`{domaindir}/servers/{servername}/logs/{servername}.log`, except for the

wli-monitoring-aggregator-debug flag. The wli-monitoring-aggregator-debug flag enables debugging for aggregator. This flag logs the aggregated document every minute and stores the log files in the {domain}\monitoring folder.

**Note:** Turning the wli-monitoring-aggregator-debug flag on generates large amounts of debug data. Therefore, you should only use this flag for debugging purposes for short periods of time.

