



# **BEA AquaLogic Service Bus™**

## **Concepts and Architecture**

# 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. Overview

Introducing AquaLogic Service Bus . . . . .	1-2
AquaLogic Service Bus Core Feature Set . . . . .	1-4
AquaLogic Service Bus Architecture . . . . .	1-8
Deployment Topology . . . . .	1-8
Development, Staging, and Production Domains . . . . .	1-9
AquaLogic Service Bus Message Brokering . . . . .	1-9
Business Services and Proxy Services . . . . .	1-10
Message Processing . . . . .	1-11

## 2. Resource and Service Configuration Concepts

Resources . . . . .	2-1
Schemas and Data Types . . . . .	2-2
Transformation Maps . . . . .	2-2
WSDLs . . . . .	2-3
WS-Policy . . . . .	2-3
Services . . . . .	2-3
Service Types . . . . .	2-4
Service Transports . . . . .	2-4
Service Interfaces . . . . .	2-5
Proxy Services and Proxy Service Providers . . . . .	2-5
Business Services and Service Accounts . . . . .	2-9

# 3. System Administration and Operation Monitoring Concepts

- Security Management . . . . . 3-1
  - User Management . . . . . 3-2
  - Console Security . . . . . 3-3
  - Transport-Level Security . . . . . 3-3
  - Message-Level Security . . . . . 3-4
- Configuration Metadata Export and Import . . . . . 3-5
- Dashboard, System Metrics, and Alerts . . . . . 3-5
  - Dashboard . . . . . 3-5
  - Metric Aggregation . . . . . 3-7
  - Alerts . . . . . 3-7
- Message Reporting . . . . . 3-8

# Overview

BEA AquaLogic Service Bus—part of BEA’s new family of Service Infrastructure Products (AquaLogic)—combines intelligent message brokering with service monitoring and administration to provide a unified software product for implementing and deploying your Service-Oriented Architecture (SOA). This converged approach adds a scalable, dynamic routing and transformation layer to your enterprise infrastructure, plus service lifecycle management capabilities for service registration, service usage, and Service Level Agreement (SLA) enforcement.

AquaLogic Service Bus is a configuration-based, policy-driven Enterprise Service Bus (ESB). It provides a feature-rich console for dynamic service and policy configuration, as well as for system monitoring and operations tasks. 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 topic introduces AquaLogic Service Bus. It is intended for enterprise architects, operations specialists, and security architects who are responsible for messaging and SOA. It includes the following sections:

- [Introducing AquaLogic Service Bus](#)
- [AquaLogic Service Bus Core Feature Set](#)
- [AquaLogic Service Bus Message Brokering](#)
- [AquaLogic Service Bus Architecture](#)

## Introducing AquaLogic Service Bus

Enterprise architects responsible for messaging and SOA initiatives face a number of challenges in today's enterprise:

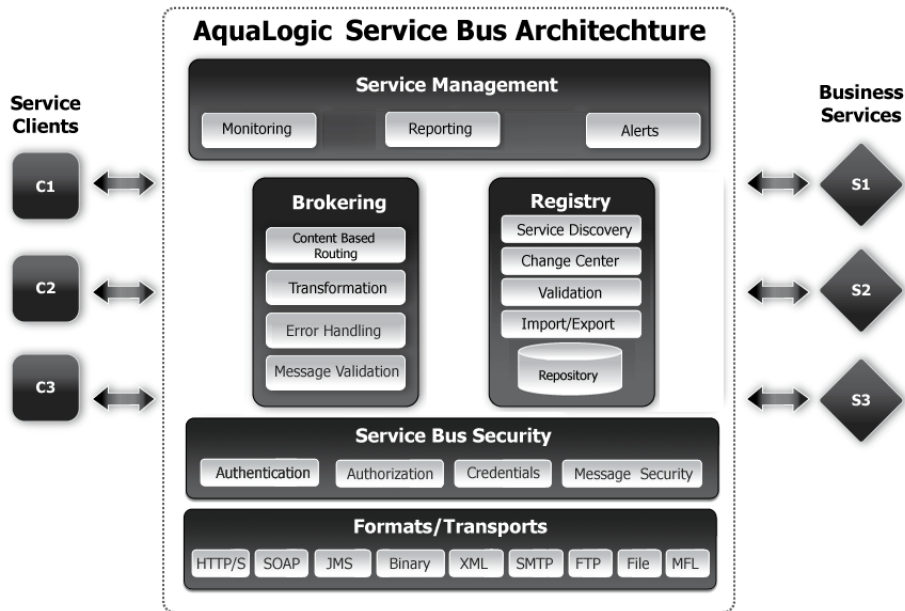
- Introducing dynamic behavior and run-time configuration capabilities into a system
- Reusing services that are developed across the enterprise and managing their lifecycle
- Ensuring consistent use of the enterprise services
- Ensuring enterprise services are secure
- Ensuring that the enterprise services comply with the IT policies
- Monitoring and auditing service usage and managing system outages

In short, the goal of the enterprise architect and other specialists is to reuse, streamline, and maintain control over their IT infrastructure. AquaLogic Service Bus is designed to help you achieve these goals.

AquaLogic Service Bus is a true ESB product specifically targeted for service-oriented integration, managing Web Services, and providing traditional message brokering across heterogeneous IT environments. AquaLogic Service Bus's lightweight, stateless, high-performance architecture delivers an intermediary for use as a core element of distributed services networks.

AquaLogic Service Bus is policy-driven. It enables you to establish loose coupling between *service clients* and *business services* while maintaining a centralized point of security control and monitoring as shown in the following figure.

Figure 1-1 AquaLogic Service Bus High-Level Architecture



With AquaLogic Service Bus you implement service integration relationships dynamically through configuration of *policies* and *proxy services*. This approach enables your service architectures to evolve rapidly with respect to the following system characteristics:

- Security
- Service location
- Availability and responsiveness
- Data formats
- Logging and monitoring
- Transport protocols and communications paradigms

Because of the intermediary nature of the proxy service, you can use AquaLogic Service Bus to resolve differences between service client and business service requirements in the following areas:

- Payload contents and schema
- Envelope protocols

- Transport protocols
- Point-to-point and publish and subscription protocols
- One way and request/response paradigms
- Synchronous and asynchronous communication
- Security compliance

AquaLogic Service Bus persists policy, proxy service, and related resource configurations in metadata that you can propagate from development through staging to production environments, and then modify as required. The AquaLogic Service Bus message brokering engine accesses this configuration information from its metadata repository.

A key design philosophy of AquaLogic Service Bus is the physical separation of management functions from service implementations. This separation allows implementations to evolve independently and dynamically as driven by the needs of the business without requiring costly infrastructure development efforts. As part of an enterprise's messaging fabric, AquaLogic Service Bus can be used horizontally across many applications and systems, spanning service implementations in different departments built by different teams.

## AquaLogic Service Bus Core Feature Set

By fusing the concepts of ESB, message brokering, and Web Services Management (WSM) into a single product, AquaLogic Service Bus provides the foundation for management and integration of messages and services.



The following table describes the core features of AquaLogic Service Bus.

**Table 1-1 AquaLogic Service Bus Core Features**

For this feature . . .	AquaLogic Service Bus . . .
<b>Routing</b>	Supports the following: <ul style="list-style-type: none"> <li>• Routes messages according to XQuery-based policies or callouts to external Web services.</li> <li>• Routing policies apply to both point-to-point and one-to-many routing scenarios (publish). For publish, routing policies serve as subscription filters.</li> </ul>
<b>Routing Transport Protocols</b>	Supports the following: <ul style="list-style-type: none"> <li>• File</li> <li>• FTP</li> <li>• HTTP(S)</li> <li>• JMS (including MQ using JMS, and JMS/XA)</li> <li>• E-mail (POP/SMTP/IMAP)</li> </ul>
<b>Messaging</b>	Supports the following models: <ul style="list-style-type: none"> <li>• Synchronous</li> <li>• Asynchronous</li> <li>• Publish</li> <li>• Subscribe</li> </ul>
<b>Message Types</b>	Supports the following message formats: <ul style="list-style-type: none"> <li>• E-mail with attachments</li> <li>• JMS with headers</li> <li>• MFL (Message Format Language)</li> <li>• Raw Data. Raw data is opaque data—that is, non-XML data for which there is no MFL file and therefore no known schema</li> <li>• Text</li> <li>• SOAP</li> <li>• SOAP with attachments</li> <li>• XML (XML that is or is not valid against a schema)</li> </ul>

For this feature . . .	AquaLogic Service Bus . . .
<b>Transformations</b>	<p>Supports the following functionality for the transformation or processing of messages:</p> <ul style="list-style-type: none"><li>• Validates incoming messages against schemas</li><li>• Selects a target service or services, based on the message content or message headers</li><li>• Transforms messages based on the target service</li><li>• Transforms messages based on XQuery or XSLT</li><li>• Supports transformations on both XML and MFL messages</li><li>• Message enrichment</li><li>• Supports call outs to Web services to gather additional data for transformation (for example, country code, full customer records, and so on.)</li></ul>
<b>Logging and Monitoring</b>	<p>Provides a rich set of functionality to audit and monitor services:</p> <ul style="list-style-type: none"><li>• You can gather statistics about message invocations, errors, performance characteristics, messages passed, SLA violations, and so on.</li><li>• The system also supports logging messages for both systems operations and business auditing purposes, search capabilities, and so on.</li><li>• Incoming messages may be logged to a reporting store. You can extract key information from a message and use it as a search index.</li><li>• The AquaLogic Service Bus Console provides a cluster-wide view of service status and statistics.</li><li>• Both business services and AquaLogic Service Bus proxy services are monitored, as are response times, message counts, and error counts.</li><li>• Statistics are gathered locally, then aggregated centrally.</li><li>• SLA rules run against aggregated data. The system raises alerts, and you can enable or disable services.</li></ul>
<b>Versioning</b>	<p>Provides the ability to deploy new versions of services and allow you to have multiple versions of message resources such as WSDLs and schemas. Versions can include changes to the WSDL, the message schema, the headers, and the security parameters.</p>

For this feature . . .	AquaLogic Service Bus . . .
<b>Service Level Agreements</b>	<p>Administrators can set service level agreements (SLAs) on the following attributes of proxy services:</p> <ul style="list-style-type: none"> <li>• Average processing time of a service</li> <li>• Processing volume</li> <li>• Number of errors, security violations, and schema validation errors</li> <li>• Administrators can configure alerts for SLA rule violations</li> </ul>
<b>Security</b>	<p>Includes the following:</p> <ul style="list-style-type: none"> <li>• Supports authentication, encryption and decryption, and digital signatures as defined in the Web Services Security (WS-Security) specification.</li> <li>• Uses SSL to support traditional transport-level security for HTTP and JMS transport protocols.</li> <li>• Supports one-way and two-way certificate based authentication.</li> <li>• Supports HTTP basic authentication.</li> </ul>
<b>Service Registry</b>	<p>Supports the following:</p> <ul style="list-style-type: none"> <li>• Stores information about services, schemas, transformations, WSDLs (Web Service Definition Language), and WS Policies.</li> <li>• Provides centralized management and distributed access</li> <li>• Allows you to browse the service registry and import resources into the registry from WebLogic Workshop or other applications.</li> <li>• Allows the propagation of configuration data from environment to environment (for example, from a development domain to a test domain to a production domain). The system allows environment specific settings to be overridden during import.</li> </ul>
<b>Error Handling</b>	<p>Supports the following:</p> <ul style="list-style-type: none"> <li>• Allows you to configure your system to format and send error messages, and return messages for consumers of services who expect a synchronous response.</li> <li>• Allows you to configure error handling for stages in the pipeline, for pipelines, and for proxy services</li> </ul>

## AquaLogic Service Bus Architecture

AquaLogic Service Bus is an intermediary that takes in messages, processes them to determine where to route them, and transforms them as specified. It receives messages through a transport protocol such as JMS and sends out messages again through the same or another specified transport protocol. Message response follows the inverse path. The message processing by AquaLogic Service Bus is driven by metadata specified as the *message flow definition* for a proxy service in the AquaLogic Service Bus Console.

## Deployment Topology

AquaLogic Service Bus runs on WebLogic Server 9.0 and can be deployed in the following configurations:

- On a single server that also serves as the administration server.
- On a cluster of servers. A cluster consists of a clustered set of managed servers that perform the message processing. A domain has only one cluster and that cluster has AquaLogic Service Bus deployed to it. This cluster can host other applications in addition to AquaLogic Service Bus. There is one administration server in a clustered domain.

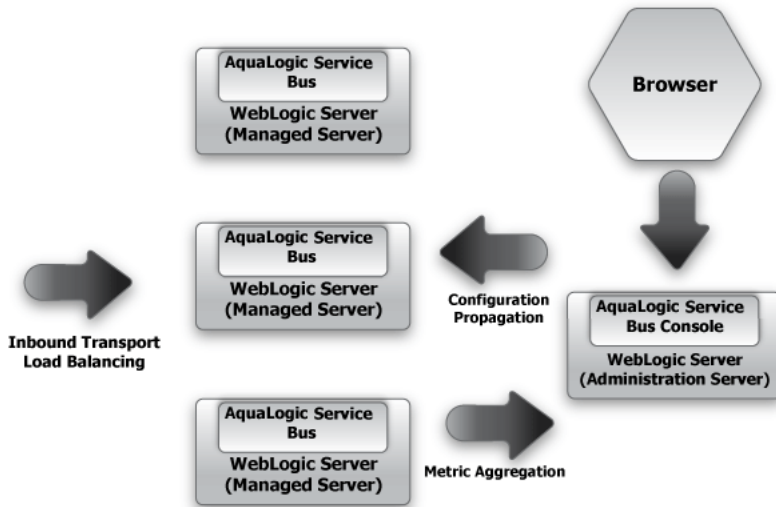
AquaLogic Service Bus can manage and control many distributed service endpoints, thereby providing centralization in the enterprise. For example, you can deploy AquaLogic Service Bus as a single hub that manages all the Web service traffic in an enterprise.

You can horizontally scale an AquaLogic Service Bus hub by clustering the underlying WebLogic Server. This allows the messaging load to be homogeneously spread over the servers in the cluster, thereby preventing bottlenecks in the system. In addition, heterogeneous AquaLogic Service Bus hubs can be connected to create a distributed AquaLogic Service Bus network.

AquaLogic Service Bus supports clustering of the WebLogic managed servers. It propagates configuration and metadata automatically to the managed servers for fast local retrieval, and it automatically collects monitoring metrics from all the managed servers for aggregation and display on the AquaLogic Service Bus Console.

The following figure shows the flow of message data in a basic AquaLogic Service Bus cluster topology.

**Figure 1-2 Clustering and High Availability**



## Development, Staging, and Production Domains

AquaLogic Service Bus supports best practices for change management in your enterprise systems by enabling you to configure resources and services in a controlled environment. You can then export the configurations for import into separate staging domains for testing and final preparation for promotion into a production domain. AquaLogic Service Bus provides you with the ability to reconfigure environment-specific elements as necessary to meet the requirements of the importing domain.

## AquaLogic Service Bus Message Brokering

AquaLogic Service Bus provides intelligent message brokering functionality for your SOA. The following sections describe the key concepts associated with AquaLogic Service Bus message structures and message processing.

## Business Services and Proxy Services

AquaLogic Service Bus provides intelligent message brokering between business services (such as enterprise services and databases) and service clients (such as presentation applications or other business services) through proxy services that you configure using the AquaLogic Service Bus Console.

### Business Services

Business services are AquaLogic Service Bus definitions of the enterprise services with which you want to exchange messages. Using the AquaLogic Service Bus Console, you configure a business service by defining its interface in terms of WSDLs and the type of transport it uses, its security requirements, and other characteristics.

For information on how to configure a business service using the AquaLogic Service Bus Console, see “Adding a Business Service” in [Business Services](#) in the AquaLogic Service Bus Console Online Help.

### Proxy Services

Proxy services are AquaLogic Service Bus definitions of intermediary Web services that are hosted locally on AquaLogic Service Bus. Using the AquaLogic Service Bus Console, you configure a proxy service by defining its interface in terms of WSDLs and the type of transport it uses, the logic of message processing in message flow definitions, and by configuring policies.

With AquaLogic Service Bus message brokering, service clients exchange messages with an intermediary proxy service instead of directly with a business service. A proxy service can have an interface that is identical (same WSDL and transport) to a business service with which the proxy service communicates, or the proxy service can have an interface that differs from that of the business service in terms of WSDL, transport type, or both.

Since a proxy service can route messages to multiple business services, you can choose to configure a proxy service with an interface that is independent of the business services with which the proxy service communicates. In such cases, you would configure a message flow definition to route a message to the appropriate business service and map the message data into the format required by the business service’s interface.

For more information about the structure of proxy services, see “[Message Flow Definition](#)” on [page 1-12](#).

## Message Processing

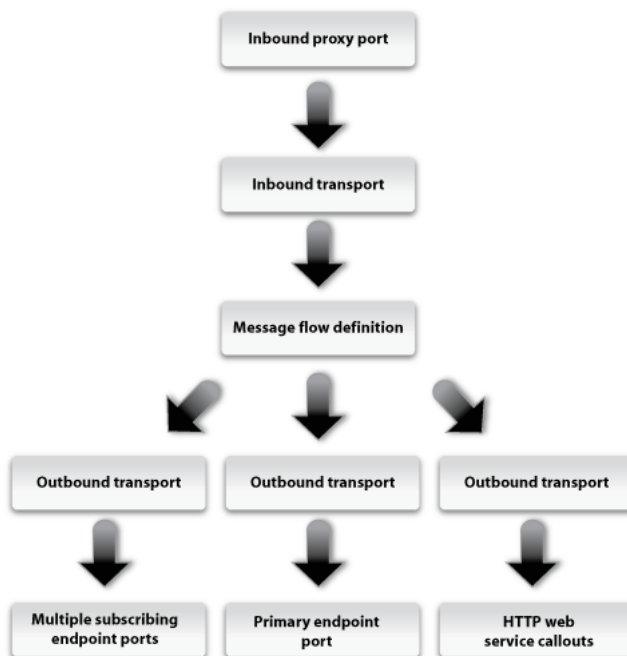
A *message* is a formula used by WebLogic Server to share information among applications. Messages can contain data or status information about application processes, or instructions for the recipient, or both. AquaLogic Service Bus enables you to route messages based on their contents and to perform transformations on that content.

The processing of messages occurs in the following sequence of events:

1. Processing of the inbound transport
2. Message flow execution
3. Processing of the outbound transport

After a message is sent to an endpoint (either a business service or another proxy service), AquaLogic Service Bus processes the response message in a similar model as that described in the preceding sequence of events.

The following figure shows a high-level view of how messages flow through AquaLogic Service Bus, from inbound endpoint (a proxy service) to outbound endpoint (the transport URL for a service—typically a business service).

**Figure 1-3 AquaLogic Service Bus Message Processing**

The following sections describe each layer involved in this message processing.

## Transport Layer (Inbound)

The first step for the message is the inbound transport layer, which is responsible for handling communication with the service client endpoint and acts as the entry point for messages into AquaLogic Service Bus. It supports a variety of communication protocols, such as HTTP and JMS. With regard to the message data itself, the inbound transport layer primarily deals with raw bytes in the form of input/output streams.

The inbound transport layer is simply for getting messages into the AquaLogic Service Bus system and sending the response back. It does not perform any data processing.

## Message Flow Definition

Pipelines, branches, and route nodes define the implementation of AquaLogic Service Bus proxy services. Using the AquaLogic Service Bus Console, you configure the logic for the manipulation of messages in proxy service message flow definitions. This logic includes such activities as



transformation, publishing, and reporting which are implemented as individual actions within the stages of a pipeline.

## Pipeline Pairs

Pipeline logic occurs in pairs of definitions consisting of a *request pipeline* definition and a *response pipeline* definition. The request pipeline definition specifies the actions that AquaLogic Service Bus performs on request messages to the proxy service before invoking a business service or another proxy service. The response pipeline definition specifies the processing that AquaLogic Service Bus performs on responses from the service invoked by the proxy service before the proxy service returns a response. Routing is performed by a route node at the end of the message flow.

**Note:** WS-Security processing and authorization are transparently performed before pipeline execution.

## Pipeline Execution Stages and Actions

Each pipeline is a sequence of stages. A stage is a user-configured processing step such as transformation or publishing. Messages fed into the pipelines are accompanied by a set of *message context variables* that contain the message contents and can be accessed or modified by actions in the pipeline stages.

The following table describes all of the actions possible in a AquaLogic Service Bus pipeline stage.

**Table 1-2 Actions in a Pipeline Stage**

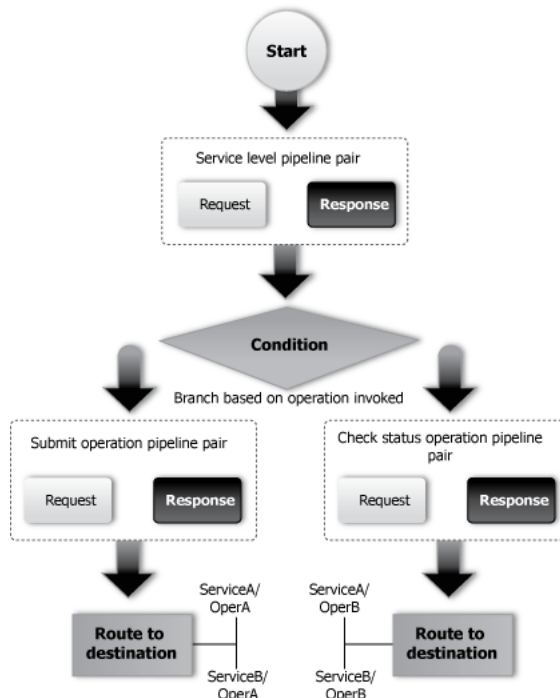
<b>Actions</b>	<b>What Happens</b>
<b>Transformation</b>	Transformation actions affect the message context. A Web services callout can be an alternative to an XQuery or XSLT transformation to set the output context variable. Inplace updates to transform a single context variable are also supported.
<b>Branching</b>	Branching actions allow the use of nested <code>if</code> structures to select a transformation.
<b>Publish</b>	A publish action enables you to define the set of endpoints to which the message is published. You use a publish action to send a copy of the message to a set of listeners during the message flow. You can define a set of XQuery or XSLT transformations that affects context variables before the message is published to each endpoint. Alternatively, you can specify that a Web services callout be used to set the context variable. <i>The changes to the predefined context variables in the publish request actions are isolated to that publish endpoint and do not affect subsequent processing by the message flow.</i>
<b>Reporting</b>	A reporting action enables writing of a reporting record with user-defined information from the context so that you can use the AquaLogic Service Bus Console to search and display this information.
<b>Jump</b>	Jump actions terminates processing in a stage. Typical use of these actions would be for error handling or to skip to the next stage in the pipeline.
<b>Logging</b>	A logging action enables logging of selected context to the WebLogic Server system log for debugging purposes.
<b>Validation</b>	A validation action validates a document against an XML or WSDL schema.

## Operational Pipelines

Each pipeline consists of a sequence of stages containing actions. However a single service level request pipeline might optionally branch out into operational pipelines (at most one per operation and optionally a default operational pipeline). The determination of the operation is done through user-selected criteria. The response processing starts with the relevant operation pipeline which then joins into a single service-level response pipeline.

The following figure shows an example of operation pipelines in a proxy service:

**Figure 1-4 Example Message Flow**



In the case of one-way operations, the response pipeline is executed with an empty message. This permits a response to be constructed for the proxy service, thus enabling bridging between request/response and one-way operations. The bridging mechanism means that proxy service input can be one-way while its output is request/response or *vice versa*. The proxy service either absorbs the response from the invoked service or generates one for the client.

## **Transport Layer (Outbound)**

The outbound transport layer is responsible for handling communication with the destination endpoint. It supports a variety of communication protocols, such as HTTP and JMS. With regard to the message data itself, the transport layer primarily deals with raw bytes in the form of input/output streams.

The outbound transport layer is simply for getting messages out of the AquaLogic Service Bus system. It does not perform any data processing.

# Resource and Service Configuration Concepts

AquaLogic Service Bus relies on user-configured metadata for resources and services to determine how to process messages. This topic is intended for IT specialists who are responsible for configuring a message broker in an SOA.

The section introduces the AquaLogic Service Bus resources and services, and provides links to more detailed information in the AquaLogic Service Bus documentation set.

This section includes the following sections:

- [Resources](#)
- [Services](#)

## Resources

AquaLogic Service Bus resources are reusable definitions or descriptions of entities that typically include metadata for those entities. Resources can be used by multiple services and provide standardized definitions or descriptions for use across an enterprise or department. Examples of AquaLogic Service Bus resources include:

- MFL schemas
- XML schemas
- XQuerys
- XSLTs

- WSDL interfaces
- Security policies (WS-Policy)

You can organize the resources and services in AquaLogic Service Bus into a set of *projects*, each with a hierarchy of folders. Organizing resources and services into projects not only avoids name conflicts, but also provides a convenient way to organize resources and services by department or other business category and search for them.

AquaLogic Service Bus also provides *sessions* to allow you to make a series of changes over a long period, and keep them private until you are ready to submit the changes as a batch of updates for deployment. Multiple sessions can be active. An optimistic locking approach is used, so it is possible that conflicts can occur which have to be resolved before a successful submit.

This section includes the following topics:

- [Schemas and Data Types](#)
- [Transformation Maps](#)
- [WSDLs](#)
- [WS-Policy](#)

## Schemas and Data Types

Schemas describe types for primitive or structured data. XML Schemas are an XML vocabulary that describe the rules that XML business data must follow. XML Schemas specify the structure of documents, and the data type of each element and attribute contained in the document. XML schemas can import or include other XML schemas. For information on how to create schemas using the AquaLogic Service Bus Console, see “Adding a New Schema” in [Resource Browser](#) in the *AquaLogic Service Bus Console Online Help*.

BEA WebLogic Integration uses a metadata language called Message Format Language (MFL) to describe the structure of typed non-XML data. The BEA Format Builder tool creates and maintains metadata as a data file called an MFL document. For information on how to create MFL documents, see the [Format Builder Online Help](#).

## Transformation Maps

Transformation maps describe the mapping between two data types. AquaLogic Service Bus supports data mapping using either XQuery or the eXtensible Stylesheet Language Transformation (XSLT) standard. In addition, MFL described data is automatically converted to

the equivalent XML for transformation with XQuery or XSLT. The resulting XML is automatically converted to MFL if the target service requires it.

For information on how to configure AquaLogic Service Bus transformation maps, see [Transforming Data Using the XQuery Mapper](#).

## WSDLs

A WSDL (Web Service Definition Language) interface defines a service interface for a SOAP or XML service. It describes the abstract interface of a service including the operations in that interface and the types of message parts in the operation signature. It can also describe the binding of the message parts to the message (packaging), and the binding of the message to the transport. In addition a WSDL can describe the concrete interface of the service (for example, the transport URL).

For information on how to configure WSDLs using the AquaLogic Service Bus Console, see “Adding a New WSDL” in [Resource Browser](#) in the *AquaLogic Service Bus Console Online Help*.

## WS-Policy

A WS-Policy describes security policy. It describes what should be signed or encrypted in a message and the security algorithms to be applied. A WS-Policy also describes what authentication mechanism should be used for the message when received. The policy could also optionally be embedded in a WSDL.

For more information on WS-Policy, see “Web Service Policies” in [Securing Inbound and Outbound Messages](#) in the *BEA AquaLogic Service Bus User Guide*.

## Services

The fundamental concept in AquaLogic Service Bus is that both proxy services and business services invoked by the proxy services are modeled as services.

Services have the following attributes:

- A set of concrete interfaces called *ports* (also called an *endpoint*), each with a transport address and associated configuration. A set of ports constitutes load balancing and failover alternatives for a business service. A proxy service has only a single port.

- A single optional abstract interface equivalent to a Java interface, which is the definition of the structure of message parts in the interface, optionally broken down by operations. Operations are equivalent to methods of a Java interface.
- A single binding that defines the packaging of message parts in the abstract interface to a concrete message.
- Policies on Web Service Security (WS-Security).

## Service Types

AquaLogic Service Bus supports SOAP and XML-based Web services, messaging services, and services that can be specified with an abstract definition. It also supports SOAP and XML services for which only the concrete interface and the service type are defined.

AquaLogic Service Bus supports the following categories of these three service types:

- WSDL Service
  - SOAP
  - XML
- Messaging Service
  - Text
  - MFL
  - XML
  - Binary
- Services having a concrete interface and service type defined, but no abstract interface
  - SOAP
  - XML

AquaLogic Service Bus supports request and response as well as one-way paradigms for both the HTTP and the JMS asynchronous transport protocols. If the underlying transport supports ordered delivery of messages, AquaLogic Service Bus can also support it.

## Service Transports

AquaLogic Service Bus supports the following transport protocols:



- JMS (for BEA and external JMS providers)
- HTTP(S)
- E-mail
- File
- FTP

For information on how to configure transport for a proxy service using the AquaLogic Service Bus Console, see “Adding a Proxy Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. For information on how to configure transport for a business service using the AquaLogic Service Bus Console, see “Adding a Business Service” in [Business Services](#) in the *AquaLogic Service Bus Console Online Help*.

## Service Interfaces

AquaLogic Service Bus relies on WSDLs for the formal description of services. For Web services, a WSDL describes what the Web service’s interface is, where it resides, and how to invoke it. AquaLogic Service Bus defines proxy services and business services in terms of two WSDL entities:

- The abstract WSDL interface defines the operations in that interface and the types of message parts in the operation signature
- The binding WSDL interface defines the binding of the message parts to the message (packaging), and the binding of the message to the transport

Using the AquaLogic Service Bus Console, you import the WSDLs into the WSDL repository. You also use the AquaLogic Service Bus Console to resolve the references in the WSDLs, which ensures all schemas and WSDLs are linked correctly. Once you have stored WSDLs in the repository, they are available for you to use when adding proxy services and business services. For messaging services, AquaLogic Service Bus uses its own representation of the interface.

For information on how to import and resolve WSDLs using the AquaLogic Service Bus Console, see “Adding a New WSDL” in [Resource Browser](#) in the *AquaLogic Service Bus Console Online Help*.

## Proxy Services and Proxy Service Providers

Proxy services are AquaLogic Service Bus definitions of intermediary Web services that are hosted locally on AquaLogic Service Bus. You define a proxy service in terms of WSDLs,

message flows, and policies. If the proxy service requires credential-level validation, you can create a *proxy service provider* to manage security credentials for the proxy service from the AquaLogic Service Bus Console.

For information on how to configure a proxy service using the AquaLogic Service Bus Console, see “Adding a Proxy Service” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*. For information on how to configure a proxy service provider using the *AquaLogic Service Bus Console*, see “Adding a Proxy Service Provider” in [Proxy Service Providers](#) in the *AquaLogic Service Bus Console Online Help*.

The following sections describe key concepts regarding the structure and definition of proxy services.

## Message Context

The heart of the proxy service is the *context*. The context is a set of XML variables that is shared across the request flow and response flow. You can add new variables to the context dynamically, and delete them. The predefined context variables contain information about the message, the transport headers, security principals, the metadata for the current proxy service and the metadata for the primary routing and publish services invoked by the proxy service. The context can be read and modified by XQuery expressions and updated by transformation and in-place update actions.

The core of the context are the variables `$header`, `$body`, and `$attachments`. These are wrapper variables that contain the SOAP header elements, the SOAP body element, and the MIME attachments, respectively. The context gives the impression that all messages are SOAP messages and non-SOAP messages are mapped to this paradigm. The following table lists the mappings for each message type.

**Table 2-1 Message mappings**

Message Type	What Happens
XML	The <code>Body</code> element in <code>\$body</code> contains the XML document. Attachments are in <code>\$attachments</code> .
binary	The <code>Body</code> element in <code>\$body</code> contains a reference XML document.
MFL	The document is transparently converted from and to XML, and appears as an XML document in the <code>Body</code> element in <code>\$body</code> .
text	The <code>Body</code> element in <code>\$body</code> contains the text.

**Table 2-1 Message mappings**

<b>file, FTP, and email</b>	In the case of pass-by-reference documents, a reference XML document in the <code>Body</code> element in <code>\$body</code> refers to the URI of the document stored in the file system by the transport.  Attachments are in <code>\$attachments</code> .
<b>SOAP</b>	The <code>Body</code> element in <code>\$body</code> contains the SOAP body. The <code>Header</code> element in <code>\$header</code> contains the SOAP header. Attachments are in <code>\$attachments</code> .

In the case of attachments, `$attachments` contains the following:

- attachments, if the attachment is XML
- a reference XML, if the attachment is binary
- text, if the attachment is text

## Message Flow Configuration

The implementation of a proxy service is specified by a *message flow* definition. The message flow defines the flow of messages through the proxy service. Four elements are used to construct a message flow. These elements are:

- A *pipeline pair*, one for the request and one for the response. The pipelines consist of a sequence of stages that specify actions to perform during request or response processing.
- A *branch node* to branch based on the values in designated parts of the message or message context or to branch based on the operation invoked.
- A *route node* used to define the message destination. The default route node is an *echo node* that reflects the request as the response.
- A *start node*.

These elements can be combined in arbitrary ways to form a tree structure with the start node always (and only) occurring as the root of the tree and the route nodes only allowed at the leaves. The request message starts at the start node and follows a path to a leaf executing actions in the request pipelines. If the leaf is a route node a response is generated (could be empty if the service is a one way service). If the leaf is an echo node, the request is also considered to be the response. The response follows the inverse path in the tree skipping actions in the branch nodes, but executing actions in response pipelines.

A route node is very flexible in its specification. It allows `if` structures and `case` structures to be combined (and nested) to define a single endpoint and operation to route the message. For information about how to configure route nodes, see “Adding a Route Node in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

A set of transformations that affects context variables can be defined before the message is sent to the selected endpoint or after the response is received. A Web services callout can be an alternative to an XQuery or XSLT transformation to set the context variable. For information on how to configure AquaLogic Service Bus transformation maps, see the *[Transforming Data Using the XQuery Mapper](#)*.

WS-Security processing as well as authorization is transparently performed before the message flow is invoked. WS-Security processing is transparently performed when invoking an business service with a ws-policy. For information on how to configure AquaLogic Service Bus security, see [Securing Inbound and Outbound Messages](#) in the *BEA AquaLogic Service Bus User Guide*.

For more information about pipeline pairs, stages, and actions, see “[Message Flow Definition](#)” on page 1-12.

For an example of a typical message flow diagram, see [Figure 1-4](#) in “[Operational Pipelines](#)” on page 1-15.

## Type System

AquaLogic Service Bus has a built-in type system that is available for use at design time. When creating an XQuery expression in a condition, in-place update action, or transformation, the variable can be declared to be of a given type in an editor to assist in easily creating the XQuery. The types can be the following:

- XML schema types or elements
- WSDL types or elements
- MFL types

## Error Handling

Each stage can have a sequence of steps to execute if an error occurs in that stage. This sequence of steps constitute an error pipeline for that stage. In addition, an error pipeline can be defined for a pipeline (request or response) or a whole proxy service. The lowest scoped error pipeline that exists is invoked on an error. This error pipeline allows you to handle the error in the following ways:

- Publish the original message to an alternate endpoint
- Formulate an error response message to be returned to the invoker of the proxy service
- Log the message
- Continue processing the message through the pipeline after modifying the context
- Raise an exception. Raising an exception transfers control to the next higher scoped error pipeline.

## Business Services and Service Accounts

Business services are AquaLogic Service Bus definitions of the enterprise services with which you want to exchange messages. A business service definition is similar to that of a proxy service, but it does not have a pipeline. If AquaLogic Service Bus needs to provide authentication when connecting to a business service, you can use the AquaLogic Service Bus Console to create a *service account* that acts as an alias resource for the required username and password pair. For more information, see [Business Services](#) and [Service Accounts](#) in the *AquaLogic Service Bus Console Online Help*.

If a business service requires credential-level validation, you must use WebLogic Server directly to manage its security credentials. For more information on business service security considerations, see [Securing Inbound and Outbound Messages](#) in the *BEA AquaLogic Service Bus User Guide*.

## Resource and Service Configuration Concepts

# System Administration and Operation Monitoring Concepts

This topic introduces AquaLogic Service Bus system administration and operation monitoring concepts. It is intended for system administrators and operators who manage and monitor AquaLogic Service Bus. It includes the following sections:

- [Security Management](#)
- [Configuration Metadata Export and Import](#)
- [Dashboard, System Metrics, and Alerts](#)
- [Message Reporting](#)

## Security Management

AquaLogic Service Bus leverages the WebLogic Server security architecture and services to support secure message exchanges between services and the clients of those services. WebLogic Server supplies implementations of the following types of security features:

- Authentication
- Identity assertion
- Authorization
- Role mapping
- Auditing
- Credential mapping

For detailed descriptions of these WebLogic Server security providers and WebLogic Server security architecture in general, see the [BEA WebLogic Server Security](#) documentation.

AquaLogic Service Bus security supports the WS-Policy specification. For more information about the WS-Policy specification, see the Web Services Policy Framework (WS-Policy) and Web Services Policy Attachment (WS-PolicyAttachment) which is available at:

<http://specs.xmlsoap.org/ws/2004/09/policy/>

WS-Policy describes what should be signed or encrypted in a message and what security algorithms should be applied. It also describes the authentication mechanism that should be used for the message when the message is received.

Using the AquaLogic Service Bus Console, you can configure a service with security policies that apply to messages in its interface. You can specify a security policy for a service or for individual messages associated with the operations of a service. When you specify a security policy for a service, the policy applies to all messages to that service.

The AquaLogic Service Bus Console also enables you to manage the credentials required for proxy service transport-level and message-level security. You manage business service and proxy service client credentials directly using WebLogic Server.

AquaLogic Service Bus enables you to use the WebLogic Server security providers at several different levels in its operation. The following sections provide introductions to the security available at each level:

- [User Management](#)
- [Console Security](#)
- [Transport-Level Security](#)
- [Message-Level Security](#)

## User Management

AquaLogic Service Bus user management is built on the unified WebLogic Server security framework. This framework enables the AquaLogic Service Bus Console to support task-level authorization based on security policies associated with roles assigned to named groups or individual users. For more information on the WebLogic Server security framework, see the [BEA WebLogic Server Security](#) documentation.

You use the AquaLogic Service Bus Console to manage AquaLogic Service Bus users, groups, and roles. For information on how to manage AquaLogic Service Bus users, groups, and roles



using the AquaLogic Service Bus Console, see [Security Configuration](#) in the *AquaLogic Service Bus Console Online Help*.

## Console Security

By default, the first user created for an AquaLogic Service Bus domain is a WebLogic Server Administrator. This user has full access to all AquaLogic Service Bus objects and functions, and can execute user management tasks to provide controlled access to AquaLogic Service Bus Console functionality. The following table shows the default roles and groups to which you can assign AquaLogic Service Bus users.

**Table 3-1 AquaLogic Service Bus Default Roles and Groups**

Role	Associated Group	Description
IntegrationAdmin	IntegrationAdministrators	Has complete access to all AquaLogic Service Bus objects and functions.
IntegrationDeployer	IntegrationDeployers	Has read access to all objects. Can create, delete, edit, import, or export resources, services, proxy service providers, or projects.
IntegrationOperator	IntegrationOperators	Has read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.
IntegrationMonitor	IntegrationMonitors	Has read access to all objects. Can export any resource, service, proxy service provider, or project.

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

## Transport-Level Security

AquaLogic Service Bus supports transport-level confidentiality, message integrity, and client authentication for one-way requests or request/response transactions (from clients to WebLogic Service Bus) over HTTPS. You can configure HTTP(S) proxy services or business services to require one of the following types of client authentication:

- BASIC (username/password) client authentication
- CLIENT CERT (two-way SSL) client authentication
- No client authentication.

When a proxy service is activated, AquaLogic Service Bus generates and deploys a thin Web application. 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/certificate manipulation.

Transport security for transports other than HTTP is supported in AquaLogic Service Bus as follows:

- For email and FTP, security is provided using a credential to connect to a FTP or email server.
- For file transport, security is provided using a login control to the machine on which the files are located.

For more information, see “Transport-Level Security” in [Securing Inbound and Outbound Messages](#) in the *BEA AquaLogic Service Bus User Guide*.

## Message-Level Security

AquaLogic Service Bus supports OASIS Web Services Security (WSS) 1.0. For more information on the WSS specification, see the OASIS Web Services Security TC which is available at:

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)

WSS defines a framework for message confidentiality, integrity, and sender authentication for SOAP messages.

Using WSS, AquaLogic Service Bus provides support for securing messages using digital signatures, encryption, or both. While not a substitute for transport-level security, WSS is ideal for end-to-end message confidentiality and integrity. It is more flexible than SSL since individual parts of the SOAP envelope can be signed, encrypted or both, while other parts are neither signed nor encrypted. This is a powerful feature when combined with the ability of WebLogic Service Bus to make routing decisions and perform transformations on the data based on the message content.

WebLogic Service Bus currently supports WSS over HTTP/S and JMS.

For more information, see “Message-Level Security” in [Securing Inbound and Outbound Messages](#) in the *BEA AquaLogic Service Bus User Guide*.

## Configuration Metadata Export and Import

AquaLogic Service Bus Console enables you to save your AquaLogic Service Bus resource configurations as metadata and export them in JAR files for import into other AquaLogic Service Bus domains. You can customize configuration settings as necessary to meet the requirements of the new environment before deploying the configuration using the Change Center in the AquaLogic Service Bus Console. This functionality supports an orderly promotion process of AquaLogic Service Bus resource configurations from staging and test environments into production.

Using the features of your source code control system in conjunction with the configuration JAR files, you can provide version and change management for your AquaLogic Service Bus configurations.

For information on how to export and import configuration metadata using the AquaLogic Service Bus Console, see [System Administration](#) in the *AquaLogic Service Bus Console Online Help*. For information on how to modify configurations for new environments using the AquaLogic Service Bus Console Change Center, see [Using the Change Center](#) in the *AquaLogic Service Bus Console Online Help*.

## Dashboard, System Metrics, and Alerts

AquaLogic Service Bus aggregates runtime statistics that you can view in a customizable dashboard to monitor system health. You can also use the AquaLogic Service Bus Console to establish service level agreements (SLAs) for the performance of your system, and configure rules that trigger alerts to provide automated responses to SLA violations.

### Dashboard

The AquaLogic Service Bus Console Dashboard displays information about system health organized by server and services. You can drill down from summary pages to detailed information about individual servers, services, and alerts.

The Dashboard shows status information for a period of time that you can configure to meet your monitoring requirements. The following table lists the metrics that the Dashboard displays for each service.

**Table 3-2 AquaLogic Service Bus Service Metrics**

Metric	Description
Average Execution Time	<p>For a proxy service, the average of the time interval measured between receiving the message at the transport and either handling the exception or sending the response.</p> <p>For a business service, the average of the time interval measured between sending the message in the outbound transport and receiving an exception or a response.</p>
Total Number of Messages	<p>Number of messages sent to the service. In the case of JMS proxy services, if the transaction aborts due to an exception and places the message back in the queue so it is not lost, each retry dequeue is counted as a separate message. In the case of outbound transactions, each retry or failover is likewise counted as a separate message.</p>
Messages With Errors	<p>Number of messages with error responses.</p> <p>For a proxy service, it is the number of messages that resulted in an exit with the system error handler or an exit with a reply failure action. If the error is handled in the service itself with a reply with success or a resume action, it is not an error.</p> <p>For a business service, it is the number of messages that resulted in a transport error or a timeout. Retries and failovers are treated as separate messages.</p>
Success/Failure Ratio	<p><math>(\text{Total Number of Messages} - \text{Messages with Errors}) / \text{Messages with Errors}</math></p>
Security	<p>Number of messages with WS-Security errors. This metric is calculated for both proxy services and business services.</p>
Validation	<p>Number of validation actions in the flow that failed. This metric only applies to proxy services.</p>

These metrics are aggregated across the cluster for the configured aggregation interval. The Dashboard displays information about the overall health of the system, refreshing the display at a specified interval.

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

## Metric Aggregation

The information displayed on the Dashboard is based on an asynchronous aggregation of data collected during system operation. In an AquaLogic Service Bus production cluster domain, the AquaLogic Service Bus data aggregator runs as a singleton service on one of the managed servers in the cluster. Server-specific data aggregation is performed on each of the managed servers in the domain. The aggregator is responsible for the collection and aggregation of data from all the managed servers at regular, configurable intervals.

## Alerts

AquaLogic Service Bus implements Service Level Agreements (SLAs) and automated responses to SLA violations by enabling you to define *Rules* that specify unacceptable service performance and the system response you require under those circumstances. You construct Rules using the AquaLogic Service Bus Console. AquaLogic Service Bus evaluates Rules against its aggregated metrics each time it updates that data.

When a Rule evaluates to `True`, it raises an *alert*. In addition to displaying information about the alert in the AquaLogic Service Bus Console Dashboard, AquaLogic Service Bus executes the action you specified for the Rule when it evaluates to `True`. You can assign any of the following types of action to a Rule:

- Send email notification
- Send a JMS message
- Invoke a Web service
- Send alert to the WebLogic Server Logger

It is also possible to configure specific operating times for alerts. For example, you can configure alerts to operate only during normal business hours.

Rule and alert processing is handled by the AquaLogic Service Bus Alert Manager. The Alert Manager resides on the same single managed server as the metric aggregator for the system.

For information on how to configure AquaLogic Service Bus SLAs, see “Alerts Rules” in [Monitoring](#) in the *BEA AquaLogic Service Bus User Guide*.

## Message Reporting

When you configure a proxy service, you can include a reporting action in its message flow. In the reporting action, you specify the information about each message that you want to have written to the AquaLogic Service Bus Reporting Data Stream. The JMS Reporting Provider picks up this data and stores it in a message reporting database that acts as the Reporting Data Store. For information on how to configure reporting actions, see “Adding an Action” in [Proxy Services](#) in the *AquaLogic Service Bus Console Online Help*.

AquaLogic Service Bus Console Message Reporting displays information from the Reporting Data Store. Message Reporting enables you to drill down from summary information to view detailed information about specific messages. You can customize the display of Message Reporting information by filtering and sorting the data to meet your reporting requirements.

**Note:** Message Reporting displays information only for messages that traverse a pipeline that includes a reporting action.

AquaLogic Service Bus Console provides purge functionality to help you manage your message data. For other data management functions, you should apply standard database administration practices to the database hosting the Reporting Data Store.

For a list of supported database platforms for the Reporting Data Store, see [Supported Database Configurations](#) in *Supported Configurations for AquaLogic Service Bus*.