BEA Introduction to AquaLogic SOA Management

Version 2.6
Document Revised: April 2007

Table of Contents

3. ALSM Concepts and Architecture

Preface

Introduction to AquaLogic SOA Management is designed to familiarize you with the basic concepts and facilities of AquaLogic SOA Management. We recommend that you read this document prior to executing the tutorials found in the online help.

CHAPTER 1

Introduction to Service-Oriented Architecture

This chapter explains some of the basic concepts behind service-oriented architecture (SOA). It focuses on the concepts you should understand in order to effectively use AquaLogic SOA Management (ALSM) to manage your SOA system. If you are already well acquainted with SOA and web service technologies, you might want to skip to the next chapter.

Service-Oriented Architecture

An SOA system implements the delivery of software services to clients over a network. SOA differentiates itself from other systems by these features:

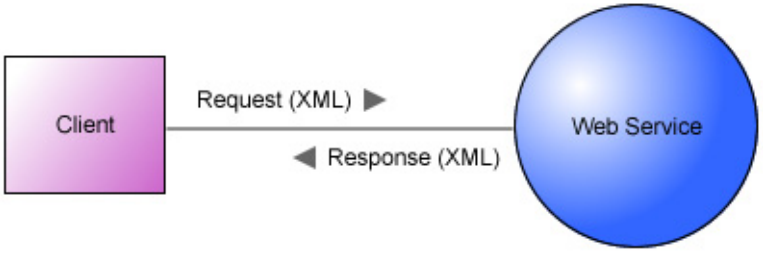
- System resources are made available as loosely-coupled, independent services.
- Services are made available through platform- and programming language- independent interfaces that are defined in a standardized way.
- Services are available both to clients and other services.

Most SOA systems are implemented using web services, which are discussed in the following sections.

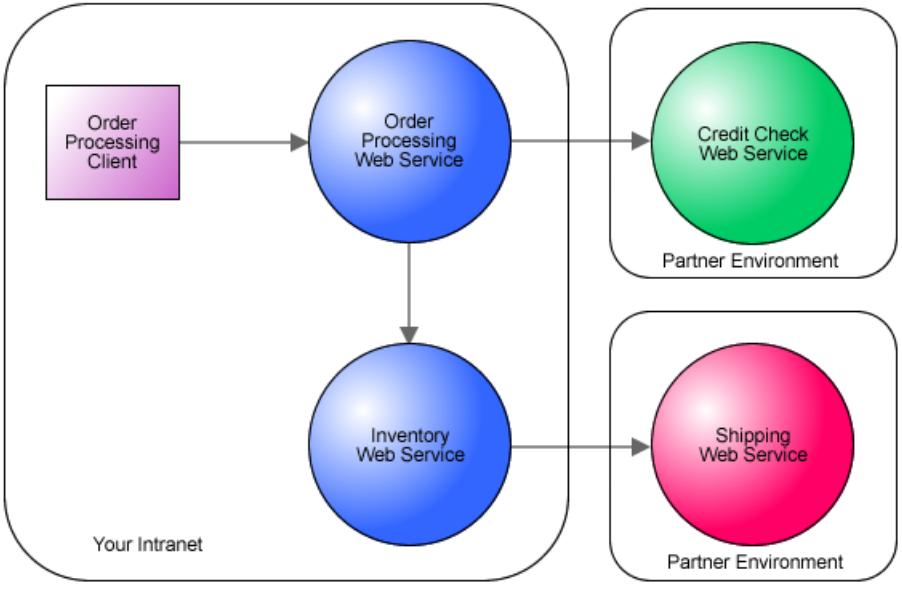
Web Services

A web service is a software component that accepts XML-based requests from a client to perform operations. In most cases, a web service returns an XML-based response after processing the request. Web services are language-, platform-, and location-independent: a client can contact a

web service that is written in a different programming language, running on a different platform, and located across the Internet.



Because web services can contact each other to request the execution of operations, they serve as ideal "building blocks" for distributed systems. Distributed systems composed of web services can span multiple machines, a corporate intranet, or the Internet. You can create a complex application that combines web services from your environment with web services outside of your environment, such as those operated by partners or suppliers.



Web Service Architecture

A web service runs in an application server that is referred to as a *container*. The installation unit that contains the web service is called a *deployment*, and is commonly represented by .war, .jar, or .ear files on Java. A deployment can contain one or more web services, as well as supporting files. You can install multiple deployments into one container.

Each web service has at least one URL at which the web service can be contacted. These URLs are called *endpoints* (ports in WSDL 1.1). Endpoints offer one or more *operations*, which define specific formats for request and response messages. In most cases, all endpoints on a service define the same set of operations.

Note: When you place a web service under ALSM, you are in fact managing one or more endpoints. In the case of multiple endpoints, you can specify different management behavior for each endpoint. For example, you might want to set up security differently outside the firewall than inside.

Web service architecture is based on these standardized elements:

- A transport protocol
- A message format protocol
- A service definition language
- A service registry

The function of each of these elements is described below.

The transport protocol indicates how messages are sent to the web service. HTTP (Hypertext Transport Protocol) is the most popular transport protocol for web services, as it makes possible universal connectivity via the Internet. HTTP is most commonly used to support request-response message patterns between a client and a web service. AquaLogic SOA Management supports HTTP and HTTPS transport protocols.

Messages between clients and web services typically use the SOAP message format protocol. SOAP is an industry-standard XML format that is programming language-independent and simple to understand. A client sends a SOAP request message to the web service, and receives a SOAP response message in return. Using SOAP, the service requestor and the service provider can communicate as long as they agree on a common transport protocol (such as HTTP) and the message's SOAP definition. This greatly increases the opportunities for reuse, as the service places essentially no constraints on the platform, language, or location of its clients. Note that

some web services exchange XML messages that do not use SOAP. AquaLogic SOA Management supports both SOAP and non-SOAP XML messages.

A web service's interface is typically defined using Web Services Description Language (WSDL), an interface description language defined in XML. To publish its operations to potential clients, a web service provides a WSDL file that describes the service's network address, the list of operations, and the messages that it uses to communicate. If a client has access to the WSDL file for a service and the proper security credentials, it can access the service with no additional information.

Web service clients typically locate a WSDL file using a published URL address. Clients can retrieve a WSDL file from a hard-coded URL location. Alternatively, you can use a service registry such as Universal Description, Discovery, and Integration (UDDI) to register and locate available web services. UDDI defines the format for a registry that allows web service providers to advertise their services and allows clients to locate the web services they need. Using UDDI, clients can dynamically discover web services that provide the required operations.

CHAPTER 2

Managing SOA Systems with ALSM

The use of service-oriented architecture (SOA) provides companies with unprecedented flexibility in how they architect their business applications. For example, SOA lets them:

- Expose existing application functionality as services (typically web services) and reuse them in new applications
- Rapidly compose business applications from services and expose these new applications as services
- Integrate business systems from end to end, across heterogeneous environments, resulting in a real-time flow of information and a real-time business environment

However, the ability to construct SOA enterprise systems poses new challenges for the organizations responsible for managing these distributed systems. Although the language-, platform-, and location-independent qualities of SOA simplify the construction of distributed systems, these qualities greatly complicate the problems of managing those systems.

Management Challenges

The disparate and complicated nature of SOA systems present basic management challenges that are common to most enterprise systems: visualizing the system, securing sensitive system components, and providing message routing. Beyond these basic management challenges, there are more specific management needs, such as tracking and resolving exceptions and managing service levels, that are imperative to the smooth operation of your system and the satisfaction of your clients.

Visualizing an SOA system

Before you can determine the basic management requirements for any system, you need to understand what components comprise the system and how these components relate to each other. Visualizing your system gives you the ability to pinpoint problems, such as traffic bottlenecks, and greatly improves your ability to manage the system.

Securing sensitive system components

Securing your SOA system is another management challenge. Most managers need to protect their services from unauthorized access (authentication). Many also need to provide different levels of access to different users (authorization), and to ensure the privacy of messaging (cryptography).

Providing message routing

An enterprise system is likely to require various forms of message routing. You might, for example, need to load balance requests, provide for service failover, or redirect messages to other services.

Managing Exceptions

SOA systems are composed of distributed, loosely-coupled services. The dependencies between services are often unclear, or entirely unknown. Some of the services comprising your system might even be outside your control. Qualities such as these make SOA systems inherently vulnerable to exceptions.

Typically, exceptions are first noticed downstream by service consumers who experience them as opaque error messages, delayed orders, lost packages, and so forth. Complaints from dissatisfied customers are often the first indication to the IT and business operations teams that something has gone wrong.

Diagnosing exceptions in an SOA system can be a frustrating and time-consuming experience. How do you tell whether an exception is caused at the network level by a transport error, or at the application level by invalid data in the request, or by an anomalous business event such as an overweight shipment or bad credit. It can take days for the IT operations team to comb the error logs of the participating software components and correlate the entries. Even then, they might not really understand what went wrong.

Because exceptions directly affect customers, management of exceptions is critically important. IT operations must detect, diagnose, and resolve exceptions quickly—in minutes rather than

hours or days. The business operations team must be notified of important exceptions in real time and be given the contextual information needed to resolve them before a customer logs a complaint.

Managing Service Levels

Companies use SOA systems to support their business activities in a number of ways. Some services are used internally by company employees. Others are provided to customers for a fee, for example on a pay-per-use or pay-per-time-period plan. Still others are provided for free as an inducement for customers to buy other services or products from the company.

These scenarios share a common management requirement: they each require a way to monitor and manage the services to ensure that their expected performance is being realized. In some cases, particularly those in which a customer pays for the service, companies stipulate guaranteed levels of performance that their customers will receive. In these cases, they need to historically track the performance so they can verify that the guaranteed levels of performance were actually delivered.

Service providers also require a way to track usage. For example, providers sometimes charge customers for services on a per-use basis. Or they simply cap the number of times a customer can use a service within a certain time period. In order to do this, they need to track usage by customer. Tracking usage also helps providers plan ahead for future capacity needs and to justify expenditure costs.

Visualizing your system, providing security, routing messages, tracking and resolving exceptions, guaranteeing service levels … How do you meet such management challenges? Service developers typically spend their resources implementing business functionality rather than system management facilities. While some implementers might incorporate system management facilities directly into their services, different organizations might do so with incompatible infrastructures, thereby making it impossible to manage a distributed system in a consistent and efficient manner.

In order to meet your management challenges with consistency and efficiency, you need a management system that lets you govern your entire SOA system, from end to end. And you need a management system that lets you apply your desired management behavior broadly, based on characteristics of your system.

AquaLogic SOA Management Features

AquaLogic SOA Management (ALSM) is a set of facilities that allow you to apply policy-based, runtime governance to your SOA system in a non-invasive manner. By tracking (and enhancing, if you so choose) the messages traveling to and from services, ALSM lets you improve performance and reliability, provide security for, and add functionality to your service-based applications without modifying the source code of the services. You can add these management enhancements incrementally, during development, testing, or at production time.

Discovery and Registration

ALSM defines a global view of your SOA system—this view is called a *sphere*. The sphere aggregates data about objects in your SOA system, such as containers, deployed applications, services, WSDL files, and so forth.

ALSM performs discovery at both the container level and the network level. ALSM automatically discovers services running on your network and registers them with the sphere. ALSM also discovers deployments to the containers in your network and maintains updated information about the deployments.

By observing message exchanges, ALSM discovers dependencies between the services, operations, and endpoints in your system.

Network Visualization and Dependency Analysis

ALSM visually depicts the discovered dependencies between services, operations, and endpoints. Dependency diagrams show the direction and amount of traffic between services as well as the current availability of the services. You can view traffic at the logical service level, or drill down and view traffic between the services' physical endpoints and operations.

Policy-Based Management

ALSM provides a policy-based approach to management, meaning that instead of programing management features to behave in a particular way, you simply declare the management behavior you want applied to your services. For example, you might declare that all request messages sent to the OrderService should be logged.

In addition to simplifying the application of management behavior, this policy-based approach enables you to manage large systems with consistency and efficiency. ALSM lets you apply policies according to any of a multitude of runtime characteristics of your system, enabling you

to manage multiple services at once. For example, you might define a logging policy that applies to all services in containers whose Life Cycle Phase attribute is set to Production.

And, if the characteristics of your system change, ALSM re-evaluates and reapplies your policies according to your policy-application criteria. For example, deploying a new service to such a container, or changing a container's Life Cycle Phase attribute from Development to Production, would dynamically prompt ALSM to begin logging the appropriate services.

You define policies using policy templates. ALSM provides policy templates for addressing most of your management needs. To address special needs, you can create custom policy templates. For example, you might create a policy template for regulating usage and then define a policy from that template that allows access 1am–5am on weekends only to users with the "admin" role.

You can define and redefine ALSM policies on the fly, without restarting your service containers or the ALSM system.

Security Management

ALSM lets you apply authentication, authorization, and cryptography policies to communications with your services.

ALSM can authenticate requests to your services by:

- Checking the signature on the request message
- Checking that the request message contains a valid Security Assertion Markup Language (SAML) assertion
- Authenticating the user-supplied credentials against a registered identity management system
- Relying upon container-based authentication

ALSM authorizes authenticated users to access your system based on their assigned roles. You can authorize access on individual operations or on classes of operations. (You can define your own operation classifications and assign them across multiple services.)

ALSM provides cryptography policies that:

- Decrypt and check signatures on inbound messages (both request and response)
- Sign and encrypt outbound messages (both response and request)

ALSM provides credential mapping policies that let you dynamically provide requests with the credentials needed to access a service. You can provide:

- A specified set of credentials for all authenticated requests to the service
- A specified set of credentials for requests from users with a specific role
- A SAML assertion for all authenticated requests to the service

Routing Management

ALSM lets you to intercept incoming requests to a service and route them to other services as needed. Using ALSM's routing policies, you can implement load balancing, failover, versioning, and redirection based on message content.

For example, in a purchasing application you might ascertain the geographic location of the buyer, from the message content, and direct the request to the service that handles orders for the appropriate region.

You can also transform incoming and outgoing messages using XSLT.

Auditing and Message Logging

ALSM lets you log the messages flowing through your SOA system. The logging mechanism is very flexible. For example, you might use logging to record a full audit trail of all communication with a service, or you might record only errors in service processing. **ALSM** lets you log:

- Messages for all operations on all services or for any subset of operations
- All types of messages (request, response, and fault), or only fault messages
- Only messages with specific content
- The entirety of messages, or only selected portions

You can also transform messages before you log them using XSLT.

Exception Management

ALSM provides end-to-end visibility into normal and exceptional business transactions. ALSM detects exceptions and provides diagnostic tools that let you quickly diagnose the underlying causes. You can resolve exceptions manually or set up resolutions that run automatically when a particular type of exception is detected.

Specifically, ALSM lets you:

- Track service faults as exceptions

- Model a business transaction as a group of correlated message exchanges called a *correlation*
- Specify patterns that constitute anomalous business events and issue exception alerts when these patterns are detected (a pattern can pertain to a single message exchange or a correlation)
- Selectively log and review message histories and use these histories to diagnose problems (you can log and review these histories by service operation or correlation)
- Manually resolve exceptions by editing and resending failed messages
- Set up resolutions that run automatically in response to particular exceptions
- Track exceptions using trouble tickets

Performance Management

By establishing performance metrics and constantly measuring your services, ALSM provides you with the information you need to provision your SOA system with appropriate resources, to set appropriate expectations of your services, and ensure that your services meet those expectations.

Specifically, ALSM lets you:

- Monitor your system performance by measuring response time, throughput, fault count, and availability of operations, services, and correlations
- Monitor performance from the point of view of the service or the client
- Set performance targets and issue an alert when a target is violated
- Automatically execute mitigation strategies in response to alerts
- Define service downtimes (periods when alerts are not issued)
- View real-time statistics and generate historical reports

Service-Level Management

ALSM lets you model your formal service-level agreements in order to ensure and verify that your services meet the expectations of your service consumers. ALSM *agreements* are composed of some number of service-level objectives, each of which is based on a performance metric

(response time, throughput, fault count, or availability) and that measures an operation or a correlation.

Specifically, ALSM lets you:

- Create agreements, composed of service-level objectives, and issue an alert when a service-level objective is violated
- Define customers to represent your service consumers and assign them to agreements
- Monitor service levels for individual customers
- Automatically execute mitigation strategies in response to alerts
- Define calendrical aspects of service-level management, such as service downtime, hours of evaluation for agreements, and start and end dates for agreements
- View real-time statistics and generate historical reports

CHAPTER 3

ALSM Concepts and Architecture

This chapter examines the architecture of AquaLogic SOA Management (ALSM) and describes how its components work together to provide management for your SOA system.

Architecture Overview

AquaLogic SOA Management is designed to enforce management policy for your SOA system. As such, ALSM runs in the execution environment of your SOA system and, logically, employs a service-oriented architecture itself.

The management system consists of a number of services that work together to implement management functionality. These services can be classified into two broad groups—*management intermediaries* and *management services*.

Management intermediaries are positioned in the flow of your SOA system's message traffic. They directly implement the management behavior you desire by observing and sometimes manipulating the message traffic. ALSM intermediaries are called *agents*. ALSM also interoperates with certain 3rd-party intermediaries.

Management services perform supporting functions that help the management system function as a whole. For example, management services discover and model the components of your SOA system and support the formulation and distribution of management policy. Some management services also implement management behavior that does not require the manipulation of message traffic, such as the aggregation and analysis of management data.

Management services communicate with each other by exchanging messages. Each of these messages, and the data they contain, is an XML document whose structure is defined by XML Schema. The totality of these schemas defines the common data model used by ALSM.

A web-based user interface, called the ALSM Console, enables you to communicate your desired management behavior to the management services, by way of XML messaging. The management application also gives you access to management data, for example, the status of your SOA system and the current configuration of your management system.

Management by Policy

ALSM manages the behavior of your SOA system by applying *management policies* to the *manageable objects* of your system. Management policies are declarations of the management behavior you desire. The manageable objects of your system are:

- Endpoints
- Services (which can comprise one or more endpoints)
- Correlations (a set of operations, typically related by the flow of messages between them)
- Entry points (an address on a hardware or software device that allows entry into your SOA system, typically on a load-balancing device)

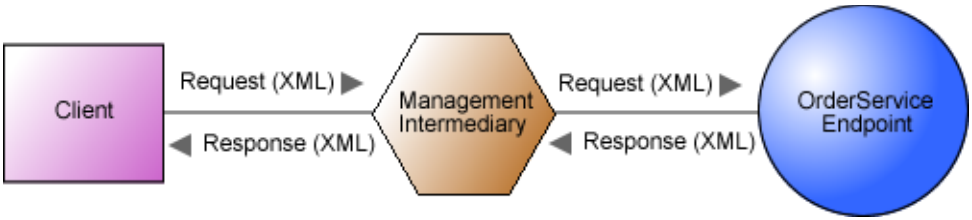
When you apply a management policy, you do so according to characteristics of the objects to which you want to apply the policy. For example, in a very simple case, you might apply a policy to a service called OrderService. In this case, the policy's application criterion is simply "service name = OrderService".

However, because ALSM periodically re-evaluates its policy application criteria, you might want to apply policies according to variable characteristics. For example, you might apply a policy to all services in containers whose Life Cycle Phase attribute is set to Production. In this case, changing a container's Life Cycle Phase attribute from Development to Production would dynamically prompt ALSM to begin applying the policy to the services in that container.

ALSM Agents

ALSM applies management policies by way of a configurable piece of software or hardware interposed in the message flow. This piece of software is called a *management intermediary*. The management intermediary intercepts messages flowing to and from the managed object and applies management policies to those messages.

For example, assume we have a service called OrderService with a single endpoint that accepts a message containing an order. ALSM manages this service by inserting an intermediary in front of the endpoint and intercepting the messages destined for the service's endpoint. Both the requester and the service are unaware that the intermediary is intercepting messages.



Once the management intermediary is in place, the service is considered to be a *managed service*. As an ALSM user, you define management policies for the service, and the intermediary applies the management policies. These policies can modify the message, redirect the message, or extract management information from the message and report it back to you.

ALSM provides a number of intermediary implementations that you can use for managing your SOA system. ALSM intermediaries are called *agents*. ALSM also interoperates with certain 3rd-party management intermediaries (see "Integration with 3rd-Party Management Intermediaries" on page 13). The remainder of this section describes ALSM's agent implementations.

Active vs. Passive Management

ALSM agent implementations are functionally of two types: those that provide active management capabilities, and those that provide passive management capabilities.

Agents designed for *active management* can examine and manipulate messages in order to augment the service's functionality. The agent can perform processing before passing the request message to the service, and before returning the service's response to the client. In this case, message processing is performed in-band, that is, before the message is passed on to the intended recipient.

Agents designed for *passive management* can examine but cannot manipulate messages before passing them on. The agent can, however, record and react to the messages that it observes. In this case, processing is performed out-of-band, that is, after the message is passed on to the intended recipient.

In general, active management is more powerful than passive management because you can manipulate messages before passing them on to the service, and you can redirect messages to different services. Passive management has the advantage of being more lightweight. Since processing is performed out-of-band, the effect of passive management on performance of the service is negligible.

You can configure agents to perform tasks such as:

- Logging messages to and from a service
- Measuring the response time of a service
- Load-balancing multiple instances of a service

The capabilities of any particular agent implementation depends on the manner in which the agent is deployed.

Agent Deployment and Types

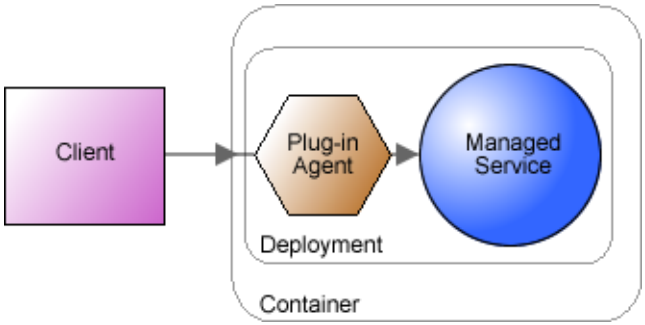
ALSM provides several types of agent, each of which is deployed in a different manner. You can deploy an agent:

- In the messaging layer of the managed service (*plug-in agent*)
- As a set of libraries in the deployment of the managed service plus a separate web application (*nano agent*)
- As a set of libraries in the deployment of the client (*client-side agent*)

Plug-In Agents

A plug-in agent runs with the service it manages and intercepts messages sent just to that service. When you use a plug-in agent, clients can continue to access the service directly, because the agent is installed in the messaging layer of the service's container. Therefore, you do not need to redirect clients to the agent's URL. A plug-in agent can manage services only in the same deployment.

In a Java container, a plug-in agent runs as either a servlet filter or a JAX-RPC handler.



Plug-in agents support active management. You configure a plug-in agent by applying management policies in the ALSM Management Console.

Nano Agents

You can use a nano agent to monitor JAX-RPC services, EnterpriseJava Beans (EJBs), databases (via JDBC), and remote method invocation services (RMIs).

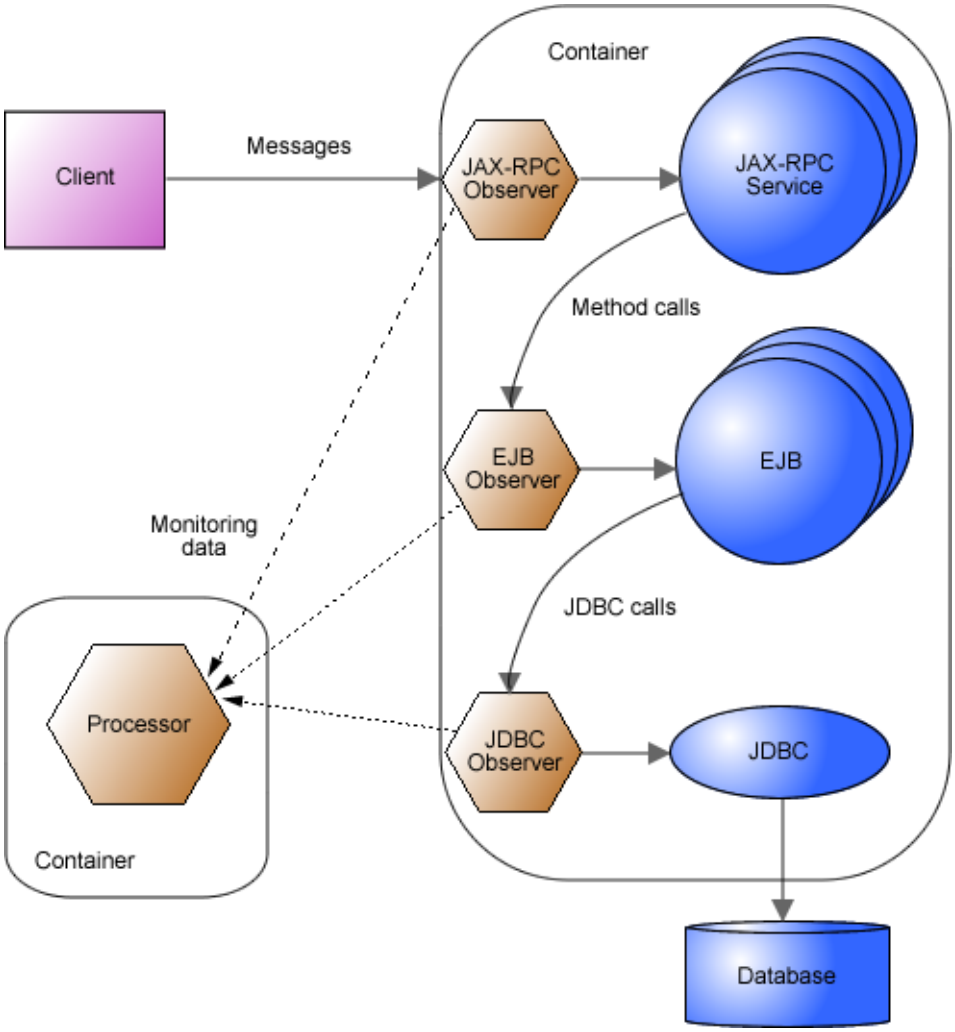
A nano-agent consists of two parts—an observer and a processor.

The nano-agent observer runs in the container hosting the component to be managed and intercepts messages or calls sent to that component. The observer reads these messages or calls as they are passed to the component. The observer converts the messages and calls to XML format and forwards these to the nano-agent processor.

The nano-agent processor runs as a web service in a separate container. The processor performs whatever processing is needed for monitoring of the services.

Nano-agent observers are classified according to the type of service they monitor, for example, EJB observers monitor EJBs. An observer can monitor any number of components of its type that are running in the container.

When you use a nano agent, clients can continue to access the component directly, because the nano-agent observer is installed in the byte code of the component's container. You do not need to redirect clients to the agent's URL.



You configure the nano agent processor by applying management policies in the ALSM Management Console. You configure the nano-agent observer by editing a configuration file in the service's container. The nano agent supports passive management only.

The nano agent architecture is extensible so that users can extend the types of components that can be monitored by the observer.

Client-Side Agent

Client-side agents enable you to extend ALSM management to your service clients. Client-side agents:

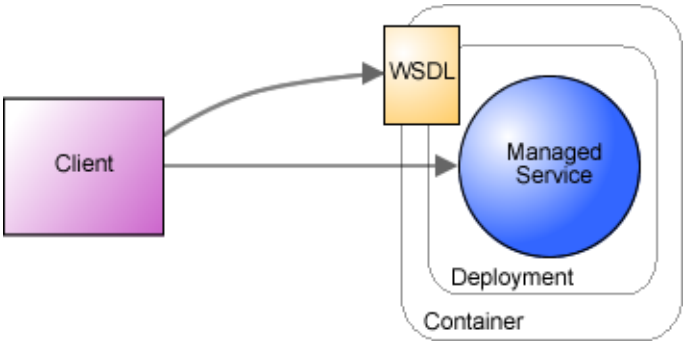
- Enable clients to look up service endpoint addresses dynamically by way of a UDDI registry or from the ALSM sphere service
- Let you enhance security by:
  - generating and attaching a WS-Security header, containing user credentials, to outbound SOAP messages
  - encrypting messages between the client-side agent and the agent managing the service
  - signing messages
- Enable ALSM to measure service response time from the client's point of view (the time from when the client issues a request until it receives the response)

Client-side agents can dynamically download cryptography and authentication policies from the service-side agent. For example, if the service-side agent is configured to require encrypted messages, it notifies the client-side agent of the requirement. The client-side agent, if configured to do so, then dynamically downloads the appropriate cryptography policy from the service-side agent to enable it to encrypt messages.

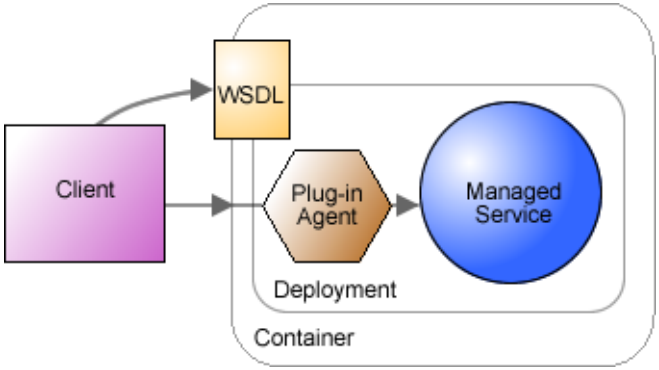
You configure the client-side agent with a configuration file located in the client's container.

How Agents Communicate

In the absence of an ALSM agent, a client contacts a service by first retrieving its WSDL file. The WSDL file lists the operations and endpoints that are provided by the service, defines the structure of the SOAP messages used to request the operations, and indicates the URL endpoints where the service can be contacted. Once the client has the WSDL file, it can contact the service and request its operations.

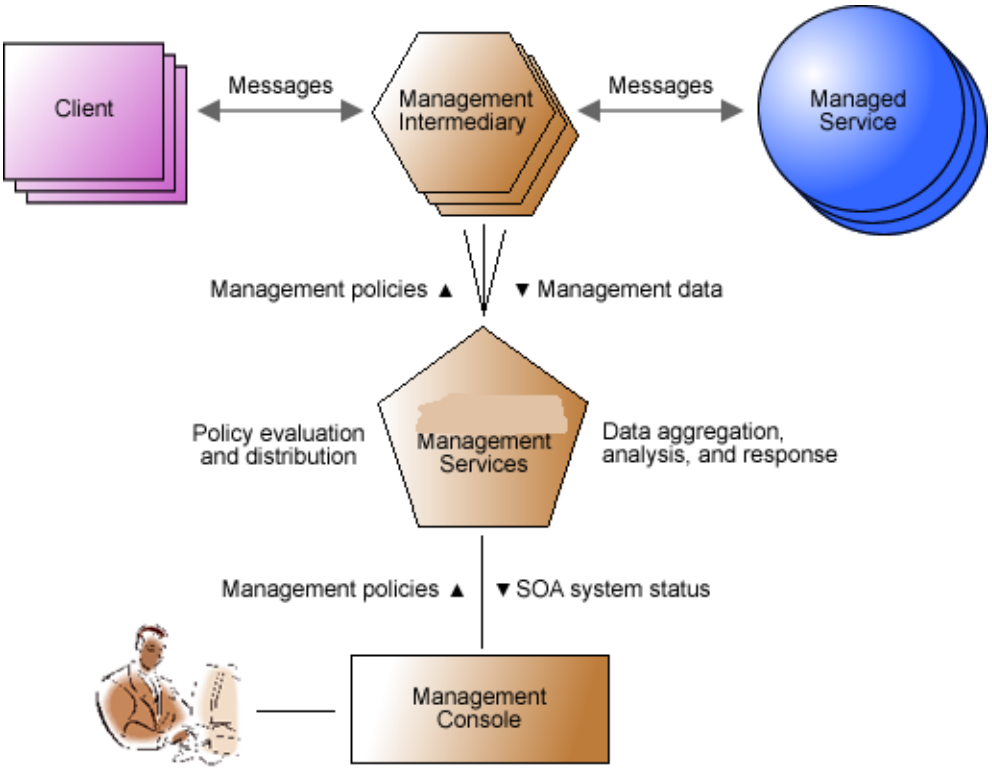


When you use a plug-in agent or nano agent, the clients continue to access the web service's WSDL file directly, because the plug-in agent is installed in the service container's messaging layer and the nano agent is installed in the service container's byte code. Therefore, you do not need to redirect clients to the agent's URL.



Management Services

Management services are those ALSM services that operate outside the flow of your SOA system's message traffic. Management services discover and model the components of your SOA system and support the formulation and distribution of management policy. The following diagram illustrates how management services fit into the AquaLogic SOA Management.



As an ALSM user, you define policies in the Management Console and specify the characteristics of the objects to which those policies should apply. The ALSM management services evaluate the policies and distribute them to the appropriate management intermediaries. The intermediaries apply the policies to your SOA system's message traffic and return management data to the management services. The management services aggregate and analyze the data and send the status of your SOA system to the ALSM Management Console.

The management services can also respond to management data as dictated by your management policies. They could, for example, execute mitigation strategies and send email notifications to interested parties.

The remainder of this section describes the most important management services.

Per-Container Services

The following services are deployed in each container in your ALSM system.

Container Service

Each container service maintains a data model of the container in which it's installed. This data model is a subset of the global data model maintained by your ALSM system. The sphere service depends on the container services to build and maintain the global data model (see "Sphere Service" on page 11).

There is one container service in each container in your ALSM *sphere*. The sphere, from an architectural standpoint, is your ALSM system's highest level of organization. Container services define the sphere. That is, the sphere is composed of all containers that host a container service, along with all services running in those containers.

The container service also acts as a coordination point for ALSM activities within the container. The container service supports these activities:

- Placing service endpoints under management
- Configuring intermediaries to apply management policies to messages
- Discovery of services installed in the container

Deployment Service

The deployment service discovers services running in the container and deploys plug-in agents as needed.

Local Traffic Analyzer Service

The local traffic analyzer service correlates messages that have passed through the container. These correlated messages are used by the distributed traffic analyzer service to monitor dependencies between services. They are also used by the ExM service to correlate messages for business processes that span multiple containers.

Sphere-Wide Services

The following services are deployed in only one container of your ALSM system.

Sphere Service

The sphere service maintains a global view of your managed SOA system. It aggregates data about the containers and services in your ALSM sphere and stores the data in a registry. It provides a SOAP-based query interface to the registry that is accessible to other ALSM management services. For, example, the Management Console uses the registry data to navigate the system.

There is one sphere service per sphere.

Service Registry Service

The service registry service registers and maintains definitions of the services in the sphere. The service definitions are stored in a relational database. This service is accessed for registration of services by way of a SOAP interface.

The service registry service is hosted in the same container as the sphere service.

Distributed Traffic Analyzer Service

The distributed traffic analyzer service correlates messages across containers. This functionality is used to generate graphs that depict message traffic in your system. It also supports the defining of correlations used for exception tracking.

The distributed traffic analyzer service is hosted in the same container as the sphere service.

Policy Manager Service

The policy manager service manages the creation, storage, and application of policies for the sphere. It periodically evaluates its policy application criteria and, according to changes in the environment or changes to the policies, reapplies the policies by pushing them out to the appropriate management intermediaries.

The policy manager service is hosted in the same container as the sphere service.

Service Level Manager Service

The management service responsible for coordinating ALSM's service-level management activities is called the service-level manager (SLM) service. The SLM service:

- Distributes service-level management policies to the appropriate management intermediaries
- Aggregates and evaluates the service-level measurements collected by those intermediaries

- Stores measurements and evaluation results in the SLM database
- Caches recent measurements for efficient lookup
- Issues alerts in response to anomalous service-level conditions

There is one SLM service per sphere. It can, but need not, be hosted in the same container as the sphere service. The SLM service can require significant processing resources. You should carefully consider where you position it within your architecture.

Exception Manager Service

The management service responsible for coordinating ALSM's exception management activities is called the exception manager (ExM) service. The ExM service:

- Distributes exception management policies to the appropriate management intermediaries
- Correlates messages for business transactions that span multiple message exchanges
- Detects patterns for exception conditions you have defined
- Stores runtime data on correlated messages and exceptions in the ExM database
- Logs and retrieves message histories
- Executes resolutions in response to detected exceptions
- Issues alerts in response to detected exceptions

There is one ExM service per sphere. It can, but need not, be hosted in the same container that hosts the sphere service. The ExM service can require significant processing resources. You should carefully consider where you position it within your architecture.

Notifier Service

The notifier service issues email notifications (by way of an email server) in response to important events detected by your ALSM system. These types of events include exceptions, service-target warnings and violations, service-level objective warnings and violations, and errors in the ALSM system itself. ALSM users can register to receive notifications for the events that they are interested in.

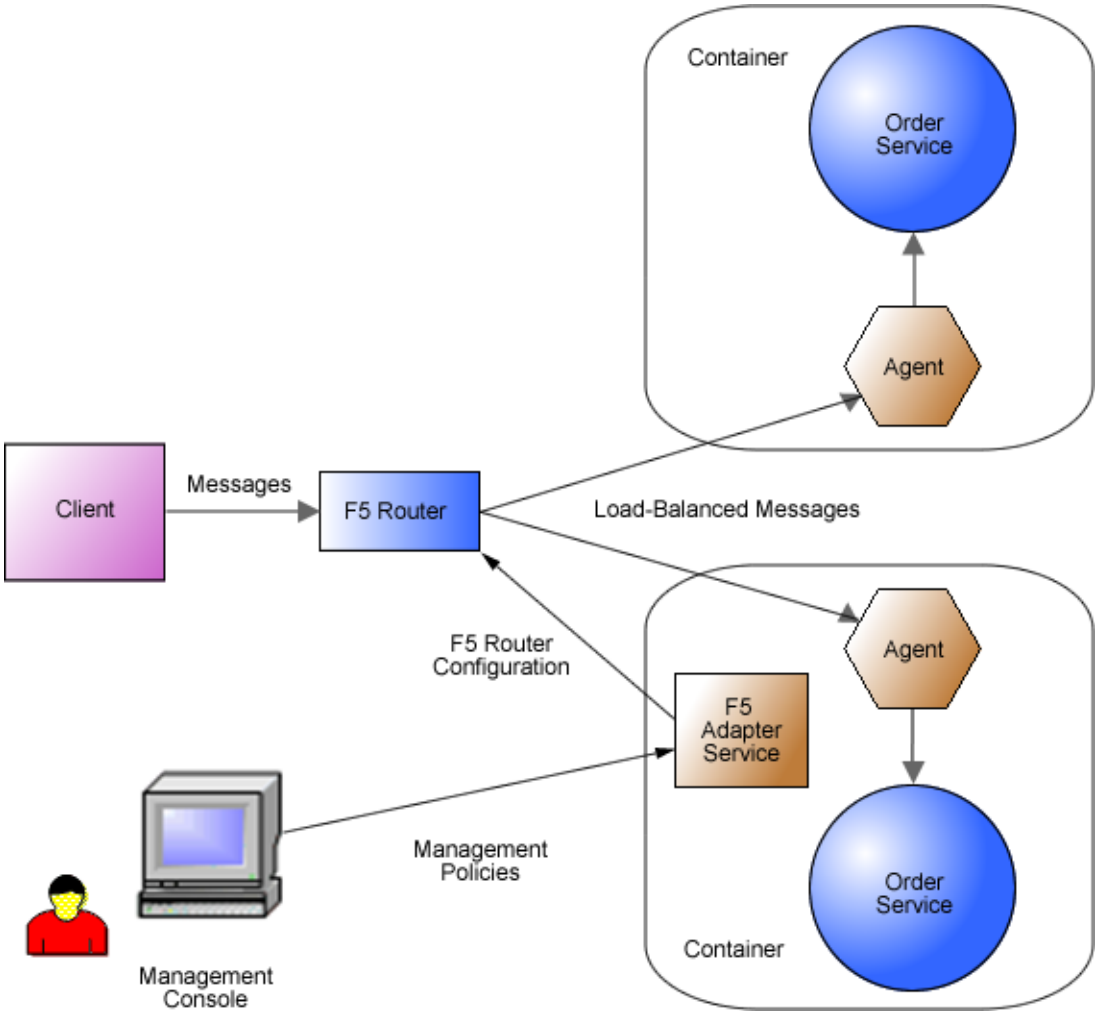
There is normally one notifier service per sphere. However, if you install the SLM and ExM services in separate containers, each of these containers can have its own notifier service.

Integration with 3rd-Party Management Intermediaries

Intermediary adapters let you apply management policies to services in cooperation with 3rd-party hardware and software management devices, such as the F5 router. The adapter enables ALSM to model and display the device and lets you configure the device's management facilities by applying management policies in the ALSM Management Console. The adapter effectively adds the 3rd-party device to your AquaLogic SOA Management.

The adapter is installed as a separate web application and runs as a service inside a container already hosting an ALSM deployment. It can, but need not, run in the same container as the managed services, the agents managing the services, or the 3rd-party management device.

For example, the following diagram shows a typical deployment of ALSM-managed services, load balanced by an F5 router. In the diagram, business traffic flows through the F5 router, which load balances the traffic across several servers. The ALSM F5 adapter service facilitates discovery of the F5 routing policies. The adapter also facilitates the reconfiguration of those policies.



ALSM Management Console

The management console provides a web-based user interface to the AquaLogic SOA Management facilities. This user interface is called the ALSM Management Console.

The management console is deployed as three separate web applications. All three can, but need not, be deployed to the same container. In any case, one is deployed to the container that hosts

the sphere service, one to the container that hosts the SLM service, and one to the container that hosts the ExM service.

Access to the ALSM Management Console is controlled through both container-level and application-level security. Authentication of users, and assignment of user roles, is performed by the container. Authorization of users to access features is performed by the management console application, based on the user roles. User accounts need to be configured for each container in which the management console applications are deployed.

The ALSM Database

AquaLogic SOA Management uses databases for persistent storage. The databases store:

- ALSM-defined objects, such as policies, services, service-level objectives, customers, correlations, and conditions
- runtime monitoring data, such as throughput, average response time, and availability
- evaluations of service-level objectives and service targets
- runtime-generated objects, such as exceptions, correlation instances, and alerts
- messages (messages can be logged in the database or in files)

Generally, each management service that requires persistent storage has its own database. These databases are individually configurable.

A Hypersonic database system is installed along with ALSM. By default, ALSM is configured to use this database system for persistent storage. The Hypersonic database system is intended for demonstration and initial evaluation purposes. For development, testing, and production systems you can configure ALSM to use a production-level, external database system.

For information on configuring ALSM to use an external database, see the online Help.