

HP Service Oriented Architecture Manager

Broker User Guide

Version: 2.52

Windows®, HP-UX, Linux



February 2008

© Copyright 2004-2008 Hewlett-Packard Development Company, L.P.

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 2004- 2008 Hewlett-Packard Development Company, L.P., all rights reserved.

Trademark Notices

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation

UNIX® is a registered trademark of The Open Group

Support

You can visit the HP Software support web site at:

www.hp.com/go/hpsoftwaresupport

This Web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Table of Contents

SOA Manager Broker-An Overview	1-1
Prerequisites	1-1
Contextual Overview	1-1
Broker Configurator	1-1
Common Handlers	1-2
Monitoring Handler.....	1-2
Logging Handler.....	1-2
Auditing Handler	1-2
Schema Validation Handler	1-2
Business Content Alerting Handler	1-3
Security Handlers.....	1-3
Using Intermediary Services.....	2-1
Overview	2-1
Viewing Intermediary Service Details	2-1
Performance Metrics.....	2-2
Undeploying an Intermediary Service	2-2
Deploying an Intermediary Service	2-2
Editing an Intermediary Service	2-3
Changing an Intermediary Service's Version	2-3
Configuring an Intermediary Service's HTTP Path.....	2-3
Removing an Intermediary Service	2-4
Enabling Protocol Switching at the Intermediary	2-4
Prerequisites	2-4
Enabling JMS-to-JMS-Two-Way Protocol Switching.....	2-5

Enabling HTTP-to-JMS-One-Way Protocol Switching	2-6
Enabling JMS-to-HTTP-One-Way Protocol Switching	2-7
Configuring Handlers	3-1
Audit Handler	3-1
Fields	3-1
Configuring the Audit Publisher	3-2
Business Metric Alerts Handler	3-2
Fields	3-3
Generic SOAP Contract Handler.....	3-3
Fields	3-3
HTTP Pass-Through Transport Header Handler.....	3-4
Invocation Handler	3-4
Fields	3-4
Log Handler.....	3-4
Fields	3-4
Schema Validation Handler.....	3-5
Service Protection Handler.....	3-5
Fields	3-5
Content Detection Handler	3-5
Fields	3-5
Scheduled Availability Handler.....	3-6
Fields	3-6
Security Auditing	3-7
Field.....	3-7
Configuring Security Auditing.....	3-7
Service Security Inbound Handler.....	3-7
SOAP Contract Handler	3-7
SOAP Dispatch Handler.....	3-7
SOAP Monitoring Handler.....	3-8
Fields	3-8

WS Security Outbound Handler	3-8
Fields	3-8
WS Security Message Processing Inbound Handler.....	3-9
Fields	3-9
XML Contract Handler.....	3-10
XML Dispatch Handler	3-10
XPath Monitoring.....	3-10
Fields	3-10
XSLT Handler.....	3-11
Fields	3-11
Classifier Handler.....	3-11
Fields	3-11
Using Custom Intermediary Services	4-1
Overview	4-1
Convert a Simple Intermediary Service	4-1
Adding Handlers.....	4-2
Adding Custom Handlers	4-2
Defining Service Providers for Custom Web Services	4-3
Enabling Content-based Routing	4-4
Getting Started	5-1
Starting the WSM Broker.....	5-1
Stopping the WSM Broker.....	5-2
Windows	5-2
UNIX	5-2
Starting the Broker Configurator Console	5-2
Installing the Broker as a Windows Service	5-3
Configuring HTTP Settings.....	5-3
Configuring the HTTP Server Port Number	5-4
Configuring the Broker's Management Channel Port.....	5-4
Configuring HTTP Server Thread Settings	5-5

Configuring HTTP Client Settings	5-5
Configuring HTTP Proxy Settings	5-6
Assigning Access to the Console	5-6
Using XPL Logging	5-7
Installing XPL Logging	5-7
XPL Tools	5-7
Configuring XPL.....	5-7
Configuring Log Levels	5-8
Viewing Logs	5-9
Using XPL Tracing	5-9
Installation.....	5-9
Windows	5-9
HP-UX	5-9
Linux	5-9
Example Configuration Entries	5-10
Implementing Load Balancing and Failover.....	6-1
Overview	6-1
Conceptual Architecture	6-2
Load Balancing Scenario	6-2
Failover Scenario	6-2
Setting Up Load Balancing and Failover	6-2
Defining Multiple Endpoints in a WSDL File.....	6-3
Configuring Load Balancing and Failover	6-3
Using Multiple Intermediaries	6-4
Using the Intermediary’s Security Features.....	7-1
Overview	7-1
Feature Matrix.....	7-1
Supported Security Scenarios	7-2
Scenario 1: Intermediary is the Entry Point for External Consumers	7-3
Scenario 2: Web Application is the Entry Point for External Consumers	7-4
Scenario 3: Intermediary is the Exit Point for External Providers.....	7-4
Transport Level Security	7-4

Message Level Security	7-5
Inbound Message Processing.....	7-6
Outbound Message Processing.....	7-6
Setting Up the Security Components	7-6
Configure a Key Store.....	7-7
Configure a CA Trust Store.....	7-7
Configure the Intermediary's SSL Port.....	7-8
Setting Up Authentication and Authorization.....	7-8
Implementing a Security Scenario.....	7-9
Inbound Transport Security.....	7-9
Enabling SSL	7-9
Enabling Authentication	7-10
Outbound Transport Security	7-10
Enabling Outbound SSL	7-10
Inbound Message Security	7-11
Outbound Message Security.....	7-12
Management Channel HTTP Basic Authorization	7-14
Troubleshooting Broker	A-1
Installation and Configuration Problems.....	A-1
Errors occurred during installation	A-1
AutoPass fails to install	A-1
Runtime Problems.....	A-2
Could not start monarch-sba.....	A-2
Failed to initialize listener.....	A-3
Unable to determine binding from message element.....	A-3
Authentication header not progressed to backend.....	A-3
Out of Memory	A-4

SOA Manager Broker-An Overview

The SOA Manager Broker (WSM Intermediary) is responsible for collecting management data for Web services. The Intermediary runs in its own Java process and delegates service requests through a proxy (intermediary service) to Web services that are deployed in a Web Service Container. An intermediary service must be created for each Web service that you want to manage.

Prerequisites

Users must have fundamental knowledge of the Java programming language and Java platform technologies including security. Users should also have fundamental knowledge of Web services principles and be familiar with their application hosting environment.

Contextual Overview

Intermediary services utilize the Intermediary's handlers, which mediate the communication between a client and a Web service. The handler can be configured with sub-handlers (referred to as common handlers) that provide varying levels of manageability (Monitoring, Logging/Auditing, etc...). The Broker Configurator is used to create intermediary services and configure handlers for intermediary services.

The Smart Business Agent (SBA) provides a method of exposing data and metrics as Web services using WS-based management protocols. Managed objects collect data and metrics from the handlers. The data is represented in the SOA Manager and viewed using the SOA Manager web interface.

Broker Configurator

The Broker Configurator is a Web application that allows you to interact with the Intermediary. In particular, the Broker Configurator is used to configure the Intermediary, create intermediary services, and configure an intermediary service's handlers.

Common Handlers

A handler can contain any number of sub-handlers known as common handlers. Together, the handlers are considered a handler chain. The common handlers for a simple intermediary service are described below. Some handlers are enabled by default when you create an intermediary service, while other handlers must be manually enabled. In addition, custom intermediary services provide an expanded list of handlers and the ability to add any custom handler.

Monitoring Handler

The Monitoring Handler collects performance data for a Web service. The data is reported over a period of time (the last 6 minutes, 1 hour, and 1 day). In particular, the handler reports:

- Average Response Time
- Maximum Response Time
- Minimum Response Time
- Security Violations
- Total Request Count
- Total Failure Count
- Total Success Count
- Availability %
- Uptime %

Logging Handler

The Logging Handler is used to collect and publish the Intermediary's log messages. The log messages can be used to troubleshoot any problems that occur with the Intermediary.

Auditing Handler

The Auditing Handler provides message tracing capabilities for a Web service. The handler can be configured to also include SOAP payload for the message.

Schema Validation Handler

The Schema Validation Handler is used to validate Doc Literal SOAP messages to ensure that they comply with the SOAP schema definitions.

Business Content Alerting Handler

The Business Content Handler generates alerts based on content that is found in SOAP requests, responses, or failure messages. The content is found in the message by applying an XPath expression.

Security Handlers

Security Handlers are used to provide both message-level and transport-level security for intermediary services.

Using Intermediary Services

This chapter explains how to manage the life cycle of intermediary services. It begins with an overview of intermediary services and then describes how to edit, deploy, view, and remove an intermediary service when using the WSM Intermediary.

Overview

An intermediary service is created for each Web service that you want to manage. The **Broker Configurator** creates and manages the life cycle of an intermediary service. Requests for managed Web services are sent to the intermediary service and then forwarded (dispatched) to the actual service's endpoints. Intermediary services can be created for both SOAP/HTTP and XML/HTTP Web services.



SOAP with attachments services is supported only if a WSDL is provided that describes the service.

Intermediary services are used to manage Web services when you want to do the following:

- Separate the management of Web services from the services' implementation.

Viewing Intermediary Service Details

The Service Details screen lets you view the details of an intermediary service. The details include the intermediary service definition and endpoint, performance data, the Web service's endpoints, and features (handlers) configuration.

To view an intermediary service's details, follow these steps:

- 1 From the Intermediary Services screen, find the intermediary service that you want to view.
- 2 From the Name column, click the intermediary service's name. The Service Details screen opens. The intermediary service's details are listed in different sections. The Features section displays which handlers are enabled and their current configuration settings.

Performance Metrics

The Service Detail screen displays a subset of the performance metrics that are collected for an intermediary service. The metrics include the Average Response Time, Total Requests, Successes, and Failures. These metrics provide a general view of how an intermediary service is performing. The full set of performance metrics is displayed in the SOA Manager server when the intermediary service is managed as part of a business service.

Undeploying an Intermediary Service

An intermediary service that is undeployed is inactive, but is not removed from the Intermediary Service list. The intermediary service is not available for requests until it is deployed. You can configure an intermediary service that is undeployed, but you cannot view any of its management data.



Any Web service management data that has been collected is lost when an intermediary service is undeployed.

To undeploy an intermediary service, follow these steps:

- 1 From the Intermediary Service list, find the intermediary service that you want to undeploy.
- 2 From the Action column, click the **undeploy** link. The status of the service changes from Operational to Inactive.

Deploying an Intermediary Service

A deployed intermediary service can receive service requests and is considered operational. An intermediary service that is operational collects management data about the Web service that it is managing. An intermediary service is automatically deployed when the intermediary service is created.

To deploy an intermediary service, follow these steps:

- 1 From the Intermediary Service list, find the intermediary service you want to deploy.
- 2 From the Action column, click the **deploy** link. The Status field updates from Inactive to Operational.
- 3 Verify that the service is operational by clicking the intermediary service WSDL endpoint listed in the Service Interface (WSDL) column. The WSDL for the service is displayed.

Editing an Intermediary Service

You can edit an intermediary service at any time. Typically an intermediary service is edited to enable/disable different handlers depending on the type of manageability that is required for the Web service.

To edit an intermediary service, follow these steps:

- 1 From the Intermediary Service list, find the intermediary service that you want to edit.
- 2 From the Action column, click the **edit** link. The Edit Service screen opens.
- 3 From the Edit Service screen, edit the intermediary service using the fields provided. The handler configuration options are detailed in Chapter 11 “Configuring Handlers”. See Chapter 15 “Using the Intermediary’s Security Features” for detailed instructions if you want to secure communication with the intermediary service.
- 4 Click **Save**. The Intermediary Service screen opens and the intermediary service is automatically deployed. The deployment is complete when the status changes to *Operational*.

Changing an Intermediary Service’s Version

Each intermediary service has a description which includes a name that identifies the service in the Broker Configurator and a version number. An intermediary service name is automatically generated when the intermediary service is created. You cannot change the intermediary service’s name, but you can change the version number.

To change an intermediary service’s version, follow these steps:

- 1 From the Configurator's main toolbar, click **List Services**. The Broker Service screen opens.
- 2 From the Action column, click the **edit** link for the intermediary service. The Edit Service screen opens.
- 3 From the Service section, select the Version field and enter a version number for the intermediary service.
- 4 At the bottom of the screen, click **Save**. The Broker Service screen opens and the intermediary service is automatically deployed. The deployment is complete when the status changes to *Operational*.

Configuring an Intermediary Service’s HTTP Path

An intermediary service’s HTTP Path is the path that will be used by a client to invoke the managed Web service. For example, if the intermediary agent is installed on "MyHost.com" and the default intermediary port is used, the URL to the Web service would be:

```
http://MyHost.com:9032/<http_path_value>
```

A path value is automatically generated when the intermediary service is created. Changing the HTTP path of an intermediary service is useful when multiple intermediary services, with different configurations, are created for the same service or when a specific URL strategy is used by your organization.

To configure a service's HTTP Path, follow these steps:

- 1 From the Configurator's main toolbar, click **List Services**. The Broker Service screen opens.
- 2 From the Action column, click the **edit** link for the intermediary service. The Edit Service screen opens.
- 3 From the Inbound Transport section, select the HTTP Path field and enter a path. The path must consist of alpha-numeric characters and begin with a forward slash (/).
- 4 At the bottom of the screen, click **Save**. The Broker Service screen opens and the intermediary service is automatically deployed. The deployment is complete when the status changes to *operational*.

Removing an Intermediary Service

When an intermediary service is removed, it is deleted from the Intermediary Service list. In addition, the service definition (WSDL) for the intermediary service is deleted from the `<install_dir>\conf\broker` directory.

To remove an intermediary service, follow these steps:

- 1 From the Intermediary Service list, find the intermediary service that you want to remove.
- 2 From the Action column, click the **remove** link. A confirmation dialog box opens and asks you to confirm the removal of the intermediary service.
- 3 Click **OK** to remove the intermediary service.

Enabling Protocol Switching at the Intermediary

You can configure the intermediary to help in communication between service consumers and service providers that follow different protocols. The intermediary provides support to switch messages between JMS and HTTP/HTTPS protocols. The intermediary, before sending the request to an endpoint, transforms the request to a format supported by the protocol at the endpoint. The intermediary, after receiving a response from the endpoint, transforms the response back to the format supported by the protocol at the client that sent the request.

Prerequisites

To use JMS as the transport model, follow these steps to make sure that the prerequisites are satisfied:

- 1 Install the JMS server separately

- 2 Configure the destinations for both inbound and outbound service messages.
NOTE: The intermediary currently supports publish/subscribe and point to point messaging model. You can also enforce policies by using this feature in a JMS transport model.
- 3 Copy the JMS client jar files corresponding to the provider to the <install_dir>/lib/ext directory on the intermediary.
- 4 Make sure that the WSDL for the intermediary Web service contains the JMS binding information.

SOA Manager currently supports the following JMS service providers:

- WebLogic 8.1
- JBoss 4.0.4
- Tibco 4.4.0
- Sonic 7.0, 7.5

During protocol switching, the following sequence of events occurs at the intermediary:

- 1 The service consumer sends a SOAP or XML message over HTTP or JMS to the intermediary.
- 2 The intermediary receives the message and invokes the transport and XML handlers to transform the message based on the protocol supported at the endpoint.
- 3 For a two-way protocol switch interaction, the intermediary creates a temporary destination to receive a response from the endpoint.
- 4 After receiving the response from the endpoint, the intermediary invokes handlers to process the message back to the protocol supported by the service consumer.
- 5 After the message is processed it is converted to the protocol supported by the service consumer.

The intermediary supports two-way protocol switching for the following scenarios:

- Communication between different JMS service providers
- Communication from HTTP to JMS
- Communication from JMS to HTTP

Refer to the following scenarios to configure the intermediary to enable protocol switching.

Enabling JMS-to-JMS-Two-Way Protocol Switching

For a JMS-to-JMS two-way protocol communication, you must make sure that the following prerequisites are satisfied:

Configure three queues, one each for the following:

- Intermediary service
- Endpoint
- Client (to receive messages)

Make sure that the endpoint information present in the WSDL (that contains the queue information) is similar to the setup information of the queue.

To enable JMS-to-JMS protocol switching, follow these steps:

- 1 Start SOA Manager Intermediary and log in to the Broker Configurator.
- 2 Click **Add New Broker Web Service**. The Step1: Import WSDL screen of the Add New Broker Service page opens.
- 3 Type the modified WSDL that contains information about the JMS endpoint in the **Browse local WSDL file:** box.
- 4 Click **Next**. The Step 2: Configure Endpoints screen of the Add New Broker Service page opens.
- 5 Verify that the parameters are similar to what you specified in the WSDL.
- 6 Click **Next**. The Step 3: Configure Broker Service screen of the Add New Broker Service page opens.
- 7 Select **JMS Transport** from the Inbound Transport table. This displays the additional parameters that you must specify for JMS transport.
- 8 Specify the following details in the JMS Transport section. The examples shown in parenthesis for each of the options are specific to a WebLogic server used for inbound transport:
 - Destination Style: Specify queue for the destination type for the JMS transport model. JMS Topic type destination style is not supported currently.
 - Vendor URI: Specify the URL of the vendor that provides JNDI (<http://bea.com>).
 - Initial Context Factory: Specify the name of the JNDI context factory (`weblogic.jndi.WLInitialContextFactory`).
 - JNDI Provider Url: Specify the URL of the JNDI server (`t3://soamw2.ind.hp.com:7001`).
 - JNDI Connection Factory Name: Specify the JNDI lookup name for the connection factory (`weblogic.jms.ConnectionFactory`).
 - JNDI Destination Name: Specify the name of the JNDI destination name (`weblogic.wsee.WLSJbossInboundQueue`).
- 9 Click **Finish**. This enables the JMS-to-JMS two-way protocol switching at the intermediary.

Enabling HTTP-to-JMS-One-Way Protocol Switching

To enable HTTP-to-JMS protocol switching, follow these steps:

- 1 Start SOA Manager Intermediary and log in to the Broker Configurator.
- 2 Click **Add New Broker Web Service**. The Step1: Import WSDL screen of the Add New Broker Service page opens.
- 3 Type the modified WSDL that contains the JMS endpoint information in the **Browse local WSDL file:** box.

- 4 Click **Next**. The Step 2: Configure Endpoints screen of the Add New Broker Service page opens.
- 5 Verify that the parameters are similar to what you specified in the WSDL.
- 6 Click **Next**. The Step 3: Configure Broker Service screen of the Add New Broker Service page opens.
- 7 Select **Http Transport** from the Inbound Transport table.
- 8 Click **Finish**. This enables the HTTP-to-JMS one-way protocol switching at the intermediary.

Enabling JMS-to-HTTP-One-Way Protocol Switching

To enable JMS-to-HTTP protocol switching, follow these steps:

- 1 Start SOA Manager Intermediary and log in to the Broker Configurator.
- 2 Click **Add New Broker Web Service**. The Step1: Import WSDL screen of the Add New Broker Service page opens.
- 3 Type the modified WSDL that contains the HTTP endpoint information in the **Browse local WSDL file:** box.
- 4 Click **Next**. The Step 2: Configure Endpoints screen of the Add New Broker Service page opens.
- 5 Verify that the parameters are similar to what you specified in the WSDL.
- 6 Click **Next**. The Step 3: Configure Broker Service screen of the Add New Broker Service page opens.
- 7 Select **JMS Transport** from the Inbound Transport table.
- 8 Click **Finish**. This enables the JMS-to-HTTP one-way protocol switching at the intermediary.

Configuring Handlers

This chapter lists the management handlers available for intermediary services in alphabetical order. Each entry includes a description of each handler as well as the handler's fields. Where applicable, example entries for the fields are provided. You can refer to the entries in the chapter when you are editing or creating an intermediary service.



When using custom intermediary services, handler ordering is important, because handlers attach information to the executing operation for other handlers to find and use. Some handlers, like encryption/decryption handlers, also modify the message as it passes along the chain.

Audit Handler

The Audit Handler collects trace information on messages sent to Web services. The auditing feature can collect a message's SOAP payload. The information collected is sent to HP SOA Manager and is stored in a database. The HP SOA Manager web interface is used to query the database to retrieve audit information. Any management application can be extended to access the audit data. For more information on using the SOA Manager Auditing feature, see the "Using Auditing" chapter in the *SOA Manager User Guide*.

Fields

- **Include detailed traces:** Captures profile data. The outcome of a Web service invocation as it passes through each handler in the handler chain for an intermediary service.
- **Payload Option:** Type of message payloads that should be logged.
- **Payload Filter:** Criteria to determine which message payloads should be logged.
- **Expression:** An XPath expression for determining which message payloads should be logged. This field is used if content-based payload logging is configured. This field is only available for custom intermediary services.

- **Namespaces:** Namespaces that are used in the expression field must be declared using the namespace prefix and the namespace URI. This field is only available for custom intermediary services.

Configuring the Audit Publisher

The Audit Publisher is an Intermediary component that publishes audit information that is collected by the audit handler.

There are two configuration options for the Audit publisher: `interval` and `threshold`. The Audit publisher sends trace messages using the value for whichever configuration option is reached first.

- `interval`– The entry sets the amount of time in milliseconds between publishing audit information.
- `threshold` – The threshold sets the number of messages that are published. When the number of messages reaches this threshold, the messages are published.

To configure the audit publisher, follow these steps:

- 1 Stop the Intermediary if it is currently started.
- 2 Use a text editor to open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Edit the audit interval and threshold values. For example:

```
<entry name="com.hp.audit.publisher.interval">100000</entry>
<entry name="com.hp.audit.publisher.threshold">10</entry>
```
- 4 Save and close the properties file.
- 5 Restart the Intermediary.

Business Metric Alerts Handler

Business content alerting lets you define a business metric for specific content that is found in the SOAP request and response message for a service (for example, an order is placed with a total that is greater than \$25,000.00). When the business metric value is found, an alert is generated and sent to the Network Services server which notifies alert recipients (email, HP SOA Manager web interface console, and so on). For more information on SOA Manager business content alerting feature, see the “Using Alert Notification” chapter in the *SOA Manager Administrator Guide*.



Business content alerts are processed by the Network Services server and sent to any recipients configured to receive business content alerts. Recipients for business content alerts are viewed in the HP SOA Manager web interface.

Fields

- **Name:** Enter a user friendly name to identify the alert. (for example, HPQ Alert)
- **Operation:** Enter an operation in the service that contains the business content you want to monitor. The XPath expression is applied to the operation. (for example, getInfo)
- **Alert applies to:** Select when you want the intermediary to search for the operation. You can select to search during requests or responses.
- **Expression:** Enter an XPath expression which selects the business content from the operation. For example, `//tns1:InfoRequest/tns1:symbol/text()`. This expression traverses the SOAP message for the InfoRequest node and selects the text found for the symbol child node.
- **Message:** A user friendly message that is sent with the alert. (for example, A \${name} alert has occurred)
- **Dynamic Properties:** A dynamic variable defined within the message. The Name field corresponds to the variable name. The XPath field corresponds to an XPath expression used to update the variable. For example, **Name:** name **Xpath:** `//s:Envelope/s:Body/t:InfoRequest/t:symbol/text()`. The XPath expression specified here is evaluated on the business content selected by the expression provided in the expression field.
- **Namespace Prefixes:** Any namespace prefixes that appear in the XPath expression (for example, **prefix:** tns1 **URI:** `http://wsm.hp.com/Finance/Request`).

Generic SOAP Contract Handler

The Generic SOAP Contract Handler detects the operation from a request. It can be used to replace the Soap Contract Handler. The handler is commonly used for SOAP services that do not have a WSDL. The handler generates a simple WSDL and does not perform any runtime checks. The handler must be used after decryption and before any handler that requires the operation. The handler only supports a single portType and binding. Operations are set as the runtime soap payload element. When using this handler, no WSDL is required in the Intermediary deployment unit.

Fields

- **namespace:** The target namespace for the generated WSDL
- **name:** The name of the WSDL
- **portType:** The name of the generated portType
- **binding:** The name of the generated binding

HTTP Pass-Through Transport Header Handler

The HTTP Pass-Through Transport Header Handler copies transport headers from either side of the intermediary (request or response). This handler must be used in conjunction with, and before, the Dispatch Handler.

The headers are configured in `<install_dir>/conf/broker/mipServer.xml`. There is a property for both a request (SOAPAction is the default) and a response (no default):

```
<entry
  name="com.hp.transport.headers.pass.request">SOAPAction
</entry>
<entry name="com.hp.transport.headers.pass.response"></entry>
```



This handler copies JMS properties when the transport used is JMS.

When you set a request property, the handler copies properties in the request message from the broker to the service. For a response property, the handler copies properties from the service to the client.

Invocation Handler

The Invocation Handler marshals XML to Java using JAXB and is used to invoke a Java class. The invocation handler is only available for custom intermediary services.

Fields

- **Classname:** The name of the Java class to be invoked
- **Packages:** The package of the Java class

Log Handler

An intermediary service's logging feature lets you indicate whether or not you want faults to be logged to the Intermediary's log file as well as the console. When enabled, log messages are included in the log file and the console. The intermediary's log file is named `broker.log` and is located at `<install_dir>/log`.

Fields

Category: This field is only available for custom intermediary services. This field lets you select a specific log category where log messages are sent. This field is optional.

Schema Validation Handler

Schema validation ensures that SOAP requests conform to a Web service's WSDL. If the schema validation feature is enabled, requests that do not strictly conform to the WSDL are not dispatched to the service endpoint and an HTTP 500 error is returned by the Intermediary. If the schema validation feature is disabled, SOAP requests are not validated before being dispatched to the service endpoint. Depending on the level of nonconformity, a SOAP request may or may not be successful.



Schema validation is only applied to services implemented using document literal SOAP operations.

Service Protection Handler

This handler limits access to endpoints being managed using a policy enforcement point. You can use this handler to specify the number of service requests that the policy enforcement point can accept. After the limit specified for the number of service requests that the policy enforcement point can accept is exceeded, this handler rejects the subsequent service request messages by sending a SOAP fault which prevents the managed endpoint from crashing or denying service requests.

Fields

- Number of requests to be accepted by the policy enforcement point by the second, minute, hour, day, week, or month.
- Time zone to be used.

Content Detection Handler

You can use this handler to verify the presence of content in a SOAP message at the Intermediary. You can check for the presence of the content in the header or the body of the message using this handler. This handler allows you to specify an XPath expression to extract the content from the message that you want to detect. Based on the presence or absence of the XPath expression in the extracted content, the handler either rejects the message from the client and returns a fault code to the client or forwards the message to the endpoint.

Fields

- XPath expression: specifies the XPath expression that you want to use to extract content.
- Namespace prefix: specifies the namespace prefix to be included with the XPath expression.

- Fault code: specifies the fault code to be generated for the specified fault type.
- Fault code (namespace URI): specifies the URI to the schema that you want to use in the event of a specified fault getting generated.
- Fault message: specifies the fault message to be generated for the specified fault type.

Scheduled Availability Handler

You can use this handler to allow or deny access to a service based on the scheduled availability time period specified for that service. The Intermediary uses this handler to allow or deny access to a service at that specific time. If the service is specified to be available at the time specified, the Intermediary forwards the message from the client to the endpoint. If the service is specified to be unavailable, the handler rejects the message and sends a SOAP fault to the client.

Fields

- Service availability: specifies if the service must be available or not at the specified time.
- Hours of operation: specifies the time for the availability or non availability of the service. You can select one of the following options:
 - Days (Non recurring): specifies that the handler is applicable on a non recurring basis on all days.
 - Workdays (recurring): specifies that the handler is applicable on a recurring basis on workdays.
 - Weekends (recurring): specifies that the handler is applicable on a recurring basis on weekends.
- Time zone: specifies the time zone to be used.
- Start time: specifies the start time of the service availability or non availability.
- End time: specifies the end time of the service availability or non availability
- Fault code: specifies the fault code for the fault type generated.
- Fault message: specifies the fault message to be generated for the specified fault type.

Security Auditing

The Security Audit Handler is used to collect security trace information (used for non-repudiation, and so on) and sends the payload to a security provider.

Field

- **Payload Option:** Use this field if you want to constrain the type of message payloads that should be logged. Only payloads for the option selected are captured and sent to the security provider.

Configuring Security Auditing

When using the Security Audit Handler, you must configure the security provider where security trace information will be sent.

Service Security Inbound Handler

The Service Security Inbound Handler performs authorization using the principal and credentials associated with an operation. This handler is used in conjunction with, and must come after the WS Security Message Processing Inbound handler. This handler must come before any handler that needs to be protected.

SOAP Contract Handler

The SOAP Contract Handler detects the operation from a request. It is a required handler that must be in every SOAP service. The handler must be used after decryption but before any handler that requires the operation. The handler can only be disabled, or the ordering changed, when using custom intermediary services.



The Generic SOAP Contract Handler can be used to replace the Soap Contract Handler for SOAP services that do not have a WSDL. For more information, see the “Generic SOAP Contract Handler” section above.

SOAP Dispatch Handler

The SOAP Dispatch Handler is used to dispatch a request to the Intermediary's Dispatcher component, which is responsible for forwarding a request to a Web service's endpoint. The handler must be last in the handler chain.

SOAP Monitoring Handler

The SOAP Monitoring Handler is used to decide if a service response is a success or failure. The handler must be used after any handler that requires the outcome and before any handler that might modify the outcome. Matches are based on SOAP fault codes.

Fields

- **Match:** Use the appropriate option to indicate whether the match means a success or a failure or all faults to be considered as failure.
- **Fault Codes:** Use this field to configure a list of codes to match. You must also include namespace for the code. For example, **Code:** `Server` **Namespace:** `http://schemas.xmlsoap.org/soap/envelope/`. When the option chosen for match is “All faults as failure”, fault codes need not be specified.

WS Security Outbound Handler

The Ws Security Outbound Handler provides support for WS-security on outbound messages (from the Intermediary to a consumer). This includes user name/password, signing, and encryption.. The handler removes the security headers if they already exist in the message. The handler then adds a new security header to the outbound message. Support for Security Assertion Markup Language (SAML) prevents existing message security headers from getting removed.

Fields

- **actor:** This specifies the target for which the header is configured. You can specify this parameter in the `<install_dir>\conf\broker\mipServer.xml` file. `<install_dir>` represents the directory in which you have installed SOA Manager Broker. If you specify this parameter, the security header for the specified actor (present in the inbound message) is not removed from the outbound message.
- **outbound.remove:** Use this parameter to specify if the security header must be removed or retained in the outbound message. You can set the value of this parameter to `true` (indicating that the security header must be removed) or `false` (to indicate that the security header must be retained) to indicate your choice. You can specify this parameter in the `<install_dir>\conf\broker\mipServer.xml` file. `<install_dir>` represents the directory in which you have installed SOA Manager Broker.
- **authMethod:** The outbound authentication method being used. Valid entries are `Cert`, `UsernameToken`, and `SSOToken`.
- **outMsgEncrypted:** Whether or not the message needs to be decrypted. This field is only relevant when using the `Cert` method. Valid entries are `true` and `false`.
- **outMsgUsername:** Username to be sent with the outgoing message

- **outMsgPassword:** Password to be sent with the outgoing message
- **outMsgSecurityProvider:** Not used
- **outMsgSignBeforeEncrypt:** Whether or not the message will be signed before it is encrypted. If set to true, the response is signed and then encrypted. If set to false, the response is encrypted and then signed. This field is only relevant when using the `Cert` method and when `outMsgEncrypted` and `outMsgSigned` are enabled.
- **outMsgSigned:** Whether or not the message has a signature that needs to be validated. This field is only relevant when using the `Cert` method. Valid entries are `true` and `false`.
- **relayRouter:** The alias to find the recipient certificate from the keystore.
- **securityProvider:** Not used

WS Security Message Processing Inbound Handler

The WS Security Message Processing Inbound handler provides support for WS-security signing and encryption. This handler is used in conjunction with the Service Security Inbound Handler and must come before any handler that reads the request body.

Both handlers are required because the authorization cannot be performed until the operation being invoked is known, but the handler that detects the operation requires the request to be decrypted first. Decryption and credential extraction is first completed using the WS Security Message Processing Inbound handler. The Soap Contract Handler detects the operation, and then the Service Security Inbound Handler uses the credentials and operation to perform authorization. . SOA Manager Broker removes the security header in message before passing the message to the end point. As a result, additional information contained in the headers does not reach the end point. Support for Security Assertion Markup Language (SAML) prevents removal of unprocessed security headers in the message before forwarding the message to the end point.

Fields

- **actor:** This specifies the target for which the header is configured. You can specify this parameter in the `<install_dir>\conf\broker\mipServer.xml` file. `<install_dir>` represents the directory in which you have installed SOA Manager Broker. If you specify this parameter, SOA Manager Broker processes the header specific to the actor specified and removes this header from the message before passing the rest of the headers to the end point. If you do not specify this parameter, the Broker uses the header that does not contain any specification for this parameter.
- **inbound.remove:** Use this parameter to specify if the processed security header must be removed or retained in the inbound message. You can set the value of this parameter to `true` (indicating that the security header must be removed) or `false` (to indicate that the security header must be retained) to indicate your choice. You can specify this parameter in the `<install_dir>\conf\broker\mipServer.xml` file. `<install_dir>` represents the directory in which you have installed SOA Manager Broker.

- **inMsgAuthMethod:** The inbound authentication method being used. Valid entries are `Cert`, `UsernameToken`, and `SSOToken`.
- **inMsgEncrypted:** Whether or not the message needs to be decrypted. This field is only relevant when using the `Cert` method. Valid entries are `true` and `false`.
- **inMsgResponseSecurity:** Whether or not the response is to be secured. If set to `false`, the response will not be signed or encrypted. If set to `true`, `inMsgEncrypted` and `inMsgSigned` will apply and the response will be signed and/or encrypted.
- **inMsgSSOEnable:** Not used
- **inMsgSignBeforeEncrypt:** Whether or not the message will be signed before it is encrypted. If set to `true`, the response is signed and then encrypted. If set to `false`, the response is encrypted and then signed. This field is only relevant when using the `Cert` method and when the `inMsgEncrypted` and `inMsgSigned` are enabled.
- **inMsgSigned:** Whether or not the message has a signature that needs to be validated. This field is only relevant when using the `Cert` method. Valid entries are `true` and `false`.

XML Contract Handler

The XML Contract Handler detects the operation from a request. It is a required handler that must be in every XML service. The handler must be used after decryption but before any handler that requires the operation. The handler can only be disabled, or the ordering changed, when using custom intermediary services for XML services.

XML Dispatch Handler

The XML Dispatch Handler is used to dispatch a request to the intermediary's Dispatcher component, which is responsible for forwarding a request to a Web service's endpoint. The handler is used for custom intermediary services for XML services. The handler must be last in the handler chain

XPath Monitoring

The XPath Monitoring handler is used to decide if a service response is a success or failure. The handler must be used after any handler that requires the outcome and before any handler that might modify the outcome. The handler is used for custom intermediary services for XML services. Matches are based on XPath expressions.

Fields

- **Match:** Use the options to indicate whether the match means a success or a failure.
- **XPath:** Use this field to enter an XPath expression used to match.

- **Namespace Mapping:** Enter any namespace prefixes that appear in the XPath expression (**prefix:** tns1 **Namespace:** http://wsm.hp.com).

XSLT Handler

The XSLT Handler runs an XSLT template on the request or response messages. A different template can be assigned for the request and response. The templates must be included in the intermediary service JAR file in order to be loaded by the Intermediary's classloader.

Fields

- **requestTemplate:** The name of the XSLT template to be applied to a request message.
- **responseTemplate:** The name of the XSLT template to be applied to a response message.

Classifier Handler

The classifier handler forwards the requests to a specific endpoint configured to the handler.

Fields

- **Enter New Classifier:** The classifier name.
- **Expression:** An XPath expression for the endpoint of the classifier.
- **Context:** Specifies if the classifier must be used for message or transport.
- **Namespaces:** Any namespaces that you might want to specify.

Using Custom Intermediary Services

This chapter explains how to use custom intermediary services. The instructions include tasks for creating and configuring a custom intermediary service definition as well as adding handlers to a custom intermediary service. In most situations, a simple intermediary service provides enough functionality to manage a Web service. However, there are situations when a custom intermediary service can be used to allow greater control of the service definition and access to custom WSM functionality.

Overview

Custom intermediary services are similar to simple intermediary services in that they act as proxies to a Web service endpoint and provide WSM capabilities in the form of handlers that are organized in a handler chain. Any handler available for a simple intermediary service is also available for a custom intermediary service. Simple intermediary services use a predefined set of handlers, while custom intermediary services are boundless. The handler chain can be customized to include a broad range of handlers (including custom handlers). The ordering of the handlers in the handler chain can be configured.

The benefits of using a custom intermediary service include the following:

- Maximum control when assigning handlers and creating the handler chain
- Support for a broad range of handlers
- Support for custom handlers
- Reuse of handlers within a handler chain (that is, multiple business metric handlers)

Convert a Simple Intermediary Service

Custom intermediary services are created by first creating a simple intermediary service (see Chapter 10) and then converting the simple intermediary service to a custom intermediary service. You can convert SOAP/HTTP and XML/HTTP simple services to custom services.

To convert a simple intermediary service to a custom intermediary service, follow these steps:

- 1 From the Intermediary Services list, find the intermediary service that you want to convert.
- 2 From the Action column, click **edit**. The Edit Service screen opens.
- 3 Click **Convert**. The Edit Custom Service screen opens and lists the handlers for the custom service. Any handlers that were configured for the simple intermediary service are also configured for the custom service. Several default handlers, which were part of the simple intermediary service but not previously visible, are listed.
- 4 Click **Save**. The Intermediary Service screen opens and the intermediary service is automatically deployed. The deployment is complete when the status changes to `Operational`. The `Style` field indicates that the intermediary service is `Custom`.

Adding Handlers

Using custom intermediary services provides greater control when adding handlers for an intermediary service. Handlers are assigned to a custom intermediary service using the Broker Configurator's Edit Custom Service screen. The available handlers are detailed in the "Configuring Handlers" chapter.

To add handlers to a custom intermediary service, follow these steps:

- 1 From the Service list, find the custom intermediary service that you want to edit. The `Style` field indicates that the intermediary service is `Custom`.
- 2 From the Action column, click **edit**. The Edit Custom Service screen opens and displays the handlers currently assigned to the intermediary service.
- 3 Use the Add a new handler drop-down list to add a handler. The handler is added to the list of handlers. Repeat this step to add additional handlers. See Chapter 11 "Configuring Handlers" for a detailed description of each handler.
- 4 Click **Save**. The Intermediary Services screen opens and the intermediary service is automatically deployed. The deployment is complete when the status changes to `Operational`.

Adding Custom Handlers

Custom intermediary services let you add your own custom handlers to an intermediary service's handler chain. To add a custom handler, you must first create the custom intermediary service and then edit the service's definition file located in the intermediary service jar file.

To add a custom handler, follow these steps:


- 1 Uncompress `<install_dir>\conf\broker\<intermediary_service_name>.jar`.
- 2 Using a text (or XML) editor, open `service.xml`.

- 3 Under the `<service>` element, add a `<handler>` element and include the fully qualified class name. For example:

```
<handler classname="com.company.HandlerClass" />
```

- 4 If the handler requires any properties, add them as elements under the handler class. For example:

```
<handler classname="com.company.HandlerClass" >
  <property1>foo</property1>
  <property2>
    <property name="foo" value="bar" />
  </property2>
  <ns1:property3>foo</ns1:property3>
</handler>
```

 If the property uses a namespace, you must declare the namespace as an attribute of the `<service>` element before using the namespace (for example, `xmlns:ns1="com.company"`).

- 5 Save and close `service.xml`.
- 6 Place the custom handler class and any dependent classes in the same directory as `service.xml`.
- 7 Re-jar the intermediary service including the custom handler class and any dependent classes.
- 8 Place the jar in `<install_dir>\conf\broker\`. The intermediary service is automatically deployed. You can use the Broker Configurator to verify that the jar has been deployed. The intermediary service is listed on the Service List and its status is `Operational`.

Defining Service Providers for Custom Web Services

The intermediary allows you to route a SOAP request to an appropriate endpoint based on the context or content of the message. Intermediary can be configured to do this routing as follows:

- When you create an intermediary Web service, if the WSDL used contains multiple end points, the intermediary lets you classify these endpoints.
- The definition for this classification is provided as properties of the Classifier handler. The following properties must be specified:
 - XPath expression – The XPath expression that should be evaluated on the incoming request
 - Context – This field indicates whether the expression should be evaluated on the transport context or content of the message. When transport is selected, the XPath expression is evaluated using XML in the following format:
 - `<header>`
 - `<header-name1>value</header-name1>`
 - `<header-name1>value</header-name1>`

– </header>

The variables <header-name1> and <header-name2> represent the HTTP headers when HTTP transport is used or JMS headers when JMS transport is used. When HTTP is used as transport, the XML file also contains the following details:

– <TCP_HOST>source_host</TCP_HOST>
– <TCP_PORT>source_port</TCP_PORT>

When a request is sent to an intermediary Web service, based on the content or context of the message, the intermediary can route the request to the appropriate endpoint. The definition for this classification is provided by using classification handlers. An incoming request can be classified.

If you enable XSLT transformation, the intermediary transforms the classified message. See the XSLT Transformation section for additional information about XSLT transformation. The intermediary then forwards the request based on the specifications in the classifier to the corresponding endpoint. You must perform the steps in the following section to enable content-based routing. See *Enabling Content-based Routing for Intermediary Web Services* for information on configuring content-based routing for intermediary Web services. Refer to the following scenario for additional information.

Consider a banking Web service where you must administer requests from customers belonging to the following classifications:

- High loan request (\$25,000 and above)
- Medium loan request (up to \$25,000)

The banking Web service must forward requests from these two types of loan requests automatically to the corresponding endpoints that handle specific types of loan requests. For example, according to the bank loan guidelines, a medium loan request does not need approval from the higher authorities in the bank. A high loan request needs approval from the manager and senior management staff. For this scenario, you can configure the banking Web service to automatically forward loan requests to the corresponding endpoints based on the loan amount requested by the customer. You can perform this configuration using the content-based routing feature that SOA Manager provides.



Content Based Routing feature is supported only for XML service types.

Enabling Content-based Routing

To enable content-based routing for the example scenario, follow these steps:

- 1 Start SOA Manager Intermediary and log in to the Broker Configurator.
- 2 Click **Add New Intermediary Web Service**. The Step1: Import WSDL screen of the Add New Broker Service page opens.
- 3 Import the desired WSDL in the **Browse local WSDL file:** box.
- 4 Click **Next**. The Step 2: Configure Endpoints screen of the Add New Broker Service page opens.

- 5 Type **high_loan** and **medium_loan** in the Classifier boxes available for each endpoint. Special characters such as %, &, and +, and so on are not supported for classifier names.
- 6 Click **Next**. The Step 3: Configure Broker Service screen of the Add New Broker Service page opens.
- 7 Change the name of the service and HTTP Path if a similar service is already deployed.
- 8 Select **Classifier Handler** from the **Features** section. The Classifier Handler section opens.
- 9 Type `high_loan` and click **Add** in the **Enter New Classifier** box.
- 10 Type `medium_loan` and click **Add** in the **Enter New Classifier** box
- 11 Select **high_loan** from the **Classifier** drop-down list and provide the following details:
 - a Specify an XPath expression for the endpoint of the classifier in the **Expression** box
 - b Select **Message** from the **Context** option
 - c Type the Prefix and the URIs for the Namespaces in the corresponding boxes
 - d Click **Save**.
- 12 Repeat steps a through d for the `medium_loan` classifier and click **Save**.
- 13 Click **Save**.
- 14 Click **Finish**.

Getting Started

This chapter provides detailed instructions for starting and configuring the WSM Broker. The WSM Broker is installed as part of the SOA Manager installation. Before beginning the instructions in this chapter, make sure you have installed SOA Manager following all the instructions in the *SOA Manager Installation Guide*. The directory where you installed SOA Manager is referred to as *<install_dir>* throughout these instructions.

This chapter also provides instructions for configuring the Broker using the Broker's configuration files. The chapter covers common configuration changes and does not include every configuration option. The Broker's configuration files are located in the *<install_dir>\conf\broker* directory of the distribution. The configuration files can be edited with a text editor. In addition, several of the configuration options discussed here can be set using the Broker Configurator.

Starting the WSM Broker

A script for both Windows and UNIX is provided to start the Broker. The script is located in *<install_dir>/bin/win32* and *<install_dir>/bin/unix*, respectively. Windows users can choose to create product icons during installation. If you accepted the default program group during installation, you can start the Broker by clicking **Start | Program Files | HP Software | SOA Manager 2.52 | Broker**.



During the SOA Manager installation, you had the option to install the WSM Broker as a Windows Service. If you chose this option, the WSM Broker is already running. Attempting to start WSM Broker again causes an error.

To start the WSM Broker:

- 1 Open a command prompt.
- 2 Depending on your platform, change directories to *<install_dir>\bin\win32* or *<install_dir>\bin\unix*.
- 3 Run the “broker” startup script. The console outputs log messages as the broker starts. The broker has started when you see the message:

```
MIP Server startup completed in # seconds.
```



If you selected to install the WSM Broker as a Windows service, the Broker may already be running. If you attempt to start the Broker again, an error message is displayed.

Stopping the WSM Broker

The WSM Broker can be stopped using the stop process methods that are appropriate for the host operating system.

Windows

Switch to the command window where the server process is running and type `Ctrl+c`. Then type `y` to terminate the process.

If the WSM Broker is running as a Windows service, the service must be stopped. To stop a Windows service, open the Control Panel and select **Administrative Tools**. From the Administrative Tools screen, select **Services**. From the Services screen, right-click the WSM Broker service and select **Stop**.

UNIX

When using Linux or HP-UX, open a terminal window and issue the following command:

```
ps -ef | grep java
```

The command lists all current Java processes, including the process number. Find the WSM Broker process and issue the `kill` command to stop the process. For example:

```
kill <process number>
```

Starting the Broker Configurator Console

Typical interaction with the Broker is through its console. The console is a Web application that runs on port 9032. To change the default port, see the “HTTP Settings” section below.

To start the Broker Configurator:

- 1 Start the Broker as described above.
- 2 Open a Browser.
- 3 Enter the following URL and substitute *<host>* with the host name where the Broker Agent is running:

```
http://<host>:9032/console
```

- 4 The login screen already contains default credentials: `admin` is the username and `password` is the password.

- 5 Click **Login**. The Brokered Services screen displays.



The WSM Broker version (including installed patches) is located above the copyright statement at the bottom of each page.

Installing the Broker as a Windows Service

If you choose not to install the Broker as a Windows service during the installation, a batch script is provided that installs the Broker as a Windows service. This allows the Broker to automatically start whenever Windows is started. The script can also be used to remove the Broker from being a Windows service.

To install the Broker to run as a Win 32 Service:

- 1 Open a command window.
- 2 Change directories to `<install_dir>\bin\win32\services`.
- 3 Run `service-manager.bat` and specify the following arguments:

```
service-manager.bat -install broker <install_dir>
```

The service has been successfully installed when the following message is outputted to the console:

```
Service "HP SOA Manager v2.52 Broker" installed.
```



The script configures the HP SOA Manager 2.52 broker service to automatically start the next time Windows is started. You must use the Windows Computer Management Console to change this behavior.

To remove the service, run the `service-manager` script and specify `-remove`. For example,

```
service-manager.bat -remove broker
```

Configuring HTTP Settings

The WSM Broker contains both an HTTP server and an HTTP client. The server is used to accept HTTP requests for Web services and is also used to interact with the Broker Configurator. The HTTP client is used to communicate with HTTP-based servers that are hosting Web services in your environment (i.e., WebLogic server). The HTTP settings allow you to change the behavior of HTTP communication and in some circumstances may help improve the performance of HTTP communication.

This section covers:

- Configuring the HTTP Server Port Number
- Changing the Broker's Management Channel Port
- Configuring the HTTP Server Thread Settings
- Configuring the HTTP Client Settings

- Configuring the HTTP Proxy Settings

Configuring the HTTP Server Port Number

The default port used by the HTTP Server and the Broker Configurator is 9032. If port 9032 is currently being used, the Broker will not start.

To change the port number:


- 1 Stop the Broker if it is currently started.
- 2 Using a text editor, open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Change the port number the `com.hp.http.server.port` entry. For example:

```
<entry name="com.hp.http.server.port">9035</entry>
```
- 4 Save and close the file.
- 5 Restart the Broker.

Configuring the Broker's Management Channel Port

The Broker's management channel port is used to publish the management WSDLs for brokered Web services (i.e., `http://host:9032/wsmf/services`). The management WSDLs are used by SOA Manager server to get management data about brokered Web services.

By default, the management channel port is set to port 9032 which is also the application channel port that receives Web service requests. To separate management channel and application channel traffic, change the management channel port.

 The management channel port is required when registering a WSM Broker with the Network Service server. If the default port number is changed, make sure that the new port number is known when the WSM Broker is being registered with the Network Service server.

For more instructions on securing the management channel, see the *SOA Manager Administrator Guide*.

To define a different server port for the management channel:

- 1 Stop the Broker if it is currently started.
- 2 Use a text editor to open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Specify a port value for the `com.hp.http.server.managementPort` element. Make sure the port is not being used by any other application on your system. For example:

```
<entry name="com.hp.http.server.managementPort">9033</entry>
```
- 4 Save and close `mipServer.xml`.
- 5 Start the Broker server.

Configuring HTTP Server Thread Settings

You can change the manner in which the HTTP server manages threads. Thread management can help increase performance and improve latency for the HTTP Server. There are three thread settings:

- `<entry name="com.hp.http.threads.max">` – The maximum number of threads allowed to be used by the HTTP server.
- `<entry name="com.hp.http.threads.min">` – The minimum number of threads allowed to be used by the HTTP server.
- `<entry name="com.hp.http.threads.maxIdle">` – The maximum amount of time in milliseconds that an HTTP server thread can remain idle.

To change HTTP server thread settings:

- 1 Stop the Broker if it is currently started.
- 2 Use a text editor to open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Configure the HTTP Server Thread settings. For example:


```
<entry name="com.hp.http.threads.max">50</entry>
<entry name="com.hp.http.threads.min">2</entry>
<entry name="com.hp.http.threads.maxIdle">60000</entry>
```
- 4 Save and close the file.
- 5 Restart the Broker.

Configuring HTTP Client Settings

The WSM Broker contains an HTTP client used to communicate to an HTTP server. In particular, the client is used to send requests to and receive responses from the containers that are hosting Web services. The client settings can improve performance between the HTTP client and an HTTP server.

- `<entry name="com.hp.http.client.keepAlive">` – Indicates the HTTP client will reuse a network connection to the server. This usually has performance benefits because the client does not need to keep opening and closing sockets. Typically, the value is set to `true`. Valid values are either `true` or `false`.
- `<entry name="com.hp.http.client.chunking">` – Allows the HTTP client to send data by breaking it into smaller chunks. Chunking information allows the client and server to process large amounts of data without using as much memory. Typically the value is set to `true`. However, some HTTP servers may not support this feature, in which case the value should be set to `false`.

To configure HTTP client settings:

- 1 Stop the Broker if it is currently started.
- 2 Use a text editor to open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Configure the HTTP Server Thread settings. For example:


```
<entry name="com.hp.http.client.chunking">true</entry>
<entry name="com.hp.http.client.keepAlive">true</entry>
```

- 4 Save and close the file.
- 5 Restart the Broker.

Configuring HTTP Proxy Settings

Many networks use a proxy server that enables access to resources that are external to a network. This is also true for external Web services that are being managed by a Broker. The `Proxy Host` and `Proxy Port` settings allow you to define a proxy server. If set, all requests sent to a brokered service are dispatched to the final endpoint through the proxy server.

However, a proxy server is not required to access addresses that are internal to the network. Therefore, if you are managing Web services that are both internal and external to the network, the `Non-proxy Hosts` setting allows you to define a set of hosts that never require the use of a proxy server.



You do not need to set the `Non-proxy Hosts` setting if you do not define a proxy server.

To configure the HTTP proxy settings:

- 1 From the Broker Configurator's main toolbar, click **HTTP Settings**. The **HTTP Settings** screen displays.
- 2 Use the `Proxy Host` and `Proxy Port` text boxes to enter a proxy server's host and port. The host value must be an IP address or the full DNS name of the server.
- 3 Use the `Non-proxy Hosts` text box to enter a list of hosts that do not require the use of a proxy server. Use the pipe character (|) to separate entries. For example:
`localhost | 15.* | 16.* | 127.*`
The local host and any hosts in the 15, 16, and 127 domain space do not require the proxy server.
- 4 Click **Save** to save your changes.

Assigning Access to the Console

The `<install_dir>\conf\broker\mipServer.xml` file allows you to define user credentials for accessing the Broker's console. In particular, you can define usernames and passwords for accessing the console. A single role, `admins`, has been implemented. All users must be associated with this role.

To add console access rights for a user:

- 1 Stop the Broker if it is currently started.
- 2 Using a text editor, open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Add a new user and password entry. For example:

```
<entry name="com.hp.mip.server.security.user">Joe User</entry>  
<entry name="com.hp.mip.server.security.password">password</entry>
```

- 4 Save and close the file.
- 5 Restart the Broker.



You can use the Broker Configurator to change a user's password. You must be logged into the Configurator as the user in order to change the password. See the *Broker Configurator Online Help* for detailed instructions.

Using XPL Logging

SOA Manager uses HP Cross Platform (XPL) logging. Installation, configuration, and usage are described below.

Installing XPL Logging

During the SOA Manager installation, you may be prompted to select the HP Software installation and data directories. You will only be prompted for this information if this is the first time you have installed an HP Software product.

The default value for the installation directory is C:\Program Files\HP Software on Windows and /opt/OV on UNIX. The default value for the data directory is C:\Program Files\HP Software\data on Windows and /var/opt/OV on Unix. The Broker log files are created in the log subdirectory of the data directory. If you do not run the Broker as an administrator, you may need to change the permissions for the log subdirectory.

XPL Tools

The HP Software Cross Platform Component contains logging and tracing tools. If you need to change the default log file configuration parameters, install the component. Run the appropriate installer in the /Support directory of the SOA Manager CD.

Configuring XPL

The Broker automatically creates log files in the log subdirectory of the HP Software data directory. The Broker log file name has the format:

broker[unique].sequence.locale

For example:

broker0.0.en_US

This is the first broker log file created for the US English locale.

The Broker creates a log file for an English locale and a second file for your system's locale if it is different from English.

The Broker creates up to 10 log files, each file containing up to 1 megabyte of data. The log files will have sequence numbers 0 through 9. When the maximum number of log files is exceeded, the sequence 0 log file is overwritten.

You can change the maximum number of log files and log file size using the HP Software Cross Platform tool, `ovconfchg`. After installing the HP Software Cross Platform Component, this program is in the `bin` directory of the HP Software installation directory. An example of using this tool is shown below.

```
ovconfchg -ns xpl.log.OvLogFileHandler -set filecount 12  
-set filesize 2
```

This command sets the maximum number of log files to 12 and the maximum log file size to 2 megabytes.



Restart the Broker for the new configuration to take effect.

You can see the current configuration using this command:

```
ovconfget
```

For more information about `ovconfchg` and `ovconfget`, see the help documentation in the `help` subdirectory of the HP Software installation directory.

Configuring Log Levels

You can change the Broker log levels using the BSE. Alternatively, you can change the log levels by editing the `logging.properties` file in the `JDK lib` directory or the `xpllogging.properties` in the `<install_dir>/conf/broker` directory. The log levels are: SEVERE, WARNING, INFO, FINE, FINER, and FINEST. By default the log level is set to INFO.

Using JRE Properties File

You can change the log level for the Broker by editing the `logging.properties` file in the `JRE lib` directory. You must restart SOA Manager and the Broker to make the changes take effect. For example, you can add the following line to the end of `logging.properties`:

```
com.hp.ov.mip.level = FINE
```

This sets the log level for the Broker to `FINE`.

Using the XPL Properties File

You can change the log level for the Broker by editing the `xpllogging.properties` in the `<install_dir>/conf/broker` directory. You must restart the Broker for the changes take effect. For example, you can add the following line to the end of the file:

```
com.hp.ov.mip.level = FINE
```

This sets the log level for the Broker to `FINE`.

Viewing Logs

You can use an editor or the BSE to view the Broker log files. In the BSE, go to a Broker's Resource View screen and click the **View Log** link. Alternatively, use an editor to view the Broker log files in the HP Software data log directory.

Using XPL Tracing

SOA Manager uses the HP Software Tracing tools for tracing. Please refer to the *HP Software Tracing Concepts Guide* for detailed information on how use the trace feature. The guide is located on the SOA Manager CD in the /Documentation directory.

Installation

Before beginning this procedure, verify if the HP Software Tracing tools are already installed on your system. You can check to see if the trace server is installed. On Unix, the trace server is installed as /opt/OV/lbin/xpl/trc/ovtrcd. On Windows, the trace server is installed as C:\Program Files\HP Software\bin\ovtrcsvc.exe.

The tracing tools are located on the SOA Manager CD in the /Support directory.

Windows

To install the tracing tools on a Windows system, double-click on /Support/HPOvXpl-<version>-release.msi.

HP-UX

To install the tracing tools on an HP-UX system, run:

```
swinstall -s /Support/HPOvXpl-<version>-HPUX11.0-release.depot \*
```

Linux

To install the tracing tools on a Linux system, run:

```
rpm -Uhv /Support/HPOvXpl-<version>-Linux2.4-release.rpm
```

Example Configuration Entries

The following SOA Manager entries are example entries for the XPL configuration file:

```
TCF Version 3.2
APP: "networkservices"
SINK: Socket "system1.acme.com" "node=192.1.60.106;"
TRACE: "mip.config" "Operation" Info Error
TRACE: "mip.config" "Parameters" Info Error
TRACE: "mip.config" "Procedure" Info Error
TRACE: "mip.metrics" "Operation" Info Error
TRACE: "mip.metrics" "Parameters" Info Error
TRACE: "mip.metrics" "Procedure" Info Error
TRACE: "mip.slos" "Operation" Info Error
TRACE: "mip.slos" "Parameters" Info Error
TRACE: "mip.slos" "Procedure" Info Error
TRACE: "mip.deploy" "Operation" Info Error
TRACE: "mip.deploy" "Parameters" Info Error
TRACE: "mip.deploy" "Procedure" Info Error
```

Implementing Load Balancing and Failover

This chapter provides instructions for setting up the load balancing and failover features that are included with the WSM Intermediary. In addition, an overview and conceptual architecture for load balancing and failover is provided.

The load balancing and fault tolerance features included with the Intermediary are primarily designed for requests made between an intermediary service and its Web service endpoints. However, load balancing and failover can also be implemented between a client and an Intermediary. The final section “Using Multiple Intermediaries” explains this scenario and provides implementation instructions.

Overview

The WSM Intermediary contains a load balancing and failover feature that automatically routes a Web service request that is made to an intermediary service to multiple endpoints. Should requests to a primary endpoint fail, a backup endpoint is automatically used instead. The endpoints are defined in a service’s definition (WSDL) file and are configured when an intermediary service is created using the Broker Configurator console. When a Web service with multiple endpoints is managed, the management information (success, response time, and so on) for each endpoint is aggregated.

Load balancing and failover is an important part of distributed applications and offers some key benefits. In particular, these features:

- Provide redundancy – Multiple instances of a Web service that are spread across different hosts means a service is always available for requests.
- Minimize downtime – Multiple instances of a Web service that are spread across different hosts allows an application to continue making requests even if one host fails or is being serviced.
- Increase reliability – Users never experience an unavailable application.
- Improve performance – Request loads are spread across different hosts, which prevents bottlenecks from occurring.

- Reduce single points of failure – Requests to an endpoint which is failing are automatically rerouted to working endpoints.

Conceptual Architecture

All requests that are sent to an intermediary service are sent to a final endpoint using the Intermediary's dispatcher. A list of available endpoints is registered with the Intermediary and is used to find endpoints that can satisfy a request.

A WSDL file is used to define a service and the endpoints (SOAP addresses) available for the service. When an intermediary service is created from the WSDL file, these endpoints are discovered and registered by the Intermediary and configured as either an active endpoint or a backup endpoint.

Load Balancing Scenario

Active endpoints are the primary addresses that are used to service a request. Multiple active endpoints can be used to share the load of servicing requests. Only after all active endpoints fail, will a backup endpoint be used. When a request is dispatched to an active endpoint, it is done using a round robin scheme. That is, an endpoint is used once and then moved to the bottom of the list of available endpoints. The next request goes to the next endpoint on the list and then that endpoint is moved to the bottom of the list and so on.

Failover Scenario

Backup endpoints are only used when all active endpoints fail. A failure occurs when an HTTP Status code is returned that is greater than or equal to 300, less than 500, or equal to 503. While the backup endpoint is being used, the Intermediary continues to try an active endpoint at 30 second intervals. When an active endpoint becomes available, requests are again routed to it and the backup endpoint is no longer used.



If you have multiple backup endpoints, requests are sent using a round robin scheme.

Setting Up Load Balancing and Failover

Load Balancing and failover is set up for each intermediary service that you create. When you create an intermediary service, each endpoint that is discovered can be configured as either an active endpoint or a backup endpoint. This section describes how to modify a WSDL file to include multiple endpoints and how to configure each endpoint as an active or backup endpoint.

Defining Multiple Endpoints in a WSDL File

The load balancing and failover feature is dependent on a WSDL file that defines multiple endpoints for a Web service. For example, if two instances of the same Web service are running on two different hosts, then a single WSDL file can be used to define the Web service and each endpoint that is available. Endpoints are defined in the `<service>` node of a WSDL file as demonstrated below for the finance service:

```
<service name="FinanceService">
  <port name="FinanceServiceSoap" binding="tns:FinanceServiceSoap">
    <soap:address
      location="http://host1:7001/FinanceService/FinanceService" />
    </port>
  <port name="FinanceServiceSoap" binding="tns:FinanceServiceSoap">
    <soap:address
      location="http://host2:7001/FinanceService/FinanceService" />
    </port>
</service>
```

The `FinanceService` above contains two SOAP address endpoints. One endpoint is located on `host1` and the other is located on `host2`. Each endpoint must be defined within a `<port>` node that also defines the `PortType` and binding.



Before creating an intermediary service using the Broker Configurator, make sure you have modified a WSDL to include multiple endpoints as demonstrated above.

Configuring Load Balancing and Failover

An intermediary service is created by using the Broker Configurator. The create service wizard steps you through the process of creating an intermediary service, including importing a WSDL file and configuring whether an endpoint should be an active endpoint or a backup endpoint.

To configure load balancing and failover:

- 1 Log in to the Broker Configurator.
- 2 Click on the **Create Brokered Web Service** link. Step 1 of the Create Brokered Service wizard displays (Step 1: Import WSDL).
- 3 Enter a WSDL that defines multiple endpoints for a Web service.
- 4 Click **next** to move to Step 2 of the wizard (Step 2: Configure Endpoints).
- 5 By default, an endpoint is configured to be the primary endpoint as indicated by the **Primary** option in the Options field. Click to select the **Backup** option if the endpoint is to be only used as a backup if a primary endpoint should fail.



Endpoints can only be configured when an intermediary service is initially created.

Using Multiple Intermediaries

Multiple Intermediaries are used to provide an additional level of assurance that no single point of failure exists between clients and an Intermediary. In this scenario, a third party load balancer, such as Cisco's IP Director, is used to balance requests between two or more Intermediaries that are running on different hosts.

Each Intermediary contains an intermediary service for the same Web service. Loads are balanced between each intermediary service and if one Intermediary fails, additional intermediaries are available to continue servicing requests. Management information (i.e., success, response time, and so on) for each intermediary service is aggregated. In addition, each intermediary service can be viewed separately in a single business service when using the HP SOA Manager web interface.

When implementing this scenario, use the instructions in the "Setting Up Load Balancing and Failover" section discussed previously for each installation of the WSM Intermediary.



It is beyond the scope of this documentation to detail installation and configuration of a third party load balancer. See the documentation that was included with your load balancer product for full installation and setup instructions.

Using the Intermediary's Security Features

This chapter provides instructions for securing the Web services application channel when using a WSM Intermediary deployment scenario. An overview section has been included that introduces many of the fundamentals of the security implementation. Users should be familiar with general security principals and Web services-based security before completing the instructions in this chapter.



The use of the security implementation is dependent on the use of the WSM Intermediary. You can use such deployment scenarios in conjunction with the WSM Intermediary and thus leverage the security features that are provided with the Intermediary and discussed in this chapter.

Overview

While emerging trends in Web services architecture indicate that the future of Web services is loosely coupled, multi-hop, document exchange style message oriented interactions; most current implementations are point-to-point request-response HTTP based. Most enterprise security groups have existing security infrastructure and products established in house. The Intermediary security architecture takes this into consideration and provides a comprehensive set of options for securing Web services either at the (HTTP) transport layer or (SOAP) messaging layer.

Feature Matrix

The following table lists the support technology that is included with the Intermediary security solution.

Security Concern	Transport Level	Message Level
Authentication	HTTP/S: basic auth HTTPS: X.509 certificates HTTP/S: SSO tokens	WS-Security: User password WS-Security: X.509 certificates WS-Security: SSO tokens
Confidentiality	SSL	WS-Security: XML-Encryption
Integrity	SSL	WS-Security: D-Sig
Auditing	SOA Manager	SOA Manager
Non-Repudiation	SOA Manager Audit Service (using D-Sig)	SOA Manager Audit Service (using D-Sig)

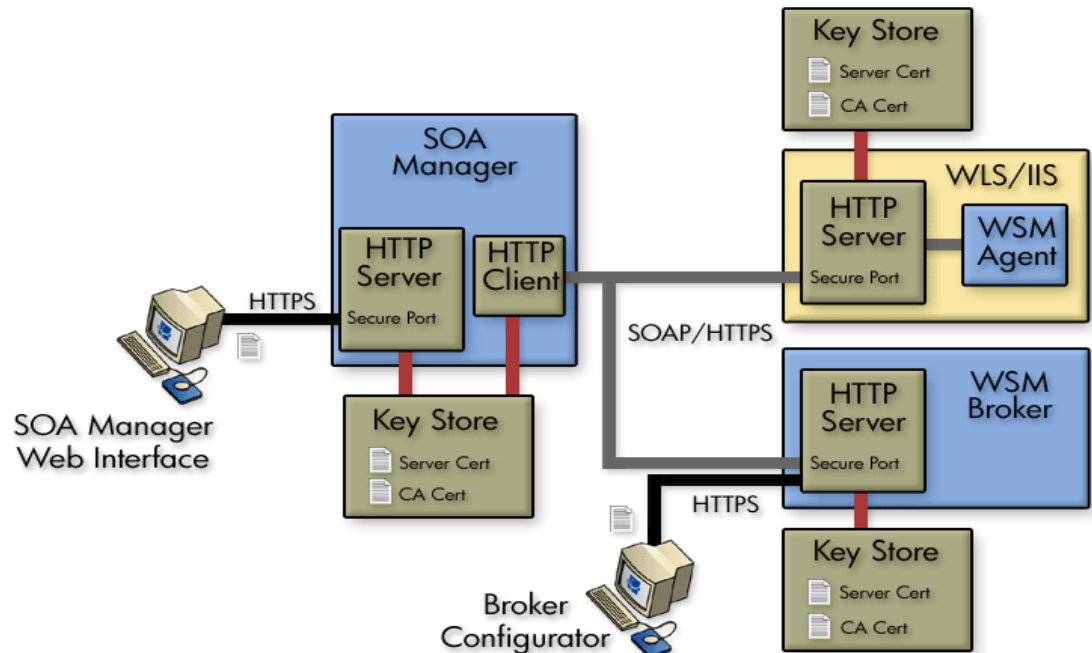
- All User Identity Management – authentication, authorization, and administration is deferred to enterprise security products.
- WS-Security implementation in the Intermediary (D-Sig, Encryption) is done using Verisign TSIK toolkit.
- Java Key Store and PKCS12 Key Stores can be used for PKI support – except that covered by the security products.

Supported Security Scenarios

This section describes end-to-end security scenarios supported by the Intermediary security implementation. There are three basic security scenarios discussed:

- Scenario 1: Intermediary is the Entry Point for External Consumers.
- Scenario 2: Web Application is the Entry Point for External Consumers.
- Scenario 3: Intermediary is the Exit Point for External Providers.

The figure below shows a high level view of the Intermediary security implementation and includes all three scenarios.



Scenario 1: Intermediary is the Entry Point for External Consumers

In this scenario, incoming HTTP/S traffic through the firewall is front-ended by the Intermediary. The Intermediary supports HTTP/S basic authentication and X.509 client certificate authentication over SSL. Alternately, the intermediary can also be configured to decrypt incoming message payload and use X.509 certificates embedded in the digital signature of the payload to authenticate the message.

Authentication/Authorization failures are tracked and sent to the SOA Manager so that alerts can be raised if the failures exceed SLO threshold values.

The security provider typically returns a security token (referred to as SSO token) as a result of successful authentication. This token can be propagated further to the back end Web service implementation either as an HTTP header or embedded in a WS-Security header in the payload. Obviously, for this to be meaningful, the back end Web service container platform must be integrated with the SSO security provider.

In case the back-end Web service container platform is not participating in the SSO, there are three options:

- Once authentication/authorization is done at the intermediary, no subsequent security authentication/authorization is done at the back end Web service implementation. In this case, firewalls may be configured to ensure that all traffic entering the Web service implementation is coming authenticated and authorized through the intermediary. The shortcoming of this approach is that business logic requiring security principal information cannot be written unless such information is also present in the message payload.

- A variation of the above option is that all actual authentication/authorization is done at the intermediary, but the intermediary presents some normalized identity to the back end Web service implementation. For example, some things like user, intermediary, password, and secret such that the back end application can be secured without having to configure firewalls. This too has the shortcoming that original security principal information is lost in the transition between intermediary and Web service implementation. However, it does make the back end implementation secure. The Intermediary (dispatcher) can be configured with credentials for basic authentication or x.509 client certificates that it can present while authenticating against back end Web service implementations. This can be done at the HTTP layer or embedded as WS-Security headers in the payload.
- If it is technically not feasible to integrate the SSO solution to the back end Web service container environment, the SSO problem can potentially be solved at the Intermediary. The Intermediary would have to know how to present credentials for represented principals in the back end Web service container realm. Some mapping must be made between incoming security principals and those known to the Web service container realm. Intermediary security does not natively support identity mapping features.

Scenario 2: Web Application is the Entry Point for External Consumers

Incoming traffic such as regular Web application requests (i.e. non-SOAP) is authenticated at the Web Server/Web Application Server layer. If this layer is already integrated with the SSO provider, it can make requests against the Intermediary by propagating the SSO security token over SSL. The tokens can be presented either as HTTP headers or embedded in the WS-Security header. The Intermediary supports both styles for re-authentication against the SSO security provider.

Alternately, the internal Web service consumer may present some other authentication credentials via HTTP/S basic authentication, X.509 certificates over SSL or WS-Security D-Sig. The Intermediary can be configured to use any of these for authentication against the security provider. In this case, the Intermediary behavior is no different than that specified in Scenario 1, where it accepted calls from external consumers.

When the Intermediary forwards the request on to its final destination, it can support all the options described in Scenario 1.

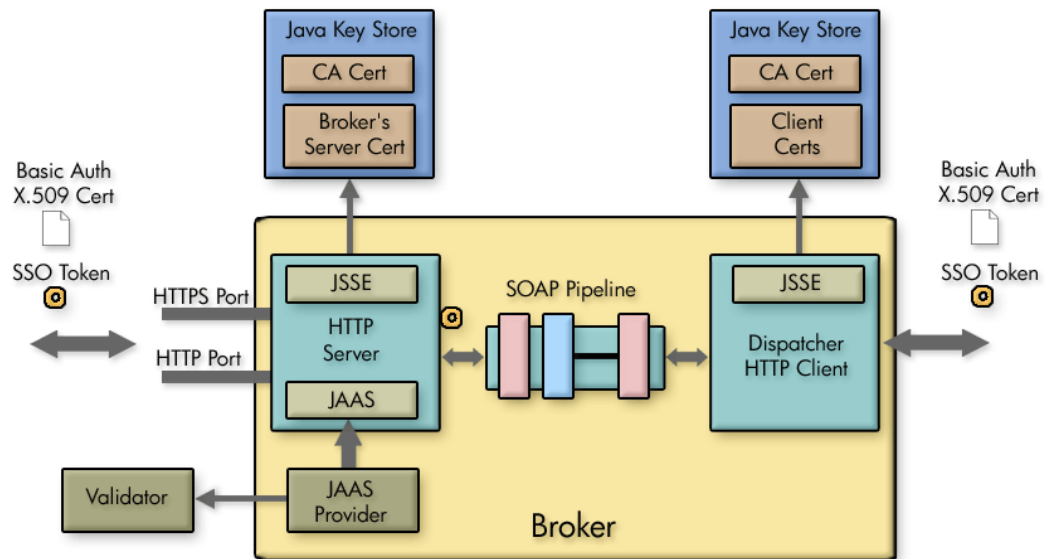
Scenario 3: Intermediary is the Exit Point for External Providers

This scenario is covered between Scenario 1 and Scenario 2 and does not require any different explanation. In addition, Intermediary security does not support SAML. However, future releases of SOA Manager will provide SAML support.

Transport Level Security

HTTP/S serving is done by the Intermediary. HTTP/S client side (known as the Dispatcher) is implemented using a performance enhanced version of Jakarta commons HTTP Client that further uses JSSE for its SSL implementation.

Each intermediary service can be configured with transport security options for inbound traffic. The following figure shows a common view of transport level security.



Message Level Security

Message level security is offered using SOAP handlers. Figure 7-1 shows a common view of message level security.

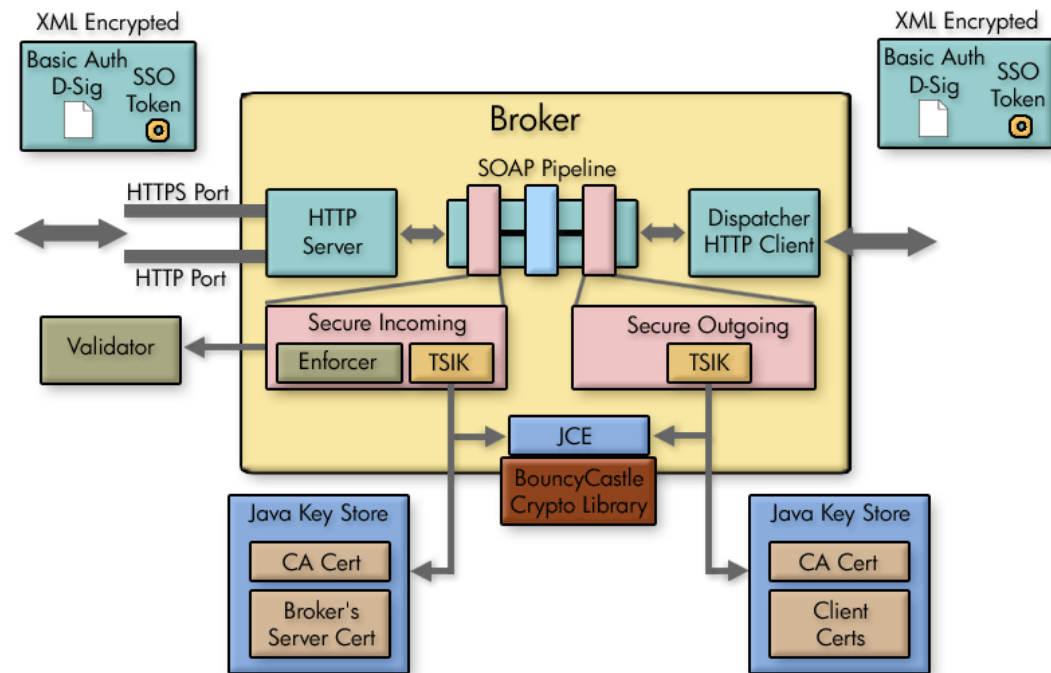


Figure 7-1: Message Level Security

Inbound Message Processing

Inbound request payload can be decrypted using the Intermediary's server certificate. This assumes that the public key for this certificate was exchanged a priori (exactly how is out of the scope of this documentation) with the caller of the message and was used to encrypt the message. Once decrypted, the digital signature of the message is validated to ensure that the message integrity has not been tampered with. The digital signature contains the clients X.509 certificate (or chain leading to CA certificate). This certificate can be used to authenticate the message sender. The message processing handler also saves this certificate in case it needs to be used to encrypt the response before returning the response to the caller.

Meta-data required for XML Encryption and D-Sig behavior is extracted from WS-Security headers. Actual underlying implementation is provided by Verisign's TSIK toolkit. This toolkit uses JCE to provide crypto algorithms. SOA Manager includes BouncyCastle JCE provider by default. We do not provide any PKI maintenance and customers are expected to use the Java Key Store.

Three types of WS-Security header credentials can be used for authentication:

- plain user:password, X.509 certificates
- incoming SSO token

Outbound Message Processing

Outbound payload can be digitally signed using the Intermediary's server certificate configured in the Java Key Store. This digital signature embeds the Intermediary's X.509 certificate into a WS-Security header. It can be used by the receiver to authenticate the intermediary. Alternately, we can also embed a WS-Security user:password or WS-Security SSO token that either entered the Intermediary or that was created by authenticating against the security provider.

Once signed, it can be encrypted using the receiver's public key. This must have been entered into the Java Key Store a priori. The key alias is then specified in the configuration.

The returned response can be decrypted using the Intermediary's server certificate and payload integrity can be validated by checking against the embedded D-Sig.

Setting Up the Security Components

As discussed in the "Overview" section, the Intermediary utilizes several external security components in order to secure communication on the application channel. The components must be configured as discussed in this section prior to implementing a security scenario. In addition, The Intermediary must be configured to use the various security components.

If you do not require the security features provided by a particular security component, you may skip the setup instruction for that component. However, if you are unsure of which security components you require or if you are testing different security capabilities, it is suggested that you setup all the security components.

- ▶ This section does not cover the security configuration at the WS Container or in the consumers (applications) that are using the Web services. Refer to your vendor's documentation for instructions on setting up security.

Configure a Key Store

The steps below detail how to use the Broker Configurator to configure a Key Store for use by the Intermediary.

- ▶ A Key Store is required in the following steps. The Intermediary security solution supports both Java Key Stores and PKCS12 Key Stores. The steps below outline the configuration for use with a Java Key Store. For information on creating a Java Key Store, see Appendix A "Creating a Java Key Store."

To configure a Java Key Store:

- 1 Start the Broker Configurator.
- 2 From the Configurator's main tool bar, click **SSL Settings**. The SSL Settings screen displays.
- 3 Set the following properties:
 - **Keystore Location:** The location of your Java Key Store (i.e., C:\crypto\scream.jks).
 - **Keystore Password:** The password for your Java Key Store.
 - **Keystore Type:** Because we are using a Java Key Store this property is set to "jks".
 - **Private Key Alias:** The alias of the Java Key Store private key.
 - **Private Key Password:** The private key password in the Key Store.
- 4 From the bottom of the screen, click **Save**.

Configure a CA Trust Store

A CA Trust Store is used to store certificates from Certificate Authorities (CA) that are to be considered trusted. In these instructions, the Trust Store is a Java Key Store populated with certificates from trusted CA's. The Java Developers Kit includes Java Secure Socket Extension (JSSE) which provides a populated Trust Store and is located in `<jdk_install>/jre/lib/security/cacerts`.

- ▶ A Key Store is required in the following steps. The Intermediary security solution supports both Java Key Stores and PKCS12 Key Stores. The steps below outline the configuration for use with a Java Key Store. For information on creating a Java Key Store, see Appendix A "Creating a Java Key Store."

To configure the intermediary to use a CA Trust Store:

- 1 From the Configurator's main tool bar, click **SSL Settings**. The SSL Settings screen displays.
- 2 Set the following properties:
 - **Truststore Location:** Trust Store location (i.e., `<jdk_install>/jre/lib/security/cacerts`).
 - **Truststore Password:** Trust Store password. By default, the Trust Store password is `changeit`.
 - **Truststore Type:** Because we are using a Java Key Store, this property is set to `jks`.



If you have changed any defaults associated with this Trust Store, the above entries will not work. Ensure settings are configured to match that of your environment.

- 3 From the bottom of the screen, click **Save**.

Configure the Intermediary's SSL Port

The Intermediary's SSL port is used to accept HTTPS requests and is used to implement transport-level security. You must define which port you want to use to accept HTTPS requests.

To configure the Intermediary's SSL Port:

- 1 From the Configurator's main tool bar, click **HTTP Settings**. The HTTP Settings screen displays.
- 2 In the HTTPS Server Port field, enter the port you want the Intermediary to use for SSL connections.
- 3 From the bottom of the screen, click **Save**.

Setting Up Authentication and Authorization

This section provides details on how to provide authentication and authorization. The intermediary supports basic authentication and authorization using basic authentication and x.509 client certificates. For either scenario, you can implement authentication and authorization for all intermediary services, specific intermediary services, or for specific operations within an intermediary service.

By applying authentication and authorization services to your Web services, you can confidently ensure that only selected consumers gain access to identified resources. The Intermediary security solution provides authentication and authorization services on a best of breed approach by integrating to well known and proven enterprise security products.

Implementing a Security Scenario

This section provides instructions for implementing security scenarios. There are scenarios for both transport-level security and message-level security. The security scenarios include options for securing inbound communication from a consumer to the Intermediary and outbound communication from the Intermediary to a Policy consumer.



Before implementing a security scenario, you must configure the security components that are used by the Intermediary (see “Setting Up the Security Components” above).

The security scenarios discussed in this section are not mutually exclusive. You may choose to implement a single scenario, or you may choose to combine several scenarios together. The scenarios you choose to implement depend on the security requirements of your environment and the security requirements of your applications. Refer to the “Overview” section above for detailed information about the Intermediary's security capabilities.

The scenarios discussed in this section include:

- Inbound Transport Security
- Outbound Transport Security
- Inbound Message Security
- Outbound Message Security

Inbound Transport Security


In this scenario, the Intermediary accepts requests from consumers using SSL and authenticates/authorizes the user using a security provider. This is a typical scenario where an enterprise needs to secure inbound communications but does not need to secure the channel when calling the actual endpoints. An example of this could be providing a service externally; once the messages are received and through the firewall, the secure channel is not needed as the messages are traveling across a private network.

Enabling SSL

The Broker Configurator is used to configure an intermediary service and enable inbound SSL connections. You can configure SSL when you create an intermediary service or you can edit an existing intermediary service.

To enable inbound SSL:

- 1 From the Broker Configurator, create a new or edit an existing intermediary service.
- 2 From the Service Configuration screen, check the **Use SSL** option located in the Inbound Transport section.
- 3 At the bottom of the screen, click **Save Changes**. The Brokered Services screen opens. The service you just configured has a Service Interface URL that indicates HTTPS. This is the URL your clients should use to access the service.

- 
- If the Key Store was configured with a signed server certificate from a Certificate Authority (CA) which is not commonly known, you may see an error message indicating that a trust relationship could not be established. If this is the case, you will need to obtain the CA's certificate and install that in the Trust Store for all clients who will access this service.

Enabling Authentication


The Broker Configurator is used to configure an intermediary service and enable authentication for inbound transport security. Users are authorized using a security provider. You can enable authentication when you create an intermediary service or you can edit an existing intermediary service.

To enable authentication:

- 1 From the Broker Configurator, create a new or edit an existing intermediary service.
- 2 From the Service Configuration screen's Inbound Transport section, check the type of authentication you want to enable:
 - **Basic Authentication:** All requests to the Intermediary need to be authenticated using a user name and password.
 - **X.509 Client Certs:** All requests to the Intermediary need to be authenticated using an X.509 certificate.
- 3 At the bottom of the screen, click **Save Changes**. Once this service is deployed, the Intermediary will communicate with security provider to ensure that the consumer has supplied the proper credentials to gain access to the service. If the user is not authenticated and/or authorized, the Intermediary will return a 404 Not Authorized error.

Outbound Transport Security

In this scenario, the Intermediary accepts requests from consumers and then forward that request to the provider using an SSL channel. This scenario can be combined with the inbound transport scenario to provide end-to-end transport-level security.

- 
- When using outbound SSL Security, a Web Service deployed in a Policy enforcement agent must be configured to use SSL from within that Policy enforcement agent. See your Policy Enforcement Agent documentation for more instructions on setting up SSL communications.

Enabling Outbound SSL

The Broker Console is used to configure an intermediary service and enable outbound SSL connections. You must enable SSL when you create an intermediary service. You cannot edit an existing intermediary service to use outbound SSL.

To enable outbound SSL:

- 1 From the Configurator's main toolbar, click **Create Brokered Web Service**. Step 1 of the Create Brokered Service wizard displays (Step 1: Import WSDL).


- 2 In the text box, specify the WSDL with HTTPS if your server will dynamically create port bindings based off of the WSDL URL. For example:

```
https://company.com/finance?wsdl
```

Or,

Click **browse** to locate a Web service's WSDL.

- 3 Click **next** to move to Step 2 of the wizard (Step 2: Configure Endpoints). A binding is created for the Web service and displays in the Select Endpoints screen. If a Web service definition contains multiple endpoints, a binding for each endpoint is listed.
- 4 From the Authentication field, click to select the **Send Credentials** check box.
- 5 Complete creating the intermediary service by following the prompts. The intermediary service is configured to use outbound SSL when you have completed creating the intermediary service and it is deployed.

 If the endpoint has a server certificate signed by a CA whose CA Certificate is not present within the trust store configured for the Intermediary, the SSL handshake will fail. Make sure the endpoint's CA's Certificate is located in the Intermediary's trust store.

Inbound Message Security

In this scenario, a consumer must authenticate with the Intermediary before messages are accepted. In addition, the consumer may choose to encrypt messages before sending them to the Intermediary; in which case, the intermediary will decrypt the messages before they are dispatched to the final endpoint. Refer to Figure 7-1 for a conceptual architecture of message-level security.

The Broker Configurator is used to configure an inbound message security handler for an intermediary service. You can enable message security when you create an intermediary service or you can edit an existing intermediary service.

To enable inbound message security:

- 1 From the Broker Configurator, create a new or edit an existing intermediary service.
- 2 From the Service Configuration screen's Feature section, click the **Inbound Message Security** option. The security options display.
- 3 Click the security option you want to enable:
 - **Username-Password Authentication:** All messages to the Intermediary need to be authenticated using a user name and password.
 - **Digital Signature Authentication:** All messages to the Intermediary need to be authenticated using a digital signature.
 - **Digital Signature Authentication with Decryption:** All messages to the Intermediary need to be authenticated using a digital signature. In addition, the Intermediary's private key is used to decrypt the message.

- 4 Click to select the **No Digital Signature or Encryption in Response** option if you do not require the response message to be encrypted or have a digital signature. If you do not select this option, the intermediary expects the response message to be encrypted and have a digital signature.
- 5 At the bottom of the screen, click **Save Changes**. Once this service is deployed, the Intermediary will communicate with the security provider for all inbound requests to ensure that the consumer has supplied the proper credentials to gain access to the service. If the user is not authenticated and/or authorized, the Intermediary will return a 404 Not Authorized error.



The Intermediary will fail to recognize a Digital signature if the XML payload is changed after it has been signed. This typically happens during debugging when the XML payload is reformatted in “pretty print” for ease of reading. If the payload is reformatted, it should not be sent to the Intermediary.

Outbound Message Security

In this scenario, The Intermediary must authenticate itself with a Policy enforcement agent before messages are processed at the Policy enforcement agent. The Policy enforcement agent and the Intermediary can share the same security provider or a Policy enforcement agent’s security provider is used to complete the authentication. In addition, the Intermediary can encrypt messages before sending them to the Policy enforcement agent; in which case, the Policy enforcement agent must be able to decrypt the messages. Refer to Figure 7-1 for a conceptual architecture of message-level security.

The Broker Configurator is used to configure an outbound message security handler for a brokered service. Requests are authorized using a Policy enforcement agent’s security provider and encryption is implemented through a Key Store (See “Configure a Key Store” above). You can enable message security when you create an intermediary service or you can edit an existing intermediary service.

To enable outbound message security:

- 1 From the Broker Configurator, create a new or edit an existing intermediary service.
- 2 From the Service Configuration screen’s Feature section, click the **Outbound Message Security** option. The security options displays.
- 3 Click the security option you want to enable:
 - **Username-Password Authentication:** All messages dispatched to a Policy enforcement agent need to be authenticated using a user name and password. The Policy enforcement agent’s security provider is used to verify the credentials and which resources can be accessed. Enter a valid Username and Password for your Policy enforcement agent in the fields provided.
 - **Sign:** All messages dispatched to a Policy enforcement agent will include a digital signature. The Intermediary’s Key Store is used to sign the outbound message.

- **Sign and Encrypt:** All messages dispatched to a Policy enforcement agent will include a digital signature and will be encrypted. The Intermediary's Key Store is used to sign the outbound message. In addition, the Intermediary's private key must be located at the Policy enforcement agent to decrypt the message.
- 4 Click to select the **No Digital Signature or Encryption in Response** option if the response message does not have digital signature and is not encrypted. If you do not select this option, the intermediary expects the response message to have a digital signature and/or be encrypted.
- 5 At the bottom of the screen, click **Save Changes**.

Management Channel HTTP Basic Authorization

HTTP basic authorization can be enabled to secure the intermediary management channel. This functionality is the same as securing the application channel.

To configure intermediary management channel security:

- 1 Stop the Intermediary if it is currently started.
- 2 Use a text editor to open `<install_dir>\conf\broker\mipServer.xml`.
- 3 Remove the comment tag and text (`<!-- -->`) from the following three property entries:

```
<entry name="com.hp.mip.security.provider.management">
    default</entry>
<entry name="com.hp.mip.security.sba.user">user</entry>
<entry name="com.hp.mip.security.sba.password">password</entry>
```

- Specify the name of the security provider for management channel in the `com.hp.mip.security.provider.management` element.
- Specify the user name for the user who is authorized to access the Web URL of the management channel in the `com.hp.mip.security.sba.user` element.
- Specify the password for the user who is authorized to access the Web URL of the management channel in the `com.hp.mip.security.sba.password` element.

For example:

A Intermediary is running on `Myhost` and its management channel is running on non-secure port 9035. The security provider, `SelectAccess`, sets up web access control for any resources under `http://Myhost:9035/wsmf/`. User `jsmith`, with password, `johnspassword`, is authorized to access these Web resources. The values of the three entries are set to:

```
<entry name="com.hp.mip.security.provider.management">
    SelectAccess</entry>
<entry name="com.hp.mip.security.sba.user">jsmith</entry>
<entry name="com.hp.mip.security.sba.password">
    johnspassword</entry>
```

- 4 Save and close `mipserver.xml`.
- 5 Start the Intermediary server.



Troubleshooting Broker

This chapter provides common troubleshooting tasks when using the WSM Intermediary.

Installation and Configuration Problems

Errors occurred during installation

Receive an error message at the end of the installation that:

The installation of HP SOA Manager is finished, but some errors occurred during the install. Please see the installation log for details.

Solution:

- 1 Check the <SOAM dir>/HP_SOA_Manager_2.52_InstallLog.xml log file for errors.
- 2 If you see install file errors, <action name="Install File" status="error" />, it means you only copied the HPSOAManagerInstaller.bin file from the SOA Manager installation CD to the system. You need to copy all of the files that are on the CD in the ../Installation directory to the system where you're trying to install the intermediary.

AutoPass fails to install

Receive an error dialog during installation that:

AutoPass, the HP Software licensing tool, failed to install properly. This installation will abort. Please refer to the <temp dir>\AutoPass_install.log log file for more details.

Solution:

- 1 Check to see if the <temp dir>\AutoPass_install.log log file exists.
- 2 If the log file exists, check for errors.

- 3 If the log file doesn't exist, check to see if there are non-English characters in the <temp_dir> name. AutoPass has a bug where it doesn't allow non-English characters in path names. If there are non-English characters in the <temp_dir> name:
 - a Uninstall Network Services.
 - b Save the value of the TMP environment variable.
 - c Change the TMP environment variable to a directory with all English characters.
 - d Install Network Services.
 - e Change the value of the TMP environment variable back to its original value.

Runtime Problems

Could not start monarch-sba

When trying to start the intermediary, receive a message:

```
[WARN] unable to locate tools.jar, possible non-sun jvm?
```

and later:

```
[SEVERE]; Could not start monarch-sba: java.lang.Exception: Monarch did not initialize.
```

Solution:

- 1 Verify that the environment variable MIP_JAVA_HOME is assigned to the Java 1.4 SDK and not the JRE.

When trying to start the intermediary, receive a message:

```
[SEVERE]; Could not start monarch-sba: java.lang.Exception: Monarch did not initialize.
```

Solution:

- 1 Turn on logging for the Smart Business Agent (SBA) to get more details about the problem.

f Change directories to <install_dir>/conf/broker.

g Edit the logging.properties file.

1. Change `log4j.category.com.hp.wsm.impact=OFF` to `log4j.category.com.hp.wsm.impact=INFO, ROLL_FILE`

2. Add the following to the end of the file

```
# ROLL_FILE - rolling file appender that writes the logs to
the file system
#
log4j.appender.ROLL_FILE=org.apache.log4j.RollingFileAppender
log4j.appender.ROLL_FILE.File=C:\\temp\\soam-broker-sba.log
```

```
log4j.appender.ROLL_FILE.MaxFileSize=512KB
log4j.appender.ROLL_FILE.MaxBackupIndex=1
log4j.appender.ROLL_FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.ROLL_FILE.layout.ConversionPattern=-->
%d{yyyyMMdd|HH:mm:ss} | %p | %t | %c{5} | %m%n
```

- 2 Restart the intermediary.
- 3 Look for errors in the C:\temp\soam-broker-sba.log file.

Failed to initialize listener

When trying to start the intermediary, receive a message:

```
...;SEVERE;An error occurred while initializing the MIP Server: ... :
failed to initialize listener
```

Solution:

- 1 Check to see if the Intermediary is already running. If you are running on Windows and selected to install the Intermediary as a service during the installation process, the Intermediary is automatically started when you reboot the system.
- 2 If the Intermediary is not running, then another application may be using the port. By default, the Intermediary uses port 9032. Change the Intermediary to use a different port.
 - a Change directories to <install_dir>/conf/broker.
 - b Edit the mipServer.xml file. Change the <entry name="com.hp.http.server.port">9032</entry> property.
 - c Start the Intermediary.

Unable to determine binding from message element

Receive the message when a request is sent to a custom intermediary service:

```
Unable to determine binding from message element: {xxx}yyy
```

Solution:

- 1 Verify that the request matches the binding specified in the WSDL.
- 2 Verify that the namespace in the request matches the namespace in the WSDL.

Authentication header not progressed to backend

The authentication header is not progressed to the backend service when a request is sent to a custom XML intermediary service.

Solution:

- 1 Verify that the `com.hp.wsm.sn.router.xml.handlers.outbound.SoapPassThroughTransportHeaderHandler` handler is configured for your custom XML intermediary service. This handler works for XML services even though it's called a SOAP handler.

Out of Memory

Receive an error that ran out of memory when running the Intermediary as a service.

Solution:

Increase the stack and heap sizes.

- 1 Modify the `<soam_dir>\bin\win32\services\service-manager.bat` file. Add the stack and heap parameters to the system properties (`@set SYS_PROPS=-Xms64m -Xmx256m -Dcom.hp.mip.autopass.home...`).
- 2 Run the bat file to remove the Intermediary service (`service-manager.bat -remove broker`).
- 3 Run the bat file again to add the Intermediary as a service with the new parameters (`service-manager.bat -install broker`).
- 4 Check that the new parameters are configured by looking in the registry under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services/broker<version num>`.

Receive an error that ran out of memory when running the Intermediary from the command line.

Solution:

Increase the stack and heap sizes.

- 1 Modify the `<install_dir>\bin\<unix | win32>\mipserver[.bat]` file. Increase the sizes for `-Xms` and `-Xmx`.
- 2 Restart the Intermediary.

A

- architecture
 - multiple brokers, 6-4
- audit publisher, 3-2
- auditing, 3-1
 - audit publisher, 3-2
- auditing handler, 1-2
- authentication
 - enabling, 7-10

B

- backup endpoint, 6-2
- broker
 - contextual overview, 1-1
 - install as Windows service, 5-3
 - management channel port, 5-4
 - SSL port, 7-8
 - starting, 5-1
 - stopping, 5-2
 - using multiple, 6-4
- broker configurator, 1-1
 - assign access, 5-6
 - starting, 5-2
- brokered service
 - auditing, 3-1
 - business content, 3-2
 - convert simple, 4-1
 - custom, 4-1
 - deploy, 2-2
 - edit, 2-3
 - fault logging, 3-1, 3-4
 - HTTP path, 2-3
 - overview, 2-1
 - performance metrics, 2-2
 - remove, 2-4
 - schema validation, 3-5
 - undeploy, 2-2

- version, 2-3
- view details, 2-1

- business content alerting, 3-2
- business content handler, 1-3

C

- common handlers, 1-2
- conceptual architecture
 - failover and load balancing, 6-2
 - multiple brokers, 6-4
- configure
 - audit publisher, 3-2
 - auditing, 3-1
 - brokered service HTTP path, 2-3
 - brokered service version, 2-3
 - business content alert, 3-2
 - failover and load balancing, 6-3
 - fault logging, 3-1, 3-4
 - HTTP, 5-3
 - key store, 7-7
 - schema validation, 3-5
 - SSL port, 7-8
 - trust store, 7-7
- contextual overview, 1-1
- counfigure
 - inbound message security, 7-11
 - inbound transport security, 7-9
 - outbound message security, 7-12
 - outbound transport security, 7-10
- custom handlers, 4-2

E

- endpoint
 - backup, 6-2
 - multiple in WSDL, 6-3
 - primary, 6-2
- environment variable, 5-1

F

- failover and load balancing
 - conceptual architecture, 6-2
 - multiple brokers, 6-4
 - overview, 6-1, 6-2
 - scenarios, 6-2
 - setup, 6-2
- fault logging, 3-1, 3-4

G

- generic soap contract handler, 3-3

H

- handlers
 - add to custom, 4-2
 - configuring, 3-1
 - custom, 4-2
 - overview, 1-2

HTTP

- brokered service URL, 2-3
- client settings, 5-5
- proxy settings, 5-6
- secure port, 7-8
- server port, 5-4
- server settings, 5-3
- threads, 5-5

I

- inbound message security, 7-11
- inbound transport security, 7-9
- installation problems, A-1
- Invocation handler, 3-4

K

- key store, 7-7

L

- logging
 - brokered service fault, 3-1, 3-4
- logging handler, 1-2

M

- management channel, 5-4
- message level security, 7-5
 - inbound processing, 7-6, 7-11

- outbound processing, 7-6, 7-12
- message trace, 3-1
- MIP_JAVA_HOME, 5-1
- monitoring handler, 1-2

O

- outbound message security, 7-12
- outbound transport, 7-10
- overview
 - architecture, 1-1
 - brokered service, 2-1
 - failover and load balancing, 6-1
 - security, 7-1

P

- payload, 3-1
- performance metrics, 2-2
- PKI, 7-2
- port, 5-4
 - management channel, 5-4
- prerequisites, 1-1
- primary endpoint, 6-2
- proxy settings, 5-6

R

- runtime problems, A-2, A-3

S

- SBA, 1-1
- schema validation, 3-5
- schema validation handler, 1-2
- security
 - feature matrix, 7-1
 - implement scenario, 7-9
 - inbound message, 7-11
 - inbound transport, 7-9
 - message level, 7-5
 - outbound message, 7-12
 - outbound transport, 7-10
 - overview, 7-1
 - scenarios, 7-2
 - setup components, 7-6
 - transport level, 7-4
- server port, 5-4

- service security inbound handler, 3-7
 - service version, 2-3
 - service-manager.bat, 5-3
 - settings
 - audit publisher, 3-2
 - HTTP, 5-3
 - key store, 7-7
 - SSL port, 7-8
 - trust store, 7-7
 - SOAP
 - endpoint, 6-3
 - SOAP contract handler, 3-7
 - SOAP dispatch handler, 3-7
 - SOAP monitoring handler, 3-8
 - SOAP pass-through transport header handler, 3-4
 - SOAP payload, 3-1
 - SSL, 7-4, 7-9, 7-10
 - enabling, 7-9
 - port, 7-8
 - stop broker, 5-2
- T**
- trace message, 3-1
 - transport level security, 7-4, 7-9, 7-10
 - troubleshooting
 - installation problems, A-1
 - runtime problems, A-2, A-3
 - trust store, 7-7
- U**
- URL
 - changing, 2-3
- W**
- Win32 service, 5-3
 - Windows service, 5-3
 - ws security message processing inbound handler, 3-9
 - ws security outbound handler, 3-8
 - WSDL
 - multiple endpoints, 6-3
- X**
- xml contract handler, 3-10
 - xml dispatch handler, 3-10
 - XPath monitoring handler, 3-10
 - XSLT handler, 3-11

