BEA AquaLogic Service Bus™

Concepts and Architecture

Version: 2.5
Document Revised: July 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

1. Overview

2. Resource and Service Configuration Concepts

3. System Administration and Operation Monitoring Concepts

4. Change Management and Resource Organization

CHAPTER 1

Overview

This topic introduces AquaLogic Service Bus. It is intended for enterprise architects, operations specialists, and security architects who are responsible for messaging and service oriented architectures (SOA). It includes the following sections:

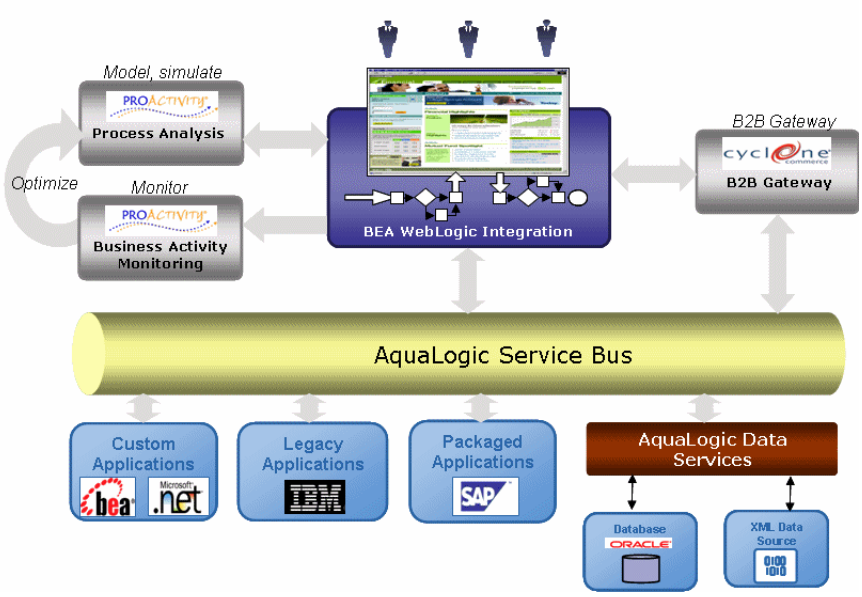
- Introducing AquaLogic Service Bus
- AquaLogic Service Bus Architecture
- AquaLogic Service Bus Core Feature Set
- AquaLogic Service Bus Message Brokering

Introducing AquaLogic Service Bus

As businesses strive to be agile, they depend on IT's ability to deliver new services and reuse existing services at the speed of business. This desire to be service-driven is creating a significant pull for IT to adopt Service-Oriented Architecture (SOA). To make SOA a reality, IT requires intelligent service infrastructure that drives and simplifies service reuse and delivers reliable integration across an inherently heterogeneous, multi-vendor computing landscape. BEA AquaLogic Service Bus—part of the BEA AquaLogic family of Service Infrastructure Products—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)

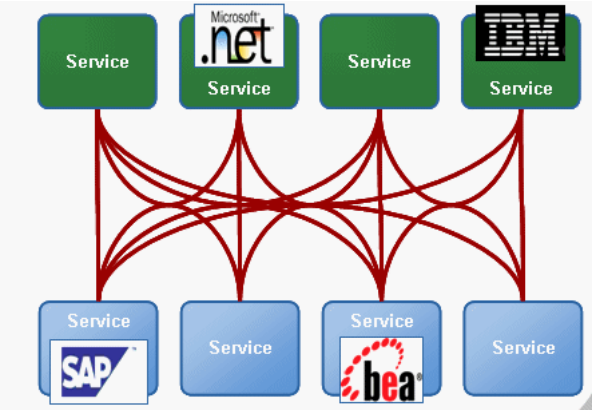
enforcement. AquaLogic Service Bus is at the heart of BEA's comprehensive business integration solution.

Figure 1-1 BEA's Comprehensive Integration Solution



AquaLogic Service Bus eliminates the service sprawl that challenges enterprises and their IT departments. The challenges include hard-wired connections between applications and services, resulting in complex, rigid, and tightly-coupled integration. In turn, this results in an impaired ability to reuse services and challenges in managing deployed services. All this results in a high total cost of ownership for the enterprise.

Figure 1-2 The Service Sprawl Challenge



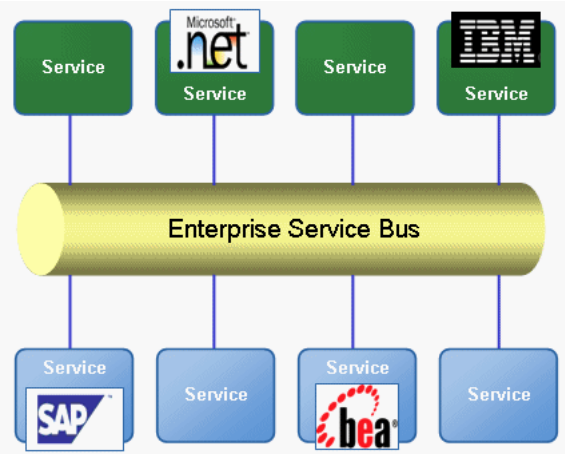
The specific challenges faced by enterprise architects responsible for messaging and SOA initiatives in today's enterprise include:

- 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 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. AquaLogic Service Bus facilitates a loosely coupled architecture, facilitates enterprise-wide reuse of services, and centralizes management—all of which results in an improved total cost of ownership. The AquaLogic Service Bus Console enables you to respond rapidly and effectively to changes in your service-oriented environment.

Figure 1-3 Eliminating Service Sprawl



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.

AquaLogic Service Bus is an ESB product specifically targeted for service-oriented integration, managing primarily Web Services, and providing traditional message brokering across heterogeneous IT environments.The lightweight, stateless, high-performance architecture of AquaLogic Service Bus 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.

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

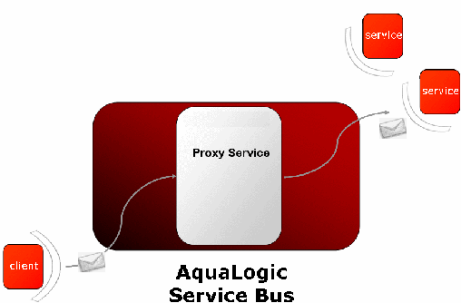
AquaLogic Service Bus persists policy, proxy service, and related resource configurations in metadata that you can propagate from development through staging to production environments, and then modify as required. The AquaLogic Service Bus message brokering engine accesses this configuration information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

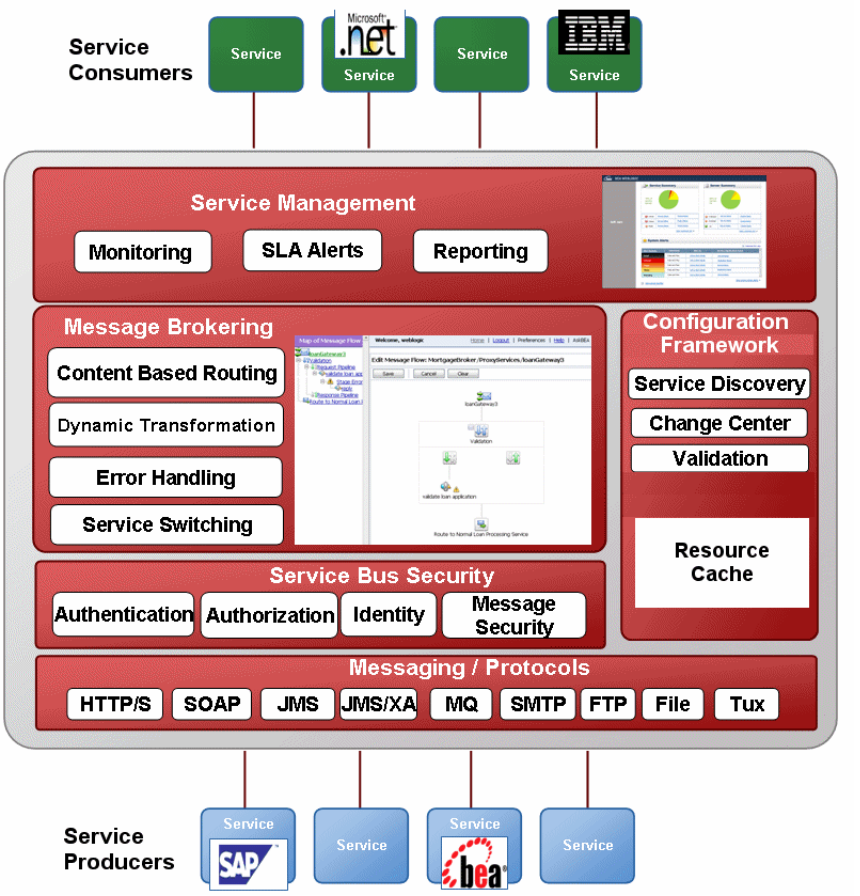
AquaLogic Service Bus is an intermediary that takes in messages, processes them to determine where to route them, and transforms them as specified. It receives messages through a transport protocol such as HTTP(S), JMS, File, FTP, and so on, and sends messages through the same or a different transport 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.

Figure 1-4 Service Consumers and Service Producers Interact with AquaLogic Service Bus



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. The following figure shows the high-level architecture, which shows AquaLogic Service Bus composed of the following subsystems: service management, message brokering, configuration framework, security, and transport and messaging protocols.

Figure 1-5 AquaLogic Service Bus High-Level Architecture



Deployment Topology

AquaLogic Service Bus runs on WebLogic Server version 9.0 or greater, 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.

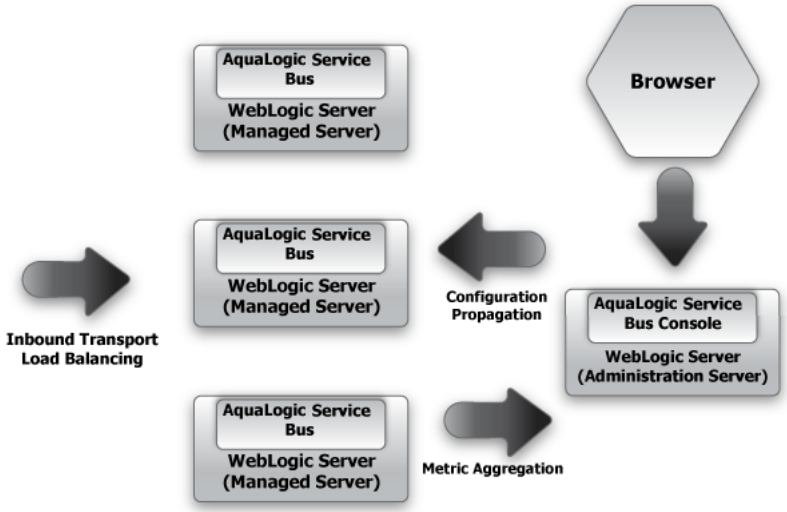
AquaLogic Service Bus can manage and control many distributed service endpoints, thereby providing centralization in the enterprise. For example, you can deploy AquaLogic Service Bus as a single hub that manages all the Web service traffic in an enterprise.

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 1-6 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. To learn about managing changes and organizing resources in AquaLogic Service Bus, see Chapter 4, "Change Management and Resource Organization."

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.

Table 1-1 describes the core features of AquaLogic Service Bus.

Table 1-1 AquaLogic Service Bus Core Features

For this feature...	AquaLogic Service Bus...
Routing	Supports the following: • Routes messages according to XQuery-based policies or callouts to external services • Routing policies apply to both point-to-point and one-to-many routing scenarios (publish). For publish, routing policies serve as subscription filters
Routing Transport Protocols	Supports the following: • File • FTP • HTTP(S) • JMS (including MQ using JMS, and JMS/XA) • E-mail (POP/SMTP/IMAP)
Messaging	Supports the following models: • Synchronous • Asynchronous • Publish • Subscribe
Message Types	Supports the following message formats: • E-mail with or without attachments • JMS with headers • MFL (Message Format Language) • Raw Data. Raw data is opaque data—that is, non-XML data for which there is no MFL file and therefore no known schema • Text • SOAP and SOAP with attachments (SOAP that is or is not described by a WSDL) • XML and XML with attachments ((XML that is or is not described by a WSDL or a schema)

Table 1-1 AquaLogic Service Bus Core Features

For this feature...	AquaLogic Service Bus...
Transformations	Supports the following functionality for the transformation or processing of messages: • Validates incoming messages against schemas • Selects a target service or services, based on the message content or message headers • Transforms messages based on the target service • Transforms messages based on XQuery or XSLT • Supports transformations on both XML and MFL messages • Message enrichment • Supports call outs to Web services to gather additional data for transformation (for example, country code, full customer records, and so on)
Testing	Provides a built-in test console in the development environment: • Allows you to test resources and inline XQuery expressions used in the message flow • Allows you to test business services and proxy services • Provides tracing of the message flow when you test a service using the test console
Logging and Monitoring	Provides a rich set of functionality to audit and monitor services: • You can gather statistics about message invocations, errors, performance characteristics, messages passed, SLA violations, and so on • The system also supports logging selected parts of messages for both systems operations and business auditing purposes, search capabilities, and so on. You can extract key information from a message and use as it as a search index. • The AquaLogic Service Bus Console provides a cluster-wide view of service status and statistics • Both business services and AquaLogic Service Bus proxy services are monitored, as are response times, message counts, and error counts • Statistics are gathered locally, then aggregated centrally • SLA rules run against aggregated data. The system raises alerts, and you can enable or disable services.

Table 1-1 AquaLogic Service Bus Core Features

For this feature...	AquaLogic Service Bus...
Versioning	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.
Service Level Agreements	Administrators can set service level agreements (SLAs) on the following attributes of proxy services: • Average processing time of a service • Processing volume • Number of errors, security violations, and schema validation errors • Administrators can configure alerts for SLA rule violations
Security	Includes the following: • Supports authentication, encryption and decryption, and digital signatures as defined in the Web Services Security (WS-Security) specification • Uses SSL to support traditional transport-level security for HTTP and JMS transport protocols • Supports one-way and two-way certificate based authentication • Supports HTTP basic authentication
Resource Cache	Supports the following: • Stores information about services, schemas, transformations, WSDLs (Web Service Definition Language), and WS Policies • Provides centralized management and distributed access to resources and services • Allows you to browse the services registered in AquaLogic Service Bus and import resources from WebLogic Workshop or other applications • 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. • Interoperability with UDDI registries—you can publish a proxy service to and import a business service from version 3-compliant UDDI registries. For more information about UDDI registries, see "UDDI" on page 3-10.

Table 1-1 AquaLogic Service Bus Core Features

For this feature...	AquaLogic Service Bus...
Error Handling	Supports the following: • Allows you to configure your system to format and send error messages, and return messages for consumers of services who expect a synchronous response • Allows you to configure error handling for stages in the pipeline, for pipelines, and for proxy services

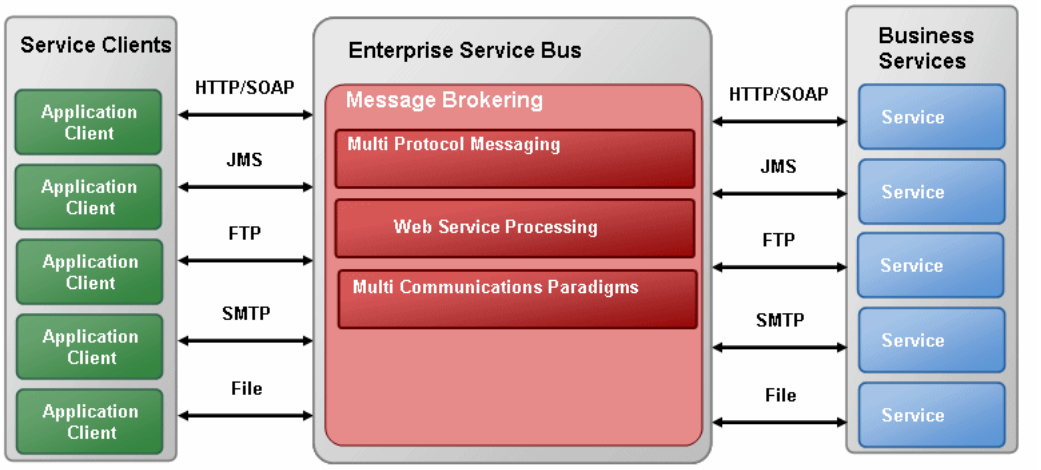
AquaLogic Service Bus Message Brokering

AquaLogic Service Bus provides intelligent message brokering functionality for your SOA.

AquaLogic Service Bus supports multiple messaging protocols: HTTP(S), JMS SAF, third party messaging using JMS provider interfaces (MQ Series, Tibco E4JMS), File, FTP, Email (SMTP/POP/IMAP). Multiple transports are supported—end-to-end guaranteed delivery is possible when the transport supports it. You can create a messaging network with JMS Store and Forward and Global Queue Names. Web Services (WSDL, SOAP enveloping) and non-SOAP-enveloped messages are supported.

AquaLogic Service Bus supports multiple communications paradigms including request/response, asynchronous/synchronous, publication/subscription, Web Services with attachments, and so on. A mix and match of transports is possible (for example, sync-to-async bridging is supported).

Figure 1-7 Message Brokering in AquaLogic Service Bus



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 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 *Using the AquaLogic Service Bus Console.*

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 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 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.

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.

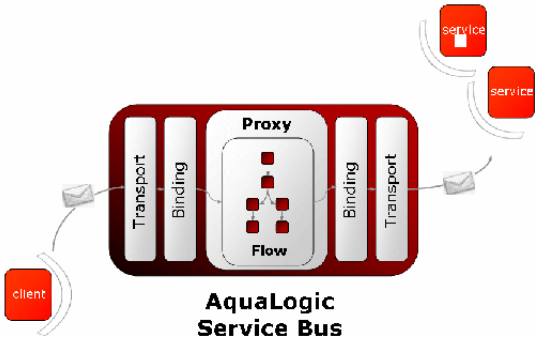
For more information about the structure of proxy services, see "Message Flow Definition" on page 1-18.

Message Processing

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 happens through the transport and binding layers of AquaLogic Service Bus.

Figure 1-8 Binding and Transport Layers in AquaLogic Service Bus



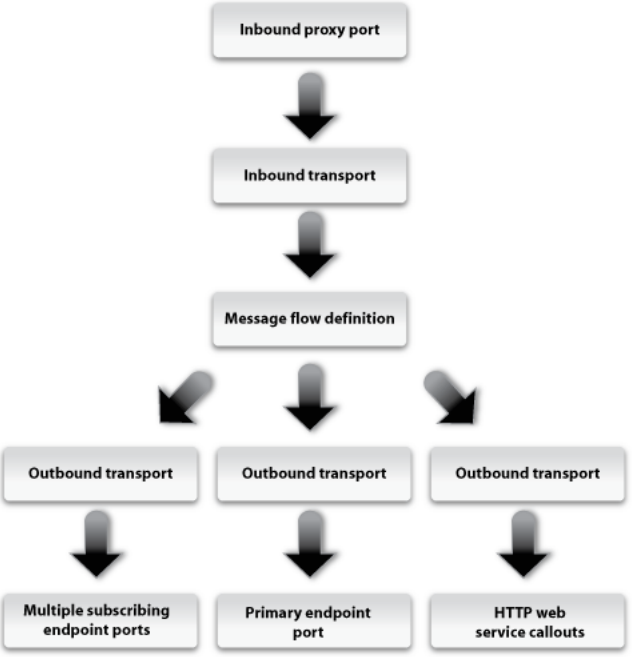
The processing of messages occurs in the following sequence of events:

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—a business service or another proxy service).

Figure 1-9 AquaLogic Service Bus Message Processing



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

Binding Layer

The binding layer:

- Packs & Unpacks Messages as necessary
- Handles security for messages
- Hands messages off to start the message flows (request and response)

Transport Layer (Inbound)

The inbound transport layer:

- Is responsible for the communication between client services (or service consumers) and AquaLogic Service Bus. 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. With regard to the message

data, the inbound transport layer primarily deals with raw bytes in the form of input/output streams.

- The inbound transport layer is responsible for getting messages into the AquaLogic Service Bus system and sending the response back. It does not perform any data processing.
- Provides support for the transport protocols, including HTTP(S), JMS, FTP, File, Email, and so on.
- Handles meta-data for messages, including endpoint URI's, transport headers, and so on.

Transport Layer (Outbound)

The outbound transport layer:

- Is responsible for the communication between business services (or service producers) and AquaLogic Service Bus. It is responsible for moving messages from AquaLogic Service Bus to the business service or proxy service and for receiving the response from the services back to AquaLogic Service Bus. It is not involved in data processing. The message data, at the transport level, is in raw bytes in the form of input/output streams.
- Provides support for the transport protocols, including HTTP(S), JMS, FTP, File, Email, and so on.
- Handles meta-data for messages, including endpoint URI's, transport headers, and so on.

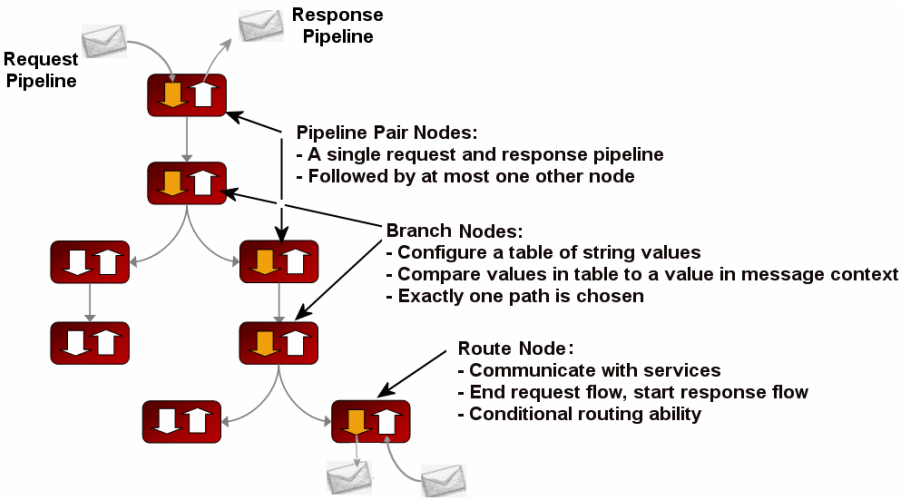
Message Flow Definition

Message Flows are the definitions of AquaLogic Service Bus proxy services. Pipelines, branch nodes, and route nodes define the implementation of AquaLogic Service Bus proxy services.

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—the logic is configured in individual actions within the message flow.

The following figure shows a high level view of the components of the message flow definition.

Figure 1-10 Components of Message Flows



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 and after the 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 message flow 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 the actions supported in a AquaLogic Service Bus pipeline stages, branch and route nodes.

Table 1-2 AquaLogic Service Bus Actions

Action[1]	Summary Description
Assign	Assign the result of an XQuery expression to a context variable
Delete	Delete a context variable or all the nodes specified by an XPath expression
For Each	Iterate over a sequence of values and execute a block of actions
If Then	Perform an action or set of actions conditionally, based on the Boolean result of an XQuery expression
Insert	Insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression
Log	Construct a message to be logged
Publish	Specify a target service for a message and configure how the message is packaged and sent to that service
Publish Table	Specify a target service for a message and configure how the message is packaged and sent to that service. Use the Publish Table to wrap a set of routing options in switch-style condition logic
Raise Error	Raise an exception with a specified error code and description
Rename	Rename elements selected by an XPath expression without modifying the contents of the element
Replace	Replace a node or the contents of a node specified by an XPath expression
Reply	Specify that an immediate reply is sent to the invoker; can be a reply with success or failure
Report	Enable message reporting for a proxy service
Service Callout	Configure a synchronous (blocking) callout to an AquaLogic Service Bus-registered proxy or business service
Skip	Specify that at run time, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow

Table 1-2 AquaLogic Service Bus Actions

Transport Headers	Set the transport header values in messages
Validate	Validate elements selected by an XPath expression against an XML schema element or a WSDL resource

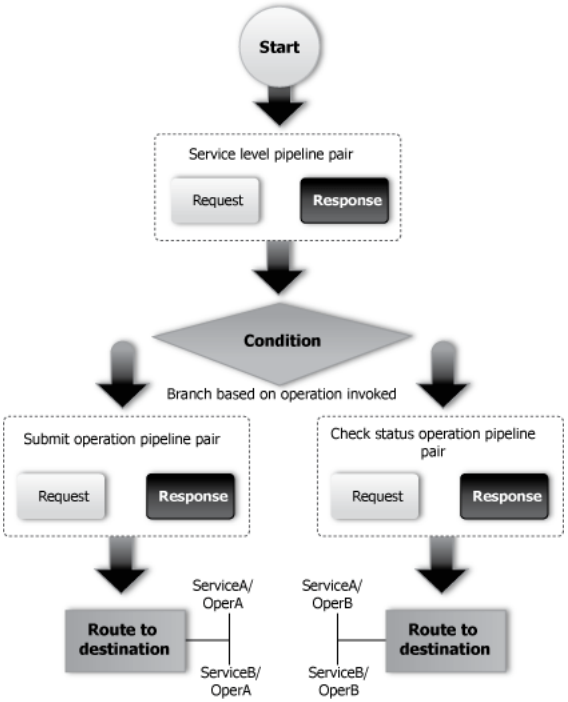
1. For information about the actions including how to configure them, see Proxy Services: Actions in *Using the AquaLogic Service Bus Console*.

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 1-11 Example Message Flow



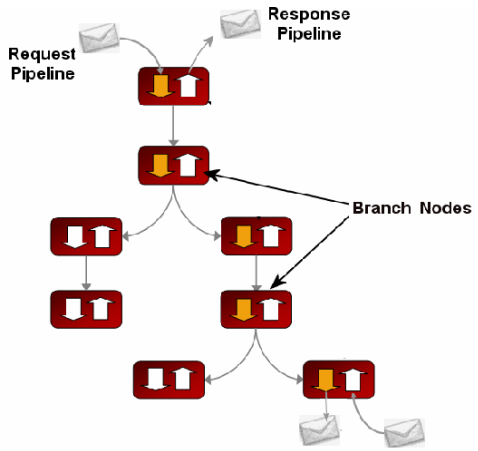
In the case of one-way operations, the response pipeline is executed with an empty message. This permits a response to be constructed for the proxy service, thus enabling bridging between request/response and one-way operations. The bridging mechanism means that proxy service input can be one-way while 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. Actions in the response flow may also be used to do post processing on the message after it has been routed to the business service or the proxy service.

Branch Nodes

A branch node allows processing to proceed down exactly one of several possible paths. Branching is driven by a simple lookup table with each branch tagged with a simple but unique string value. A variable in the message context is designated as the lookup variable for that node, and its value is used to determine which branch to follow. If no branch matches the value of the lookup variable, then a default branch is followed. Setting the value of the lookup variable must be done before reaching the branch node. This approach ensures that exceptions do not occur

within the branch node itself. A branch node may have several descendants in the message flow tree: one for each branch including the default branch.

Figure 1-12 Branch Nodes in a Message Flow



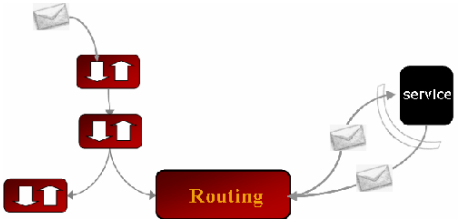
Route Nodes

The route node is used to perform request and 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, request processing is considered finished. When the route node receives a response message, response processing begins. The route node supports conditional routing as well as outbound and response transformations.

Note that you can choose to configure conditions in a route node or in a branch nodes.

The route node represents the boundary between request and response processing; consequently, it cannot have any descendants in the message flow tree.

Figure 1-13 Proxy Service Route Node Communicates With Services


service
Routing

CHAPTER 2

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

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 XML Schemas in *Using the AquaLogic Service Bus Console.*

AquaLogic Service Bus 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*.

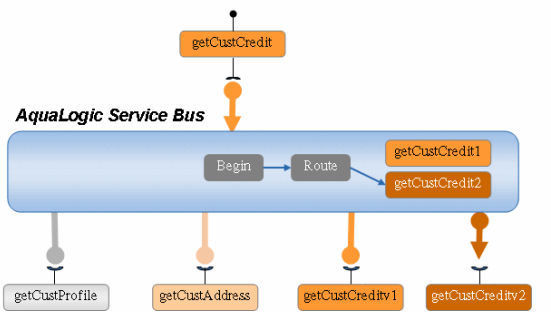
Transformation Maps

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.

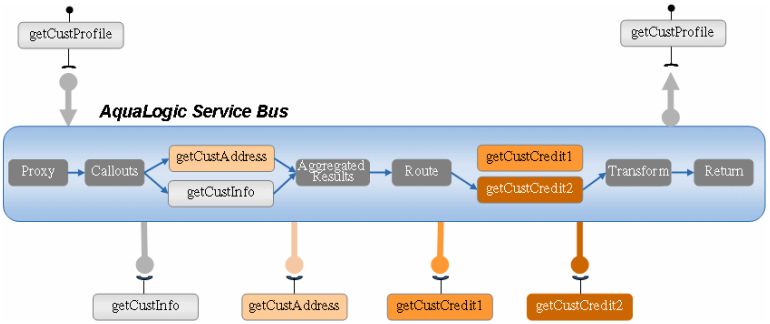
Conditional routing and transformations support content-based routing, allowing AquaLogic Service Bus to mediate between disparate service endpoints and intelligently route between them. In this way, loose coupling of SOA endpoints is achieved. A common example of content based routing is to support a scenario in which different versions of a service are made available. In the following figure, content-based routing at a route node in a proxy service mediates between two versions of a `getCustomerCredit` service.

Figure 2-1 Content-Based Routing in AquaLogic Service Bus



AquaLogic Service Bus allows you to enrich services by combining transformation and routing functions. Let us expand on the `getCustomerCredit` service use case described in the preceding figure to describe a scenario in which combining transformation and conditional routing functionality in a proxy service can result in enriched services and better reuse of those services. In this example scenario, Service Callouts are made to each of two services (getCustInfo and getCustAddress) and the results of the callouts combined by the proxy service. The same content-based routing is configured for `getCustomerCredit` service as was configured in the preceding figure.

Figure 2-2 Service Enrichment in AquaLogic Service Bus



For information on how to create AquaLogic Service Bus XQueries using the BEA XQuery Mapper, 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 WSDLS in *Using the AquaLogic Service Bus Console.*

WS-Policies

A WS-Policy describes a 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.

Policies are referenced by an URI, either embedded within a WSDL, an HTTP URI, or a policy URI (for example. `policy:myPolicy`). Policy URIs can reference in-built policies.

For more information on WS-Policy, see the *AquaLogic Service Bus Security 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 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 and messaging services. It also supports SOAP and XML services for which only the concrete interface and the service type are defined.

Messages to SOAP-based services are SOAP messages containing XML wrapped in a SOAP envelope. Messages to XML-based services are XML but can be of any type allowed by the proxy service configuration. Messages to a messaging service (a business service or proxy service created to be of messaging service type) can be of different types. A messaging service can exchange messages of very different content-type. These exchanges can be either request/response or one-way. Unlike Web services, the content-type of the request and response need not be the same. Messaging based services do not have WSDL definitions. To learn more about this service type, see Business Services in *Using the AquaLogic Service Bus Console*.

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

- WSDL Service
  - SOAP
  - XML

  - attachments described by the WSDL
- Messaging Service
  - Text
  - MFL
  - XML
  - Binary
  - untyped
  - attachments where the interface is not described by a WSDL
- 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 *Using the AquaLogic Service Bus Console.* 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 *Using the AquaLogic Service Bus Console.*

Service Interfaces

AquaLogic Service Bus relies on WSDLs for the formal description of Web 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 WSDLs in *Using the AquaLogic Service Bus Console.*

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 an interface, 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 *Using the AquaLogic Service Bus Console.* 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 *Using the AquaLogic Service Bus Console.*

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

Message Context

All messages sent to and received by the proxy service are defined internally in the proxy service by a set of properties that holds the message data and meta-data related to that message. This set of properties is known as the Message Context (context) and is implemented using Context Variables. It is defined by an XML schema. Each Context Variable relates to a different property.Some Context Variables are predefined and others are user defined.The heart of the proxy service is the *Message context*. For a complete description of the Message Context and context variables used in the message flow, see Message Context in *Using the AquaLogic Service Bus Console.*

Predefined 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. You typically use an XQuery expression to manipulate the context variable in the message flow. It can also be modified using transformation and in-place update actions.

The message related context variables `$header`, `$body`, and `$attachments` represent the canonical format of the message in the message flow. 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 `Body` element in `$body` contains the XML document. Attachments are in `$attachments`.
binary	The `Body` element in `$body` contains a reference XML document. Attachments are in `$attachments`.
MFL	The document is transparently converted from and to XML, and appears as an XML document in the `Body` element in `$body`. Attachments are in `$attachments`.
text	The `Body` element in `$body` contains the text. Attachments are in `$attachments`.

Table 2-1 Message mappings

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

In the case of attachments, `$attachments` contains the following for each attachment:

- attachment, 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 response is then finally sent from the top of the tree if the interface or operation was request/response, otherwise the response is discarded.

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 *Using the AquaLogic Service Bus Console*.

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 *Transforming Data Using the XQuery Mapper*.

WS-Security processing as well as authorization is transparently performed at the Start node. WS-Security processing is transparently performed when invoking a business service with a ws-policy. For information on how to configure AquaLogic Service Bus security, see the *AquaLogic Service Bus Security Guide*.

For more information about pipeline pairs, stages, and actions, see "Message Flow Definition" on page 1-18.

For an example of a typical message flow diagram, see Figure 1-11 in "Operational Pipelines" on page 1-21.

For more information on Message Flows, see Modeling Message Flows in the *AquaLogic Service Bus User Guide*.

Type System

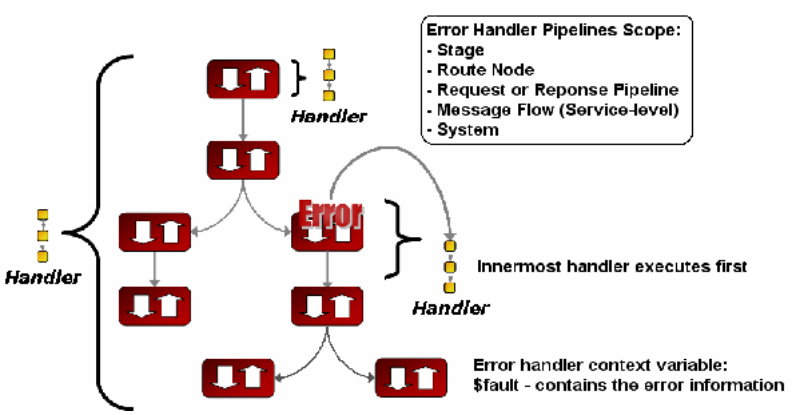
AquaLogic Service Bus has a built-in type system that is available for use at design time. When creating an XQuery expression in a condition, in-place update action, or transformation, the variable can be declared to be of a given type in an editor to assist in easily creating the XQuery. The types can 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 for an entire proxy service. The lowest scoped error pipeline that exists is invoked on an error.

Figure 2-3 Stage, Node, and Service-Level Error Handlers



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 *Using the AquaLogic Service Bus Console*.

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 the *AquaLogic Service Bus Security Guide*.

CHAPTER 3

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
- UDDI

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 *Using the AquaLogic Service Bus Console.*

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 resources, except cannot create, edit, or delete users, groups, roles, credentials, or access control policies.
IntegrationDeployer	IntegrationDeployers	Has read access to all objects. Can create, delete, edit, import, or export resources, services, proxy service providers, or projects.
IntegrationMonitor	IntegrationMonitors	Has read access to all AquaLogic Service Bus resources.
IntegrationOperator	IntegrationOperators	This group has the following privileges: • Has read access to all AquaLogic Service Bus resources • Has access to create, view, edit and delete alert rules • Has access to session management, including create, commit, discard and undo of sessions.

For information on how to manage AquaLogic Service Bus users, groups, and roles using the AquaLogic Service Bus Console, see Security Configuration in the *Using the AquaLogic Service Bus Console.*

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

For more information, see the *AquaLogic Service Bus Security 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`

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 AquaLogic Service

Bus to make routing decisions and perform transformations on the data based on the message content.

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

For more information, see the *AquaLogic Service Bus Security 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.

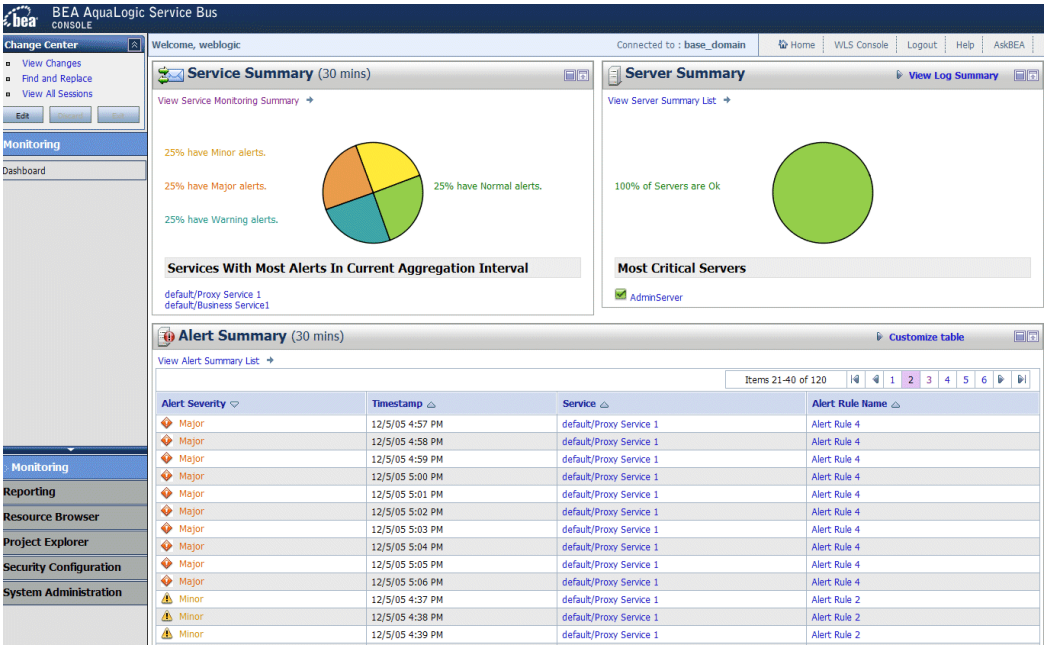
Using the features of your source code control system in conjunction with the configuration JAR files, you can provide version and change management for your AquaLogic Service Bus configurations.

For information on how to export and import configuration metadata using the AquaLogic Service Bus Console, see System Administration in the *Using the AquaLogic Service Bus Console*. 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 *Using the AquaLogic Service Bus Console*.

Dashboard, System Metrics, and Alerts

AquaLogic Service Bus aggregates run-time 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.

Figure 3-1 AquaLogic Service Bus Dashboard



Dashboard

The AquaLogic Service Bus Console Dashboard displays information about system health organized by server and services. The health of a service is further 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	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. 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.
Total Number of Messages	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.
Messages With Errors	Number of messages with error responses. 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. 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.
Success/Failure Ratio	(Total Number of Messages - Number of Messages with Errors)/Messages with Errors
Security	Number of messages with WS-Security errors. This metric is calculated for both proxy services and business services.
Validation	Number of validation actions in the flow that failed. This metric only applies to proxy services.

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 *Using the AquaLogic Service Bus Console.*

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.

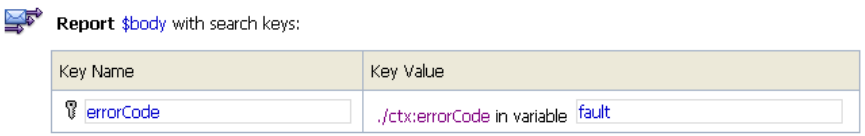
Rule and alert processing is handled by the AquaLogic Service Bus Alert Manager. The Alert Manager resides on the same single managed server as the metric aggregator for the system.

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 following figure shows an example Report action:

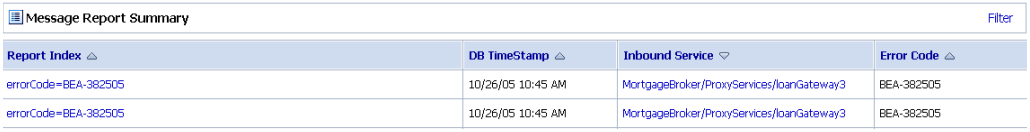
Figure 3-2 Example Report Action



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 *Using the AquaLogic Service Bus Console*.

The AquaLogic Service Bus Console Message Reporting module displays information from the Reporting Data Store. Message Reporting enables you to drill down from summary information to view detailed information about specific messages.

Figure 3-3 Example Message Report Summary in the AquaLogic Service Bus Dashboard



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.

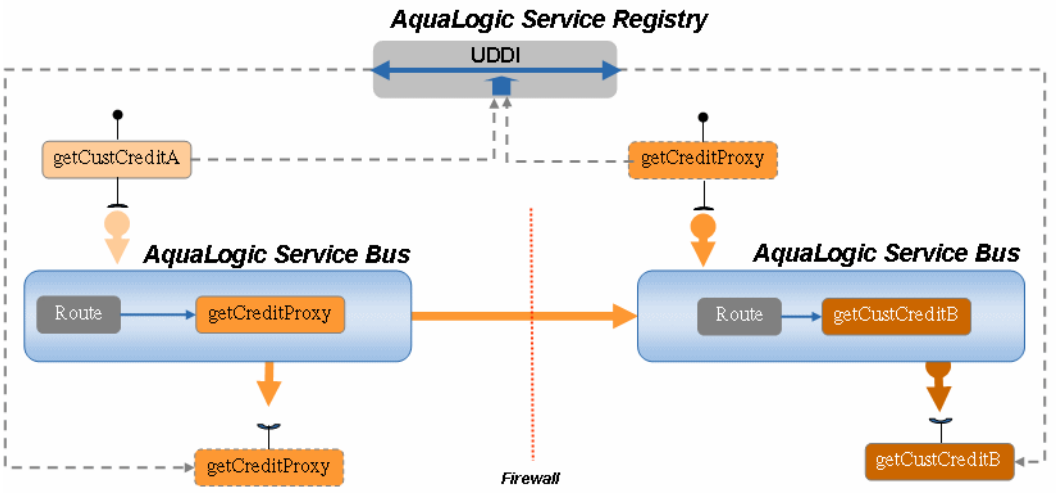
AquaLogic Service Bus Console provides purge functionality to help you manage your message data. For other data management functions, you should apply standard database administration practices to the database hosting the Reporting Data Store.

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

UDDI

Universal Description, Discovery and Integration (UDDI) registries are used in the enterprise to share Web services and in doing so UDDI services help companies organize and catalog these services for sharing and re-use in the enterprise or with trusted external partners.

Figure 3-4 AquaLogic Service Bus Leverages UDDI



The preceding figure illustrates how a UDDI registry can be made available to a distributed enterprise to facilitate reuse and sharing of services. In this case, customer credit services are registered with the UDDI registry and can be used by distributed instances of AquaLogic Service Bus.

A UDDI registry service for Web services is defined by the UDDI specification at the following URL:

`http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3`

UDDI registries are based on this standard specification. It provides the details on how to publish and locate information about Web services using UDDI. The specification does not define run-time aspects of the services (it is only a directory of the services). UDDI provides a framework in which to classify your business, its services, and the technical details about the services you want to expose.

BEA AquaLogic Service Registry is a version 3 compliant UDDI registry certified to work with AquaLogic Service Bus. It is not provided with AquaLogic Service Bus. It is licensed independently from BEA.

For information about AquaLogic Service Registry, see the product documentation at the following URL:

http://edocs.bea.com/alsr/docs21/index.html

AquaLogic Service Bus and UDDI

The AquaLogic Service Bus Console makes the AquaLogic Service Registry or any version 3 UDDI-compliant registry accessible and easy to use. In working with UDDI, AquaLogic Service Bus promotes the re-use of standards based Web services. In this way, AquaLogic Service Bus resources can be searched for and discovered and used by a wide and distributed audience. Web services and UDDI are all built on a set of standards, so re-use promotes the use of acceptable, tested Web services and application development standards across the enterprise. The Web services and interfaces can be catalogued by type, function, or classification so that they can be discovered and managed more easily.

A UDDI registry can be configured to work with AquaLogic Service Bus. Proxy services in AquaLogic Service Bus can be published to the registry. After the registry is configured, you can publish AquaLogic Service Bus proxy services to it. A registry entry has certain property types associated with it and these property types are defined when the registry is created.

You can use the AquaLogic Service Bus Console to publish proxy services to AquaLogic Service Registry. For information on how to publish proxy services to a UDDI registry, see “Publishing a Proxy Service to a UDDI Registry” in System Administration in *Using the AquaLogic Service Bus Console*. You must have an account set up in the UDDI registry to do this. You can publish all proxy services to a UDDI registry—this includes the following service types: WSDL, Messaging, Any SOAP, and Any XML.

You can import services from the registry as AquaLogic Service Bus business services. For information on how to import business services to AquaLogic Service Bus, see “Importing a Business Service from UDDI Registry” in System Administration in *Using the AquaLogic Service Bus Console*. The service types supported are WSDL services with SOAP over HTTP binding and AquaLogic Service Bus proxy services (used primarily in multi-domain deployments). When importing a WSDL-based service, if multiple UDDI binding templates are encountered, then a new business service is created for each binding template.

Permissions in BEA AquaLogic Service Registry were developed so that administrators can manage users' permissions in BEA AquaLogic Service Registry and create views into the

registry, specific to the needs of the different user types. User permissions set in AquaLogic Service Bus govern access to the registries, their content, and the functionality available to you.

The Benefits of Using UDDI to Solve an Enterprise Problem

The goal is to solve a complex multi-tiered problem by allowing everyone in the enterprise to benefit from increasing the visibility of business services and resources used in business applications. Enterprise architects need a way to ensure that the people who need the services can find them and that developers are not developing services that already exist. Management must ensure that services comply with technology, business policies, and application standards and IT and business leaders need to be able to control how the services interoperate both inside and outside the firewall. These basic requirements in the enterprise today are met using the architectural benefits of service-oriented architecture (SOA). A SOA can reliably map business processes with enterprise applications, inspire integration and reuse of applications, and foster effective governance of SOA services—often at dramatically lower costs and with fewer resources.

The benefits of using a global registry can be summed up in the following items:

- Alignment of business services with changing business needs—A business services registry allows business analysts to determine if the tools for implementing an enterprise's business process are available to address pressing business needs. Access to this information enables them to align processes and services used within the enterprise with changes in business requirements.
- Standards compliance and services integrity—A standards-based registry enables businesses to adhere to business and technical standards and the SOA on which the business is based.
- Visibility of the business services portfolio—Finding the right resources at the right time to implement a timely solution eliminating redundancy and cutting cost.
- Improved development cycle—Faster time to market. In other words, less discovery and development time is required searching for the resources that meet the business needs.

Related Topics

- AquaLogic Service Registry
- UDDI in *BEA AquaLogic Service Bus User Guide*

- Technical Notes can be found at the following URL. Specifically, *Using WSDL in a UDDI Registry* is recommended:

  http://www.oasis-open.org/committees/uddi-spec/doc/tns.htm

- UDDI product and development tool information is available on the OASIS UDDI Solutions page at the following URL:

  http://uddi.org/solutions.html

- The UDDI specifications, which are available at the following URL:

  http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

  These specifications define:

  – SOAP APIs that applications use to query and to publish information to a UDDI registry

  – XML Schema schemas of the registry data model and the SOAP message formats

  – WSDL definitions of the SOAP APIs

  – UDDI registry definitions (tModels) of various identifier and category systems that can be used to identify and categorize UDDI registrations

- Technical notes and best practice documents that help you deploy and use UDDI registries effectively are available on the OASIS UDDI Web site at the following URL:

  http://uddi.org

CHAPTER 4

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 resource cache. The resources in the AquaLogic Service Bus resource cache include WSDLs, XML Schemas, XQueries, XSLTs, MFLs, WS-Policies, Business Services, and Proxy Services.

This section includes the following topics:

Resource Cache Users

AquaLogic Service Bus is focused on supporting a set of trusted IT department specialists who manage the resources and services in the resource cache 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 resource cache.

Integration monitor users have full read access to the resource cache 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 resource cache and can only change the operational characteristics of the services.

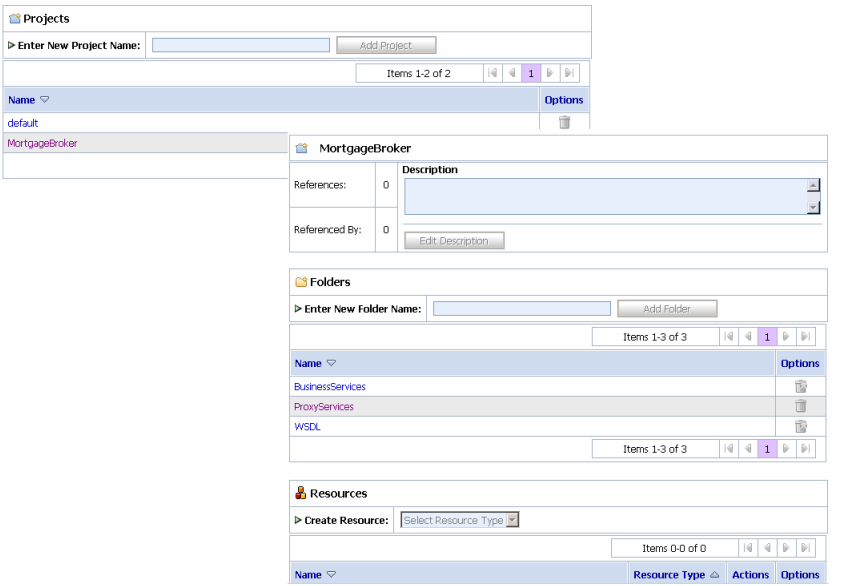
AquaLogic Service Bus is not focused on supporting large numbers of end users managing their resources with fine grained security, as would be required in a shared enterprise repository and services directory or in trading partner management systems.

For more information about AquaLogic Service Bus users and roles, see Security Configuration in *Using the AquaLogic Service Bus Console*.

Projects and Folders

The resources in the AquaLogic Service Bus resource cache can be organized into separate projects. The projects can contain folders, which in turn, can contain other folders. The following figure shows the project and folder views in the AquaLogic Service Bus Console.

Figure 4-1 Project Explorer View of AquaLogic Service Bus Projects and 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.

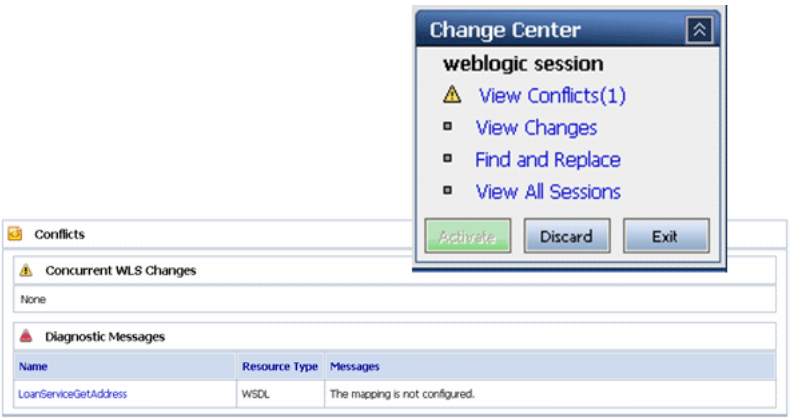
Resources can be moved between projects or folders and can be renamed. Resources that are located in one project can reference and use resources that are defined in other projects. Dependencies are preserved when resources are renamed and moved. All references to a renamed or moved resource are automatically adjusted. For more information about referenced resources, see "Tracking Dependencies" on page 4-5.

For more information, see Project Explorer in *Using the AquaLogic Service Bus Console*.

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.

Figure 4-2 Session Management in the Change Center



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 in transactions 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 *Using the AquaLogic Service Bus Console*.

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 *Using the AquaLogic Service Bus Console*.

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 resource cache.

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 *Using the AquaLogic Service Bus Console*.

For more information, see Using the Change Center in *Using the AquaLogic Service Bus Console*.

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 *Using the AquaLogic Service Bus Console*.

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*.