HP OpenView Service Oriented Architecture Manager

WSM Broker Administrator Guide

Version: 2.11

Windows, HP-UX, Linux

Aug 2006

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

Support

You can visit the HP OpenView web site at:

http://www.hp.com/managementsoftware/support

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

HP OpenView 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 support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter 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 log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

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

Contents

Table of Contents

3 Using Brokered Services 3-1

4 Using Custom Brokered Services 4-1

Introduction

This chapter contains overview information about the *WSM Broker Guide* and the WSM Broker. In particular, this chapter includes the following sections:

- Document Overview
- Contextual Overview
- Broker Configurator
- Common Handlers

Document Overview

The *WSM Broker Administrator Guide* provides instructions for setting up a Broker and brokered services for the purpose of managing Web services. The guide provides a brief contextual overview of the Broker and sequentially lists tasks that must be performed to deploy the broker and manage brokered services. Lastly, the guide provides instructions for configuring the broker's components and runtime behaviors.

The guide does not provide instructions for setting up an end-to-end management solution. The *HP OpenView Service Oriented Architecture (SOA) Manager Administrator Guide*, together with this guide, should be used to set up a complete Web Services Management (WSM) solution when using the WSM Broker.

Audience

The Administrator Guide is primarily intended for Systems Integrators and Systems Administrators who are responsible for integrating and enabling management components in their application environment.

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

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

Brokered services utilize the Broker'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 brokered services and configure handlers for brokered 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 Network Services and viewed using the Business Services Explorer.

➤ The SOA Manager currently implements the Web Service Management Framework (WSMF), which is an HP-authored precursor version of standard WS-based management protocols.

Figure 1-1 provides a high-level overview of the architecture and context of the WSM Broker.

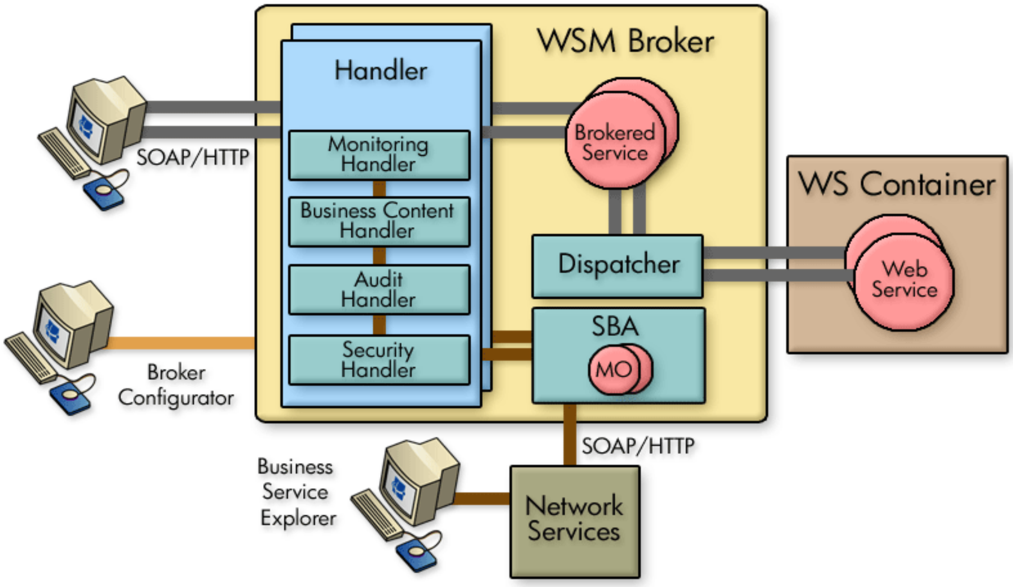


Figure 1-1: Contextual Overview

Broker Configurator

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

Common Handlers

As shown in Figure 1-1, 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 brokered service are described below. Some handlers are enabled by default when you create a brokered service, while other handlers must be manually enabled. In addition, custom brokered 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 Broker’s log messages. The log messages can be used to troubleshoot any problems that occur with the Broker.

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 brokered services. Authentication and Authorization is provided by HP OpenView Select Access integration.

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 OpenView** | **SOA Manager 2.1** | **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 OpenView SOA Manager v2.1 Broker" installed.
```

> The script configures the HP SOA Manager 2.1 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 the Network Services 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.

> The SOA Manager also integrates with Select Access, which can be used to secure access to the Broker's console. See Chapter 9 "Integrating with Select Access" in the *SOA Manager Administration Guide* for more information.

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 OpenView 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 OpenView installation and data directories. You will only be prompted for this information if this is the first time you have installed an HP OpenView product.

The default value for the installation directory is `C:\Program Files\HP OpenView` on Windows and `/opt/OV` on Unix. The default value for the data directory is `C:\Program Files\HP OpenView\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 OpenView 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 OpenView 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 OpenView Cross Platform tool, `ovconfchg`. After installing the The HP OpenView Cross Platform Component, this program is in the bin directory of the HP OpenView 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 OpenView 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 the BSE

To change a Broker's log levels from the BSE:

1 From the BSE main tool bar, click **IT Services**. The IT Services screen displays and the Summary tab is selected.

2 From the list of **WS Intermediary Services**, click the IT service you want to view. The WS Intermediary Service view screen displays.

3 From the Contained Resources section, click the resource you want to view. The Resource View screen displays.

4 Click **Edit/Query Log Levels**.

5 Specify `MIP` for the logger. You can also set the log level for individual packages. The Broker packages begin with `com.hp.ov.mip`.

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 Network Services 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 OpenView data log directory.

Using XPL Tracing

SOA Manager uses the HP OpenView Tracing tools for tracing. Please refer to the *HP OpenView 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 OpenView 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 OpenView\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
```

3

Using Brokered Services

This chapter provides instruction for managing the lifecycle of brokered services. Topics include creating, editing, deploying, viewing, and removing a brokered service when using the WSM Broker. In addition, an overview of brokered services is provided.

Overview

A brokered service is created for each Web service that you want to manage. The Broker Configurator is used to create and manage the lifecycle of a brokered service. Requests for managed Web services are sent to the brokered service and then forwarded (dispatched) to the actual service's endpoints. Brokered services can be created for both SOAP/HTTP and XML/HTTP Web services.

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

Brokered services are used to manage Web services when you want to:

- Interpose manageability for Web services that are deployed in a WS container that does not offer native manageability.
- Separate the management of Web services from the services' implementation.
- Provide message-level and transport-level security when a WS container does not include native security features.

Creating a SOAP/HTTP Brokered Service

A SOAP/HTTP brokered service is created using a Web service's definition (WSDL) file. As part of the creation process, an identical WSDL is created and written to the <*install_dir*>`\conf\broker` directory.

➤ Before creating a brokered service, make sure you have deployed at least one Web service in your WS container and know the location of its WSDL.

To create a SOAP/HTTP brokered service:

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. In the Browse local WSDL file text box, enter the path to a WSDL file or click **Browse...** to locate a Web service's WSDL. For example:

   ```
   C:\services\FinanceService.wsdl
   ```

   ```
   /usr/etc/FinanceService.wsdl
   ```

   Or, in the Specify Remote WSDL URL text box, enter the URL to a Web service's WSDL. For example:

   ```
   http://myhost:myport/myservice?wsdl
   ```

4. 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.
5. The **Encoding** check box allows you to change the encoding of the request XML. By default, the check box is selected and the request XML is UTF-8 encoded before it is sent to the final endpoint. If you want to retain the incoming XML encoding, clear the check box.
6. By default, the 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. If the Web service only contains one endpoint, the endpoint is used whether it is a backup or not. For more information on load balancing and failover, see Chapter 6 "Implementing Load Balancing and Failover".
7. If you want the Broker to send authentication information when communicating with the endpoint, click to select the **Send Credentials** check box. The credentials will need to be verified by the Web Services Container that is hosting the endpoint. For more information on security, see Chapter 7 "Using the Broker's Security Features".
8. Click **next** to move to Step 3 of the wizard (Step 3: Configure Brokered Service).
9. From the Service section, accept the default service name and version, or enter a new name and version.

   ➤ After a brokered service is created you cannot change the Service name.

10. Accept the default handler configuration or configure the brokered service using the fields provided. A check in the option field indicates that a handler is selected and is enabled. The handler configuration options are detailed in Chapter 5 "Configuring Handlers". See Chapter 7 "Using the Broker's Security Features" for detailed instructions if you want to secure communication to/from the brokered service.
11. Click **finish**. The brokered service is created and automatically deployed. The Brokered Services screen displays and lists the brokered service as well as its status.
12. Verify that the service is operational by clicking the brokered service WSDL endpoint listed in the Service Interface (WSDL) column. The WSDL for the service displays.

13 Repeat this procedure to create additional brokered services.

Creating an XML/HTTP Brokered Service

Brokered service can be created for XML/HTTP services to support legacy Web services that were created prior to SOAP. The Broker Configurator provides a wizard that creates an XML/HTTP Brokered service.

To create an XML/HTTP brokered service:

1 From the Broker Configurator's main toolbar, click **List Services**.

2 Click on the **Create XML Service** link. Step 1 of the wizard displays (Step 1: Create Service).

3 Complete the following fields:

- **Service Name**: The Web service name.
- **Service Namespace**: The Web service's Target Namespace.
- **Encoding**: The check box allows you to change the encoding of the request XML. If selected the request XML is UTF-8 encoded before it is sent to the final endpoint. Do not select this check box if you want to retain the incoming XML encoding.
- **Service Endpoint**: The URL of the Web service.

4 Click **next** to move to Step 2 of the wizard (Step 2: Configure Service).

5 From the Service section, accept the default Service Name and Version, or enter a new Service Name and Version.

> After a brokered service is created you cannot change the Service Name.

6 Accept the default configuration or configure the custom brokered service using the fields provided. The configuration options are detailed in Chapter 5 “Configuring Handlers”. See Chapter 7 “Using the Broker's Security Features” for detailed instructions if you want to secure communication to/from the brokered service.

7 Click **finish**. The custom brokered service is created and automatically deployed. The Brokered Services screen displays and lists the brokered service as well as its status. The Style field indicates that the brokered service is `Simple XML Service`.

8 Verify that the service is operational by clicking the brokered service WSDL endpoint listed in the Service Interface (WSDL) column. The WSDL for the service displays.

9 Repeat this procedure to create additional custom brokered services.

Viewing Brokered Service Details

The Service Details screen allows you to view the details of a brokered service. The details include the brokered service definition and endpoint, performance data, the Web service's endpoints, and features (handlers) configuration.

To view a brokered service's details:

1 From the Brokered Services screen, find the brokered service that you want to view.

2 From the Name column, click the brokered service's name. The Service Details screen displays. The brokered service's details are listed in different sections. The Features section also 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 a brokered service. The metrics include: the Average Response Time, Total Requests, Successes, and Failures. These metrics are provided to get a general view of how a brokered service is performing. The full set of performance metrics are displayed in the Network Services server when the brokered service is managed as part of a business service.

Undeploying a Brokered Service

A brokered service that is undeployed is inactive, but is not removed from the Brokered Service list. The brokered service is not available for requests until it is deployed. You can configure a brokered service that is undeployed; however, you cannot view any of its management data.

Any Web service management data that has been collected is lost when a brokered service is undeployed.

To undeploy a brokered service:

1 From the Brokered Service list, find the brokered 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 a Brokered Service

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

To deploy a brokered service:

1 From the Brokered Service list, find the brokered 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 brokered service WSDL endpoint listed in the Service Interface (WSDL) column. The WSDL for the service displays.

Editing a Brokered Service

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

To edit a brokered service:

1 From the Brokered Service list, find the brokered service that you want to edit.

2 From the Action column, click the **edit** link. The Edit Service screen displays.

3 From the Edit Service screen, edit the brokered service using the fields provided. The handler configuration options are detailed in Chapter 5 “Configuring Handlers”. See Chapter 7 “Using the Broker’s Security Features” for detailed instructions if you want to secure communication with the brokered service.

4 Click **Save**. The Brokered Services screen displays and the brokered service is automatically deployed. The deployment is complete when the status changes to `Operational`.

Changing a Brokered Service's Version

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

To change a brokered service's version:

1 From the Configurator's main toolbar, click **List Services**. The Brokered Services screen displays.

2 From the Action column, click the **edit** link for the brokered service. The Edit Service screen displays.

3 From the Service section, select the Version field and enter a version number for the brokered service.

4 At the bottom of the screen, click **Save**. The Brokered Services screen displays and the brokered service is automatically deployed. The deployment is complete when the status changes to `Operational`.

Configuring a Brokered Service's HTTP Path

A brokered service's HTTP Path is the path that will be used by a client to invoke the managed Web service. For example, if the Broker agent is installed on "MyHost.com" and the default broker 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 brokered service is created. Changing the HTTP path of a brokered service is useful when multiple brokered 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:

1. From the Configurator's main toolbar, click **List Services**. The Brokered Services screen displays.
2. From the Action column, click the **edit** link for the brokered service. The Edit Service screen displays.
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 Brokered Services screen displays and the brokered service is automatically deployed. The deployment is complete when the status changes to `operational`.

Removing a Brokered Service

When a brokered service is removed, it is deleted from the Brokered Service list. In addition, the service definition (WSDL) for the brokered service is deleted from the <*install_dir*>`\conf\broker` directory.

To remove a brokered service:

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

4

Using Custom Brokered Services

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

Overview

Custom brokered services are similar to simple brokered 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 brokered service is also available for a custom brokered service. However, simple brokered services use a predefined set of handlers, while custom brokered services are boundless. That is, the handler chain can be customized to include a broad range of handlers (including custom handlers). In addition, the ordering of the handlers in the handler chain can be configured.

The benefits of using a custom brokered service include:

- 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 (i.e., multiple business metric handlers)

Convert a Simple Brokered Service

Custom brokered services are created by first creating a simple brokered service (see Chapter 2) and then converting the simple brokered service to a custom brokered service. You can convert both SOAP/HTTP and XML/HTTP simple services to custom services.

To convert a simple brokered service to a custom brokered service:

1 From the Brokered Services list, find the brokered service that you want to convert.

2 From the Action column, click **edit**. The Edit Service screen displays.

3 Click **Convert**. The Edit Custom Service screen displays and lists the handlers for the custom service. Any handlers that were configured for the simple brokered service are also configured for the custom service. Several default handlers, which were part of the simple brokered service but not previously visible, are now listed.

4 Click **Save**. The Brokered Service screen displays and the brokered service is automatically deployed. The deployment is complete when the status changes to `Operational`. The Style field indicates that the brokered service is `Custom`.

Adding Handlers

As mentioned in the “Overview” section, using custom brokered services provides greater control when adding handlers for a brokered service. Handlers are assigned to a custom brokered 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 brokered service:

1 From the Service list, find the custom brokered service that you want to edit. The Style field indicates that the brokered service is `Custom`.

2 From the Action column, click **edit**. The Edit Custom Service screen displays and displays the handlers currently assigned to the brokered 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 5 “Configuring Handlers” for a detailed description of each handler.

4 Click **Save**. The Brokered Services screen displays and the brokered service is automatically deployed. The deployment is complete when the status changes to `Operational`.

Adding Custom Handlers

Custom brokered services allow you to add your own custom handlers to a brokered service’s handler chain. In order to add a custom handler, you must first create the custom brokered service and then edit the service’s definition file located in the brokered service jar file.

To add a custom handler:

1 Uncompress <*install_dir*>`\conf\broker\`<*broker_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 (i.e., `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 brokered service including the custom handler class and any dependent classes.

8 Place the jar in <*install_dir*>`\conf\broker\`. The brokered service is automatically deployed. You can use the Broker Configurator to verify that the jar has been deployed. The brokered service is listed on the Service List and its status is `Operational`.

5

Configuring Handlers

This chapter provides a reference of the management handlers available for brokered services. The reference is listed in alphabetical order and includes a description of each handler as well as the handler's fields. Where applicable, example entries for the fields are provided. This reference can be used when you are editing or creating a brokered service.

When using custom brokered 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. In addition, the auditing feature can collect a message's SOAP payload. The information collected is sent to the Network Services server and is stored in a database. The BSE is used to query the database to retrieve audit information. Moreover, 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 Administrator Guide*.

Fields

- **Include detailed traces**: Used to capture profile data. This feature captures the outcome of a Web service invocation as it passes through each handler in the handler chain for a brokered 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 brokered 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 brokered services.

Configuring the Audit Publisher

The Audit Publisher is a Broker 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:

1 Stop the Broker 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 Broker.

Business Metric Alerts Handler

Business content alerting allows you to define a business metric for specific content that is found in the SOAP request and/or response message for a service (i.e., 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, BSE console, etc...). For more information on the 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 BSE.

Fields

- **Name**: Enter a user friendly name to identify the alert. (i.e., `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. (i.e., `getInfo`)
- **Alert applies to**: Select when you want the broker 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. (i.e, `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: (i.e., **Name**: name **Xpath**: `//s:Envelope/s:Body/t:InfoRequest/t:symbol/text()`
- **Namespace Prefixes**: Enter any namespace prefixes that appear in the XPath expression (i.e., **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. In addition, operations are set as the runtime soap payload element. When using this handler, no WSDL is required in the Broker 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 broker (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>
```

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 brokered services.

Fields

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

Log Handler

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

Fields

Category: This field is only available for custom brokered services. This field allows you to select a specific log category where log messages are sent. This field is optional. For more information on the Broker’s Logging features, see “Using the Logging Feature” in Chapter 2.

Schema Validation Handler

Schema validation ensures that SOAP requests conform to a Web service’s WSDL. If the schema validation feature is enabled, then 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 Broker. If the schema validation feature is disabled, then 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.

Security Auditing

The Security Audit Handler is used to collect security trace information (used for non-repudiation, etc.) and sends the payload to a security provider. For example, when using Select Access to control authorization, the traces can be viewed using the Select Access Audit Report Viewer.

Fields

- **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. For Select Access, the security provider will log audit messages to the Enforcer using `SOAP Messages.INFO`. See the Select Access documentation for information on how to configure Audit Policies and Servers.

To configure a security provider, add the following property in <*install_dir*>`\conf\broker\mipServer.xml`. This example sets Select Access as the security provider.

```
<entry name="com.hp.mip.security.audit.provider">SelectAccess</entry>
```

Service Security Inbound Handler

The Service Security Inbound Handler performs authorization using the principal and credentials associated with an operation. The authorization is done using a configured security provider such as Select Access. This handler is used in conjunction with, and must come after the WS Security Message Processing Inbound handler. In addition 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 brokered 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 Broker'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.
- **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/.`

Ws Security Outbound Handler

The Ws Security Outbound Handler provides support for ws-security on outbound messages (i.e., from the Broker to a WS container). This includes user name/password, signing, and encryption.

Fields

- **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 then signed. This field is only relevant when using the `Cert` method and when the 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, then the Soap Contract Handler detects the operation, and then the Service Security Inbound Handler uses the credentials and operation to perform authorization.

Fields

- **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, then the response will not be signed or encrypted. If set to true, then inMsgEncrypted and inMsgSigned will apply and the response will be signed and/or encrypted.
- **inMsgSSOEnable**: This field is 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 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 brokered services for XML services.

Xml Dispatch Handler

The XML Dispatch Handler is used to dispatch a request to the Broker's Dispatcher component, which is responsible for forwarding a request to a Web service's endpoint. The handler is used for custom brokered 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 brokered 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 (i.e., **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 brokered service JAR file in order to be loaded by the Broker'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.

6

Implementing Load Balancing and Failover

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

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

Overview

The WSM Broker contains a load balancing and failover feature that automatically routes a Web service request that is made to a brokered 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 a brokered service is created using the Broker Configurator console. When a Web service with multiple endpoints is managed, the management information (i.e., success, response time, etc…) 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

Load balancing and failover share the same common architecture shown in Figure 6-1. All requests that are sent to a brokered service are sent to a final endpoint using the Broker's dispatcher. A list of available endpoints is registered with the Broker 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 a brokered service is created from the WSDL file, these endpoints are discovered and registered by the Broker and configured as either an active endpoint or a backup endpoint.

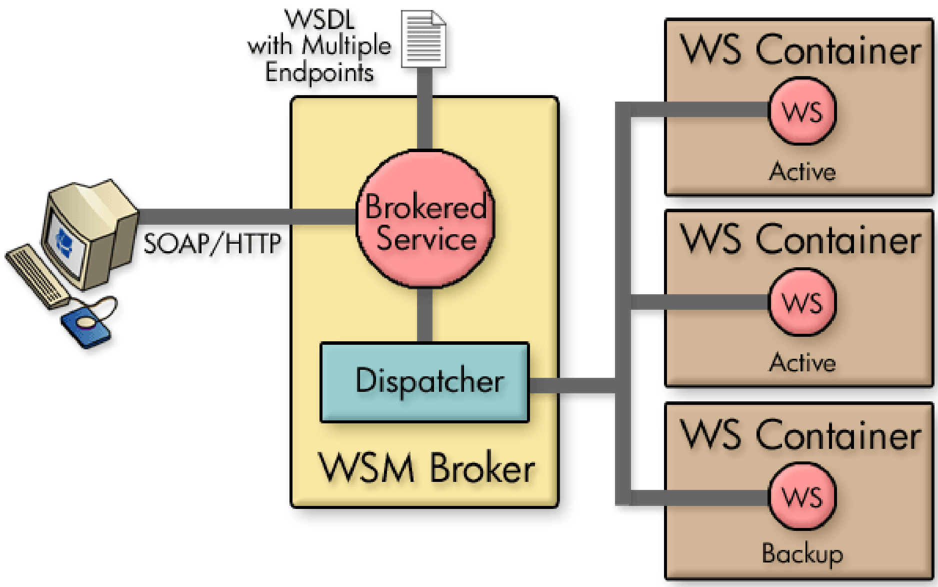


Figure 6-1: Load Balancing and Failover

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 Broker 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 the HTTP service that is provided by your WS Container supports the `Retry-After` HTTP header property, the Broker uses the interval specified by this property instead of the 30 second retry interval.

➤ 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 brokered service that you create. When you create a brokered 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 a brokered service using the Broker Configurator, make sure you have modified a WSDL to include multiple endpoints as demonstrated above.

Configuring Load Balancing and Failover

A brokered service is created by using the Broker Configurator. The create service wizard steps you through the process of creating a brokered 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 a brokered service is initially created.

Using Multiple Brokers

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

Each Broker contains a brokered service for the same Web service. Loads are balanced between each brokered service and if one Broker fails, additional Brokers are available to continue servicing requests. Management information (i.e., success, response time, etc…) for each brokered service is aggregated. In addition, each brokered service can be viewed separately in a single business service when using the BSE.

Figure 6-2 below shows a conceptual architecture when using two Brokers. More than two Brokers can be used depending on your requirements. When implementing this scenario, use the instructions in the "Setting Up Load Balancing and Failover" section discussed previously for each installation of the WSM Broker.

> 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.

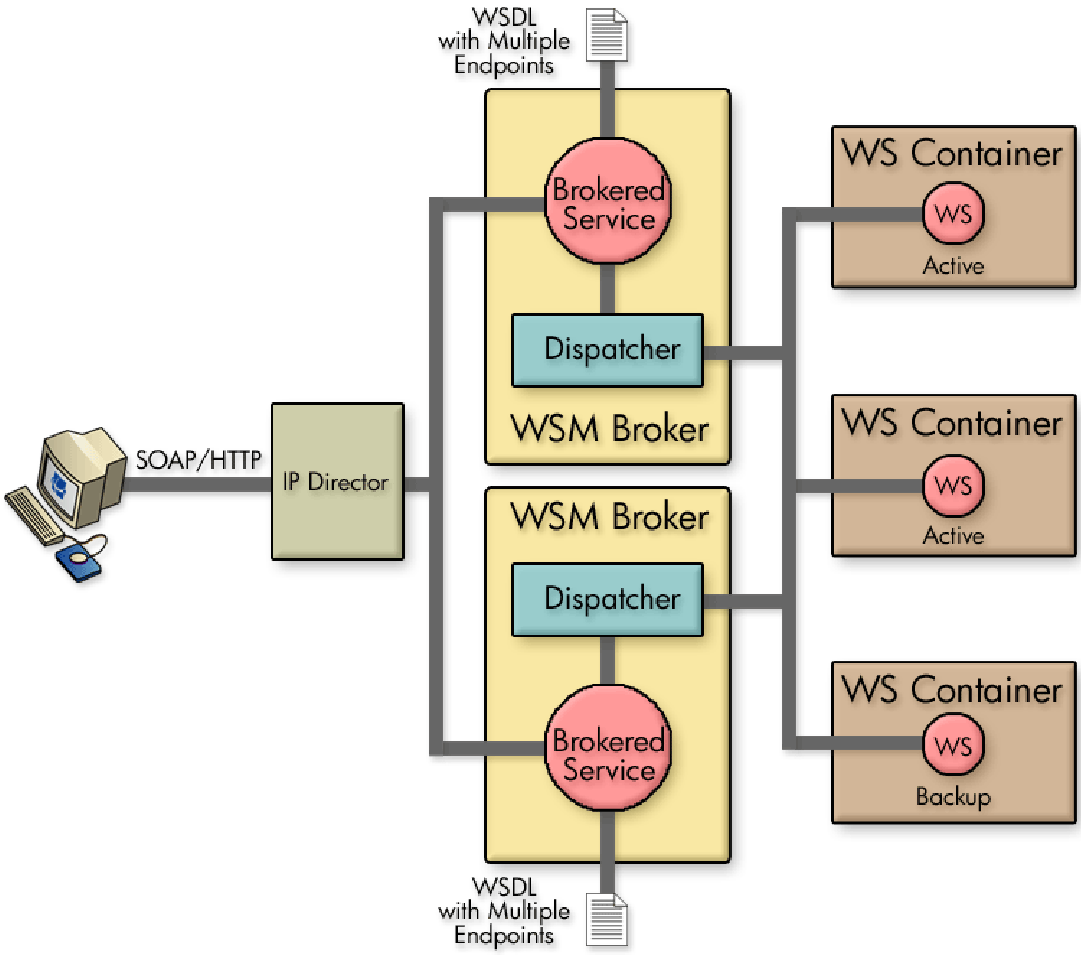


Figure 6-2: Using Multiple Brokers for Load Balancing and Failover

7

Using the Broker's Security Features

This chapter provides instructions for securing the Web services application channel when using a WSM Broker 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 Broker. If you are using a WSM deployment scenario that uses the WSM Agents (J2EE Agent or .NET Agent), then you can implement the security features natively provided by the WS Container (WLS or IIS). However, you can use such deployment scenarios in conjunction with the WSM Broker and thus leverage the security features that are provided with the Broker 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 Broker 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 Broker security solution.

Table 7-1: Supported Security Technology Stack

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

- All User Identity Management – authentication, authorization, and administration is deferred to enterprise security products. SOA Manager currently integrates with HP OpenView Select Access.
- WS-Security implementation in the Broker (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 Broker security implementation. There are three basic security scenarios discussed:

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

Figure 7-1 below shows a high level view of the Broker security implementation and includes all three scenarios.

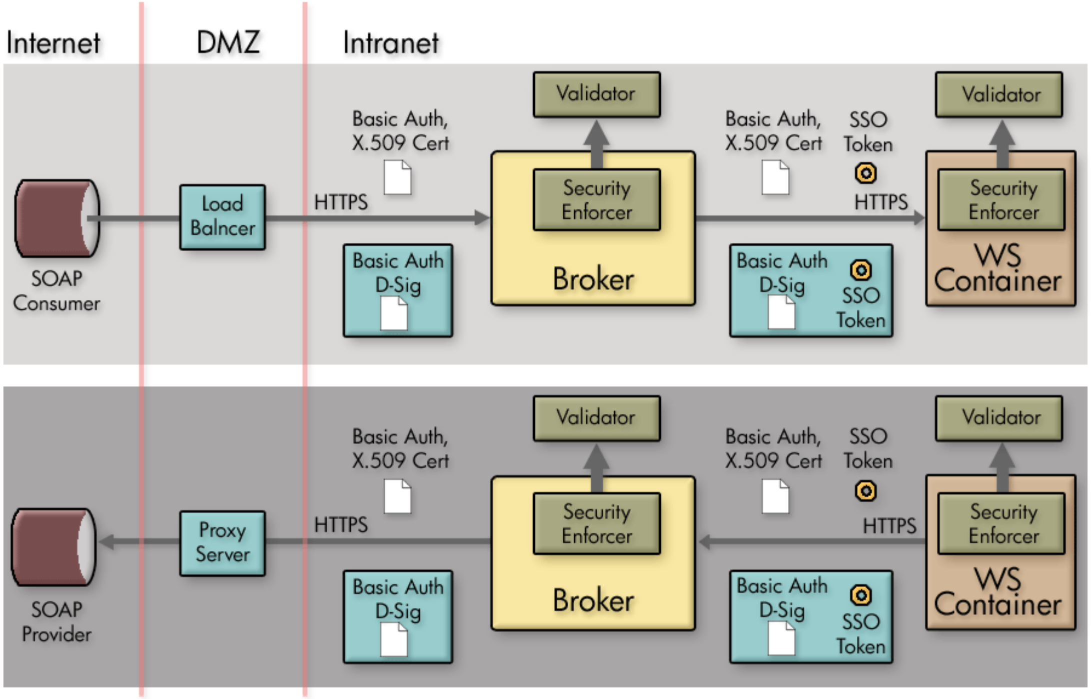


Figure 7-1: Security Implementation

Scenario 1: Broker is the Entry Point for External Consumers

In this scenario, incoming HTTP/S traffic through the firewall is front-ended by the Broker. The Broker supports HTTP/S basic authentication and X.509 client certificate authentication over SSL. Alternately, the broker 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. The actual authentication/authorization is delegated to security products such as Select Access.

Authentication/Authorization failures are tracked and sent to the Network Services 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 broker, 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 broker. 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 broker, but the Broker presents some normalized identity to the back end Web service implementation. For example, some things like user, broker, 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 broker and Web service implementation. However, it does make the back end implementation secure. The Broker (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 Broker. The Broker 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. Broker 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 Broker 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 Broker 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 Broker can be configured to use any of these for authentication against the security provider. In this case, the Broker behavior is no different than that specified in Scenario 1, where it accepted calls from external consumers.

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

Scenario 3: Broker 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, Broker 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 Broker. 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 brokered service can be configured with transport security options for inbound traffic. Figure 7-2 shows a common view of transport level security.

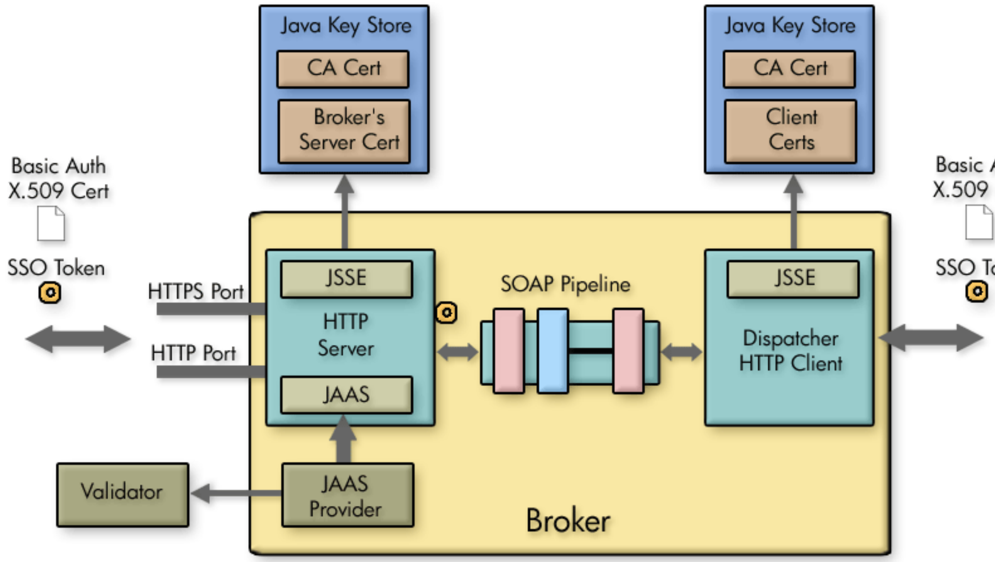


Figure 7-2: Transport Level Security

Message Level Security

Message level security is offered using SOAP handlers. Figure 7-3 shows a common view of message level security.

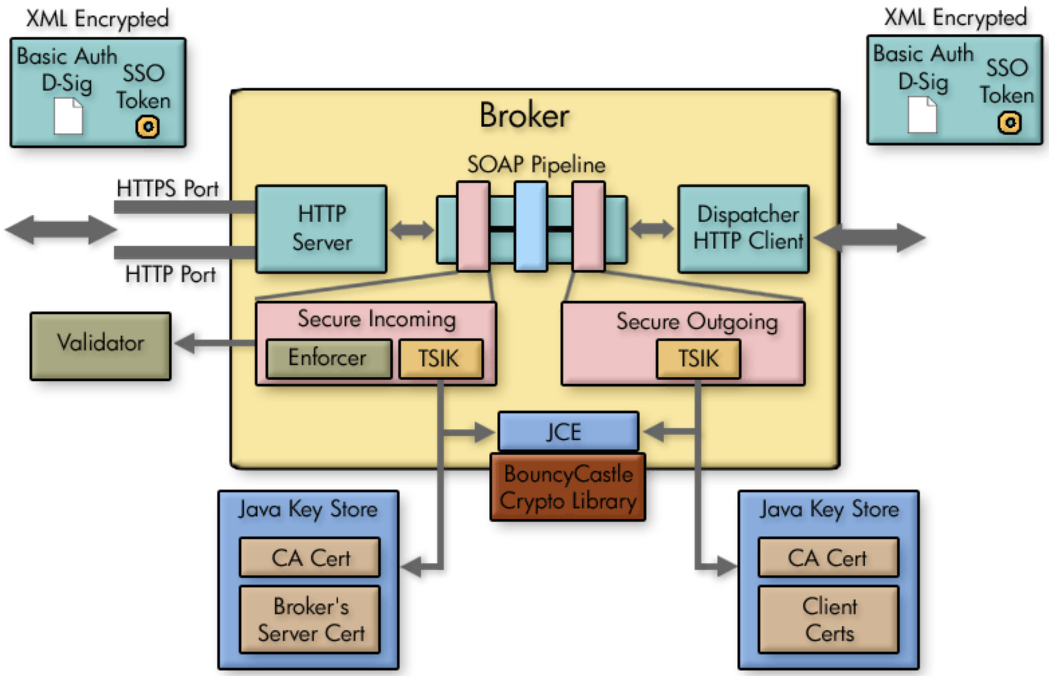


Figure 7-3: Message Level Security

Inbound Message Processing

Inbound request payload can be decrypted using the Broker'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
- authentication/authorization is delegated to Select Access APIs

Outbound Message Processing

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

➤ A Key Store is required in the following steps. The Broker 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 Broker 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 broker 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 Broker's SSL Port

The Broker'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 Broker'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 Broker 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 broker supports basic authentication and authorization using basic authentication and x.509 client certificates. For either scenario, you can implement authentication and authorization for all brokered services, specific brokered services, or for specific operations within a brokered 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 Broker security solution provides authentication and authorization services on a best of breed approach by integrating to well known and proven enterprise security products. At the present time integrations are provided with Hewlett-Packard's Select Access

Using Select Access

The Broker security solution supports integration with Select Access. Select Access provides an identity management solution for securing access to IT services and resources. To complete the Select Access instructions in this section, you must have:

- Completed the Select Access integration instructions in Chapter 9 of the *SOA Manager Administrator Guide*
- A general understanding of the Select Access Policy Builder
- General understanding of the WSM solution
- A WSM Broker deployed and running
- Access to a Web service endpoint deployed and running

Setting Up Basic Authentication Only

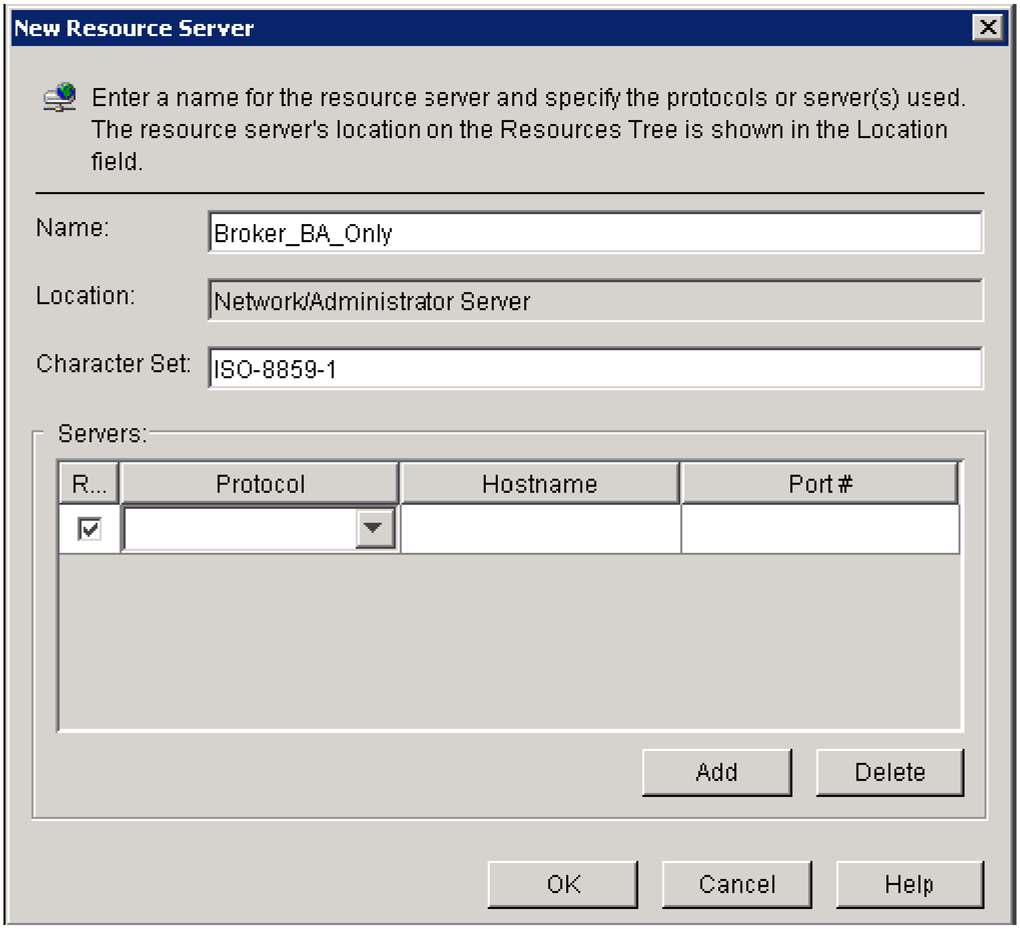
The instructions in this section are used to set up Select Access to provide Basic Authentication Only for users that are accessing resources on the Broker's application channel. The instructions use the Select Access Administration console. The Select Access Administration console is used to define resources that are to be secured as well as create polices and permissions for those resources. If you are not familiar with Select Access, you may need to consult the Select Access documentation while completing some of the instructions in this section.

Define a Select Access Resource Server for the Broker

You must create a Select Access resource server that is mapped to the Broker server. This Select Access resource server is used to control access to the server using basic authentication only. The resource server contains the protocol, host name, and port number of the Broker.

To define a Select Access resource server for the Broker,

1. From the Select Access Policy Builder Resources Tree, right-click Resource Access and select **New | Folder**. The New Folder dialog box displays.
2. In the Name field, enter a name for the folder.
3. Click **OK**. When asked to clear the Policy Validator cache, select **OK**. The folder is created and is added to the Policy Builder Resources Tree under Resource Access.
4. Right-click the newly created folder and select **New | Resource Server**. The New Resource Server dialog box displays.
5. In the Name box, enter a name for this new resource server (i.e., Broker_BA_Only). Any name that clearly identifies the server can be used.
6. On the bottom of the window click **Add**. A new entry displays under the Servers section.



New Resource Server

Enter a name for the resource server and specify the protocols or server(s) used. The resource server's location on the Resources Tree is shown in the Location field.

Name: Broker_BA_Only

Location: Network/Administrator Server

Character Set: ISO-8859-1

Servers:

R...	Protocol	Hostname	Port #
☑			

7 Enter the following server information for the server where the Broker is located:

- **Protocol**: The protocol used to access the Broker (`HTTP`)
- **Hostname**: <*broker_host_name*>
- **Port #**: The Broker's port number used for HTTP connections (`9032`)

> Only one server is added to this resource server and must match the settings in the `selectaccess.properties` file.

8 Click **OK** to save this resource server. When asked to clear the Policy Validator cache, select **OK**. The resource server is listed in the Policy Builder Resources Tree.

Mapping a Select Access Resource for Basic Authentication Only

You must create a Select Access resource that is mapped to the Broker server’s Select Access security settings. This Select Access resource is used to control access to the server using Basic Authentication only for HTTP transport level security.

A property entry in `selectaccess.properties` sets the authentication resource path in Select Access:

```
#SelectAccess service authentication resource path
AuthenticationResource = /authentication
```

To define a Select Access Resource for Basic Authentication Only:

1 From the Policy Builder Resources Tree, select the resource server that contains the Broker Server that you want to secure using Basic Authentication Only.

2 Right-click on the resource server, and select **New | Resource** from the menu. The New Resources dialog box displays.

3 In the Name field, enter the authentication-only resource path as pre-defined in the `AuthenticationResource` property in the `selectaccess.properties` file. For example, `authentication`.

4 Click **OK**. When asked to clear the Policy Validator cache, select **OK**. The resource is listed in the Policy Builder Resources Tree under the appropriate broker resource server. Any requests to the Broker are authenticated but no authorization is performed.

Setting up Basic Authorization

The instructions in this section are used to set up Select Access to provide basic authorization for users that are accessing resources on the Broker's application channel. The instructions use the Select Access Administration console. The Select Access Administration console is used to define resources that are to be secured as well as create polices and permissions for those resources. If you are not familiar with Select Access, you may need to consult the Select Access documentation while completing some of the instructions in this section.

Define a Select Access Resource Server for the Broker

You must create a Select Access resource server that is mapped to the Broker server. This Select Access resource server is used to control access to the server using basic authorization. The resource server contains the protocol, host name, and port number of the Broker.

To define a Select Access resource server for the Broker:

1 From the Select Access Policy Builder Resources Tree, right-click Resource Access and select **New | Folder**. The New Folder dialog box displays.

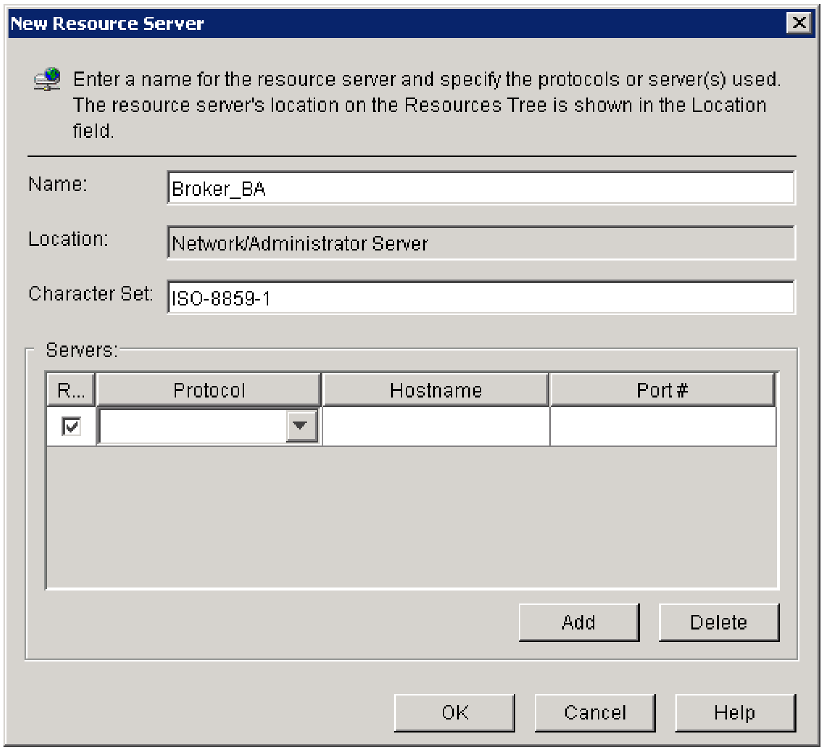
2 In the Name field, enter a name for the folder.

3 Click **OK**. When asked to clear the Policy Validator cache, select **OK**. The folder is created and is added to the Policy Builder Resources Tree under Resource Access.

4 Right-click the newly created folder and select **New | Resource Server**. The New Resource Server dialog box displays.

5 In the Name box, enter a name for this new resource server (i.e., Broker_BA). Any name that clearly identifies the server can be used.

6 On the bottom of the window click **Add**. A new entry displays under the Servers section.



Enter a name for the resource server and specify the protocols or server(s) used. The resource server's location on the Resources Tree is shown in the Location field.

Name: Broker_BA

Location: Network/Administrator Server

Character Set: ISO-8859-1

Servers:

R...	Protocol	Hostname	Port #
☑			

7 Enter the following server information for the server where the Broker is located:

— **Protocol**: The protocol used to access the Broker (`HTTP`)

— **Hostname**: <*broker_host_name*>

— **Port #**: The Broker's port number used for HTTP connections (`9032`)

Only one server is added to this resource server and must match the settings in the basic authentication section in the `selectaccess.properties` file.

8 Click **OK** to save this resource server. When asked to clear the Policy Validator cache, select **OK**. The resource server is listed in the Policy Builder Resources Tree.

9 Use the Identities Tree to apply the basic authorization rules for users of this resource server.

Setting Up X.509 Authorization

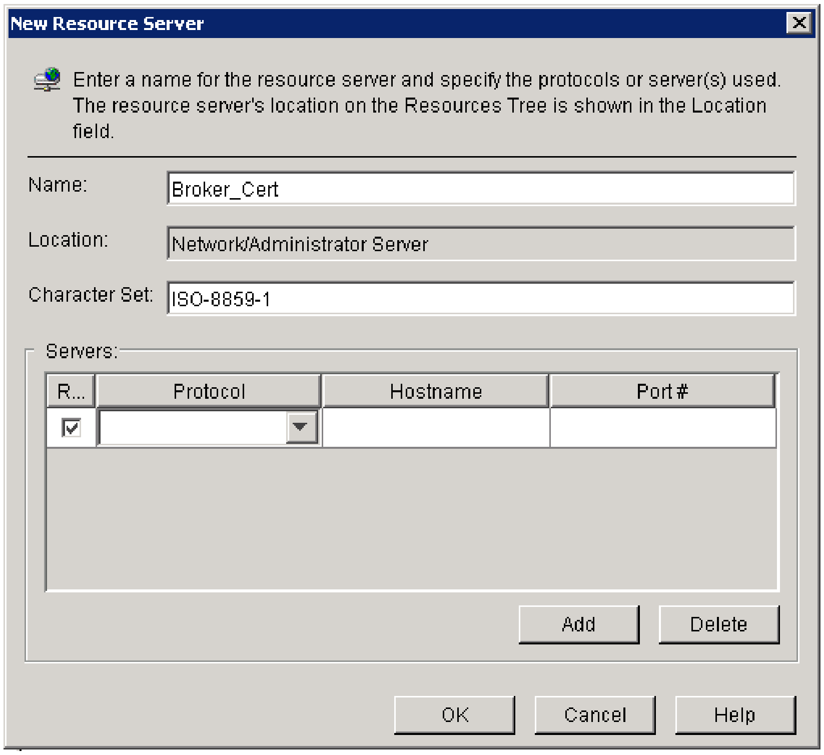
The instructions in this section are used to set up Select Access to provide x.509 authentication for users that are accessing resources on the Broker's application channel. The instructions use the Select Access Administration console. The Select Access Administration console is used to define resources that are to be secured as well as create polices and permissions for those resources. If you are not familiar with Select Access, you may need to consult the Select Access documentation while completing some of the instructions in this section.

Define a Select Access Resource Server for the Broker

You must create a Select Access resource server that is mapped to the Broker server. This Select Access resource server is used to control access to the server using certificate-based authorization. The resource server contains the protocol, host name, and port number of the Broker.

To define a Select Access resource server for the Broker:

1. From the Select Access Policy Builder Resources Tree, right-click Resource Access and select **New | Folder**. The New Folder dialog box displays.
2. In the Name field, enter a name for the folder.
3. Click **OK**. When asked to clear the Policy Validator cache, select **OK**. The folder is created and is added to the Policy Builder Resources Tree under Resource Access.
4. Right-click the newly created folder and select **New | Resource Server**. The New Resource Server dialog box displays.
5. In the Name box, enter a name for this new resource server (i.e., Broker_Cert). Any name that clearly identifies the server can be used.
6. On the bottom of the window click **Add**. A new entry displays under the Servers section.



New Resource Server

Enter a name for the resource server and specify the protocols or server(s) used. The resource server's location on the Resources Tree is shown in the Location field.

Name: Broker_Cert

Location: Network/Administrator Server

Character Set: ISO-8859-1

Servers:

R...	Protocol	Hostname	Port #
☑			

7. Enter the following server information for the server where the Broker is located:
   - **Protocol**: The secure protocol used to access the Broker (`HTTPS`)
   - **Hostname**: <*broker_host_name*>
   - **Port #**: The port number used for SSL connections (i.e., `443`)

Only one server is added to this resource server and must match the settings in the certificate-based authentication section in the `selectaccess.properties` file.

8 Click **OK** to save this resource server. When asked to clear the Policy Validator cache, select **OK**. The service is listed in the Policy Builder Resources Tree.

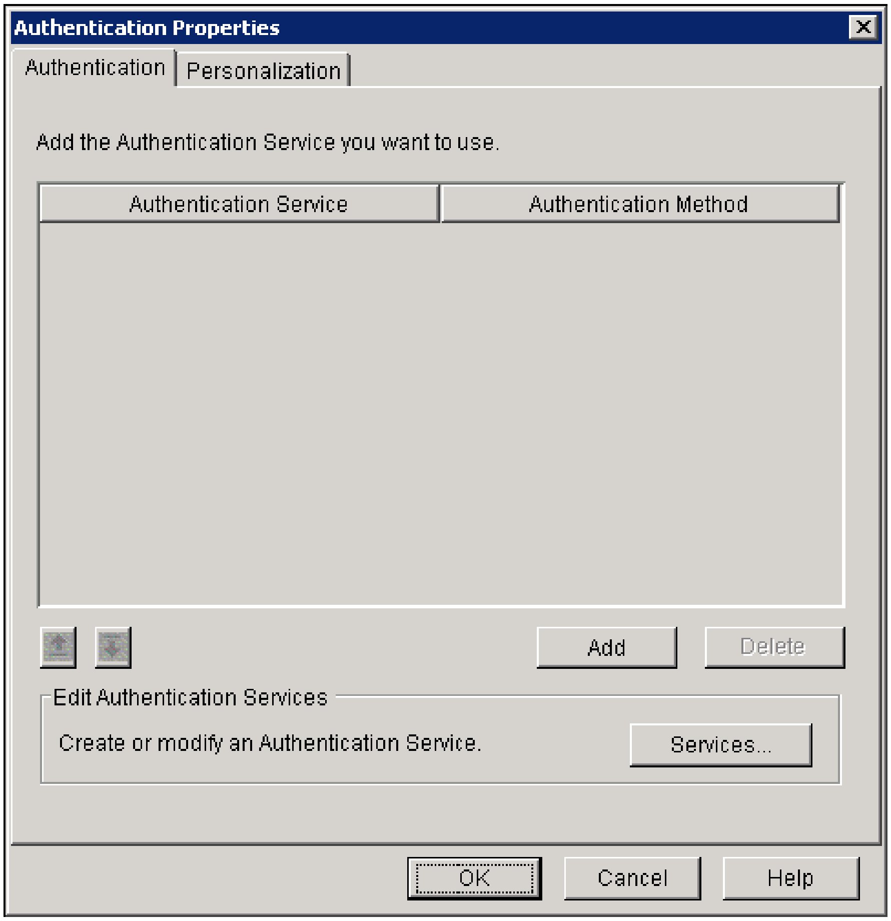
9 Use the Identities Tree to apply the x.509 rule for users of this resource server.

Enable Select Auth for Basic Authorization and X.509 Certificate Authorization

The Policy Builder Identities Tree contains a Select Auth column that must be enabled for each broker resource server for which you want to enable authorization.

To enable Select Auth for Basic and X.509 authorization:

1 From the Policy Builder Identities Tree, right-click the first column on the same row as the resource server for Basic Authorization, select **Enable Select Auth** from the pop-up menu. The Authentication Properties dialog box displays.



Authentication Properties

Add the Authentication Service you want to use.

Authentication Service	Authentication Method

Edit Authentication Services

Create or modify an Authentication Service.

2 Click **Add**. The Available Authentication Services dialog box displays.

3 Select the **password** authentication service and click **Add**. The service is listed in the Selected Services column.

4 Click **OK**. The Authentication service is added to the list of authentication services in the Authentication Properties dialog box.

5 Click **OK**. When asked to clear the Policy Validator cache, select **OK**. The Select Auth icon shows that Select Auth for the selected resource server is enabled.

6 Repeat this procedure for a resource server for X.509 Certificate Authorization.

Mapping Resources in Select Access

A Select Access resource server provides authorization on a global basis. In other words, for every brokered service configured within the Broker to use authorization, the resource server is used to authorize users. These services have the same group of consumers authorized to use them.

There may be circumstances where you want to identify a specific consumer or group of consumers to be authorized to use specific brokered Web services or specific brokered Web service operations. To do this, you must create resources in Select Access that maps to the brokered Web services or brokered Web service operations in the Broker and then apply user permissions to the resources.

To map resources in Select Access:

1 Login to the Select Access Policy Builder.

2 From the Policy Builder Resources Tree, select the resource server that contains the resources that you want to secure. Make sure the server you select represents the authorization type you are using (basic authorization or certificate authorization).

3 Right-click on the resource server, and select **New | Resource** from the menu. The New Resources dialog box displays.

4 In the Name field, provide a name for this resource which maps to a name of a resource on the Broker. The name is different depending on whether you are implementing transport-level or message-level security. In addition, if you are securing a specific operation, the name must include the operation name.

— If you are authorizing at the transport level, the name must match the brokered service's URL path as defined in the HTTP header. For example, if a brokered service has the URL: `HTTP://`*<host_name>*`:`*<port>*`/FinanceServiceProxy`, then the resource name would be `FinanceServiceProxy`. If you do not know the URL used to access to the brokered service, you can use the Broker Configurator's Service detail screen to get a brokered service's URL value.

Web services are typically associated with a port binding that is used as part of the URL which is passed in the message header. For example:

`HTTP://`*<host_name>*`:`*<port>*`/FinanceServiceProxy/`
`FinanceServiceSoapBinding`

➤ If your client invokes a brokered service using the port binding in the address, you must include the binding in the name for the resource. For the above example, the resource name would be `FinanceServiceSoapBinding`. This URL is defined in the WSDL for a brokered service. You can use the Broker Configurator's Brokered Services screen to view a brokered service's WSDL. The URL is typically located near the bottom of the WSDL.

— If you are authorizing at the message level, the resource name must match the Web service name that is defined in the WSDL for the brokered service. If you are not sure of the Web service name, you can use the Broker Configurator's Brokered Services screen to view the WSDL for a brokered service. The service name is typically located towards the bottom of the screen in the `<wsdl:service>` element.

5. Click **OK**. The resource is saved and is listed on the Resources Tree.
6. Use the Users Tree policy matrix to apply access permissions to this resource.
7. Repeat this procedure to create and map additional resources.

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 Broker as well as outbound communication from the Broker to a WS Container.

Before implementing a security scenario, you must configure the security components that are used by the Broker (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 Broker'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 Broker accepts requests from consumers using SSL and authenticates/authorizes the user using a security provider such as Select Access. 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. Refer to Figure 7-2 for a conceptual architecture of transport-level security.

Enabling SSL

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

To enable inbound SSL:

1 From the Broker Configurator, create a new or edit an existing brokered 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 displays. 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 a brokered service and enable authentication for inbound transport security. Users are authorized using a security provider such as Select Access (See "Using Select Access" above). You can enable authentication when you create a brokered service or you can edit an existing brokered service.

To enable authentication:

1 From the Broker Configurator, create a new or edit an existing brokered 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 Broker need to be authenticated using a user name and password. Select Access is used to verify the credentials and which resources can be accessed.

— **X.509 Client Certs**: All requests to the Broker need to be authenticated using an X.509 certificate. Select Access is used to verify the credentials and which resources can be accessed.

3 At the bottom of the screen, click **Save Changes**. Once this service is deployed, the Broker will communicate with Select Access 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 Broker will return a 404 Not Authorized error.

Outbound Transport Security

In this scenario, the Broker 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. Refer to Figure 7-2 for a conceptual architecture of transport-level security.

> When using outbound SSL Security, a Web Service deployed in a WS Container must be configured to use SSL from within that WS Container. See your WS Container documentation for more instructions on setting up SSL communications.

Enabling Outbound SSL

The Broker Console is used to configure a brokered service and enable outbound SSL connections. You must enable SSL when you create a brokered service. You cannot edit an existing brokered 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 brokered service by following the prompts. The brokered service is configured to use outbound SSL when you have completed creating the brokered 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 Broker, the SSL handshake will fail. Make sure the endpoint's CA's Certificate is located in the Broker's trust store.

Inbound Message Security

In this scenario, a consumer must authenticate with the Broker before messages are accepted. In addition, the consumer may choose to encrypt messages before sending them to the Broker; in which case, the broker will decrypt the messages before they are dispatched to the final endpoint. Refer to Figure 7-3 for a conceptual architecture of message-level security.

The Broker Configurator is used to configure an inbound message security handler for a brokered service. Users are authorized using a security provider such as Select Access (See "Using Select Access" above) and decryption is implemented through a Key Store (See "Configure a Key Store" above). You can enable message security when you create a brokered service or you can edit an existing brokered service.

To enable inbound message security:

1. From the Broker Configurator, create a new or edit an existing brokered 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 Broker need to be authenticated using a user name and password. Select Access is used to verify the credentials and which resources can be accessed.
   - **Digital Signature Authentication**: All messages to the Broker need to be authenticated using a digital signature. Select Access is used to verify the credentials and which resources can be accessed.
   - **Digital Signature Authentication with Decryption**: All messages to the Broker need to be authenticated using a digital signature. In addition, the Broker's private key is used to decrypt the message. Select Access is used to verify the credentials and which resources can be accessed while the Broker's Key Store is used to manage the private key used for decryption.
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 broker 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 Broker will communicate with Select Access 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 Broker will return a 404 Not Authorized error.

➤ The Broker 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 Broker.

Outbound Message Security

In this scenario, The Broker must authenticate itself with a WS Container before messages are processed at the WS Container. The WS Container and the Broker can share the same security provider or a WS Container's security provider is used to complete the authentication. In addition, the Broker can encrypt messages before sending them to the WS Container; in which case, the WS Container must be able to decrypt the messages. Refer to Figure 7-3 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 WS Container'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 a brokered service or you can edit an existing brokered service.

To enable outbound message security:

1 From the Broker Configurator, create a new or edit an existing brokered 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 WS Container need to be authenticated using a user name and password. The WS Container's security provider is used to verify the credentials and which resources can be accessed. Enter a valid Username and Password for your WS Container in the fields provided.

— **Sign**: All messages dispatched to a WS Container will include a digital signature. The Broker's Key Store is used to sign the outbound message.

— **Sign and Encrypt**: All messages dispatched to a WS Container will include a digital signature and will be encrypted. The Broker's Key Store is used to sign the outbound message. In addition, the Broker's private key must be located at the WS Container 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 broker 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 broker management channel. This functionality is the same as securing the application channel.

To configure broker management channel security:

1. Stop the Broker 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 Broker 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 Broker server.

8

Troubleshooting

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

Installation and Configuration Problems

Errors occurred during installation

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

```
The installation of HP OpenView 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_OpenView_SOA_Manager_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 broker.

AutoPass fails to install

Receive an error dialog during installation that:

```
AutoPass, the OpenView 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 broker, 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 broker, 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.RollingFileAppend
er
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.PatternLay
out
log4j.appender.ROLL_FILE.layout.ConversionPattern=-->
%d{yyyyMMdd|HH:mm:ss}|%p|%t|%c{5}|%m%n
```

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

Failed to initialize listener

When trying to start the broker, receive a message:

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

Solution:

1. Check to see if the Broker is already running. If you are running on Windows and selected to install the Broker as a service during the installation process, the Broker is automatically started when you reboot the system.
2. If the Broker is not running, then another application may be using the port. By default, the Broker uses port 9032. Change the Broker 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 Broker.

Unable to determine binding from message element

Receive the message when a request is sent to a custom brokered 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 brokered service.

Solution:

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

Select Access enforcer cannot connect to validator

Receive the message

```
Error for /console/auth/j_security_check
java.lang.NoClassDefFoundError:
com/hp/wsm/sn/common/security/auth/selectaccess/RouterTransaction
```

Solution:

1 The `mip-addons.jar` file is missing.

2 On the Broker system, create an `addons` directory under `<install_dir>/lib`.

3 From the SOA Manager CD, copy `/Addons/mip-addons.jar` to `<install_dir>/lib/addons`.

4 Refer to the “Integrating with Select Access” section in the *Administrator Guide* for more details about the Select Access integration.

XML message not being passed to Select Access

The XML message is not being passed to Select Access in a custom XML service.

Solution:

1 Verify that an XML introspection handler is configured in the custom XML service’s handler list. This handler is needed since the broker does not pass XML requests to Select Access automatically.

2 Refer to the Creating an “XML Introspection Service” section of the *Integrator Guide* for details.

Out of Memory

Receive an error that ran out of memory when running the Broker 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 Broker service (`service-manager.bat -remove broker`).

3 Run the bat file again to add the Broker 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 Broker 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 Broker.

A

Appendix A Creating a Java Key Store

Java Key Stores are used to create secure communication channels based on the SSL standard. Key Stores are created using the Java Keytool utility, which is a key and certificate management tool provided by Sun Microsystems and distributed with the Java Development Kit. This appendix is designed to act as a fast track to creating a Java Key Store intended to be used with the security scenarios introduced in Chapter 7 “Using the Broker’s Security Features” for the specific purpose of securing the communication channel to and from the Broker. This tutorial is NOT intended to be a replacement to the documentation provided by Sun as it pertains to the Java Keytool. For detailed information on the Java Keytool, see Sun’s documentation. To complete the tasks in this appendix, you must have:

- A general understanding of the Public Key Infrastructure
- A general knowledge of the Sun JDK
- Installed the JDK and insured it is located on the computer’s PATH

Step 1: Create a Private Key and the Initial Java Key Store File (JKS file)

Open a command prompt and execute the following command:

```
keytool -genkey -keystore scream2.jks -alias scream -keyalg RSA
```

This will start the creation of the private key and Key Store. Provide answers to the prompts as they appear. You will receive prompts similar to the following:

Enter Key Store password:

This is the password you will use to access the Key Store in the future.

What is your first and last name?

This is the identity of the owner of the Key Store. Enter the fully qualified DNS name of the server.

➤ When asked for your first and last name, use the fully qualified domain name of the system you will use this Key Store on. Failure to do this will result in a failure of the SSL connection.

What is the name of your organizational unit?

Enter any departmental information you want associated with the Key Store.

What is the name of your organization?

Enter the name of the organization this Key Store will be associated with.

What is the name of your City or Locality?

Enter you city name.

What is the name of your State or Province?

Enter state or province.

What is the two-letter country code for this unit?

Enter country code. (e.g. US, UK, etc…)

The next prompt is a review prompt displaying the information you just entered. If everything is correct, type `y` and press Enter. If you need to make corrections, press Enter and follow the prompts.

If you typed `y` above to continue, you will be asked for a key password. This is a password that will be associated with the private key only. You can use the same password you provided to for the Key Store if desired. It is a matter of personal preference.

You should know have a Key Store created.

Step 2: Generate a CSR request

Execute the following command:

```
keytool -certreq -keystore scream2.jks -alias scream -file
scream2.csr
```

Enter the Key Store password when prompted. When completed, you should have a CSR to send to a certificate authority.

Step 3: Obtain a Signed Certificate from a Certificate Authority

Using the CSR file created in step 2, contact a Certificate Authority to obtain a digital certificate for your server.

If you are in a test environment, you can obtain a test certificate. Test certificates are generally easier and quicker to obtain. If using a test certificate, ensure that the test root certificate is also obtained from the CA you are using.

Below are some URLs to Certificate Authorities:

VeriSign

http://www.verisign.com

Thawte

http://www.thawte.com

GlobalSign

http://www.globalsign.net/

http://www.belsign.be/

Step 4: Import Signed Server Certificate to Key Store

Once you obtain a signed certificate from a Certificate Authority, you need to import that certificate to the Java Key Store. Copy the certificate file you received from the authority to same directory as your Java Key Store.

Edit the file so that all of the text above `-----BEGIN CERTIFICATE-----` is removed. This is approximately 45 lines of text which you will remove. When completed, you should be left with something with looks like the following:

```
-----BEGIN CERTIFICATE-----
MIIDtDCCAx2gAwIBAgIBETANBgkqhkiG9w0BAQQFADCBmzELMAkGA1UEBhMCVVMxEzARB
gNVBAgTCk5ldyBKZXJzZXkxFTATBgNVBAcTDE1vdW50IExhdXJlbDEgMB4GA1UEChMXSG
V3bGV0dC1QYWNrYXJkIENvbXBhbnkxDTALBgNVBAsTBERSUEUxEjAQBgNVBAMTCUplZmY
gVHVjazEbMBkGCSqGSIb3DQEJARYManP2332tAaHAuY29tMB4XDTA0MDMzMDE3MTUzM1o
XDTA1MDMzMDE3MTUzM1oweTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCk5ldyBKZXJzZXkx
IDAeBgNVBAoTF0hld2xldHQtUGFja2FyZCBDb21wYW55MQ0wCwYDVQQLEwREUlBFMSQwI
gYDVQQDExtzY3JlYW0uYW1lcmljYXMuaHBxY29ycC5uZXQwgZ8wDQYJKoZIhvcNBBEEBQ
ADgY0AMIGJAoGBANeRXdQBWOXbVjbmsL0kmf9Ql0nq9mvJh7ehZsbNZQN2wspcLYrfb1v
4769BxbegdW/uY9LnNP0a3vVLg2hLWWx8L703SLd7S/ztZF8QU/RAE1w6pxDT+KsHwtfn
OAlBj2FEcChrIEQI2PVUIcw8PpQ/HAMNRj7DVEvR2Po9B5wHAgMBAAGjggEnMIIBIzAJB
gNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2
F0ZTAdBgNVHQ4EFgQUNpLKwfcQM0GP219/V5I3H81smGwwgcgGA1UdIwSBwDCBvYAU9Uj
i64FvhWKvwHlB0LdGi+Q+FKhgaGkgZ4wgZsxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpOZ
XcgSmVyc2V5MRUwEwYDVQQHEwxNb3VudCBMYXVyZWwxIDAeBgNVBAoTF0hld2xldHQtUG
Fja2FyZCBDb21wYW55MQ0wCwYDVQQLEwREUlBFMRIwEAYDVQQDEwlGDTWmIFR1Y2sxGzA
ZBgkqhkiG9w0BCQEWDGp0dWNrQGhwLmNvbYIBADANBgkqhkiG9w0BAQQFAAOBgQBXKfZ2
oQCqf5mWyiqJ3bZpcFamNmHtoXlBkZmgIx5D9ITD0PJ+eQaeRzFS1Pphv2rrYvddpsAs7
sjXjTSNXNjNNCnAZTsvFB7j8wKFQObPT6XmgevJ2kVwEIFoYxpNKGoZYPyCBkopEHR5KX
z+ClPA/z7+iqnB9iAmV/Pgib9Obg==
-----END CERTIFICATE-----
```

To insure you do not overwrite the original file, save this file with a slightly different file name and close the file.

Execute the following command:

```
keytool -import -keystore scream2.jks -alias scream -file
scream2_import.crt –trustcacerts
```

Ensure that the alias name is the same as the private key name.

This should result in a response from the keytool similar to the following:

```
"Certificate was added to keystore"
```

You now have a successfully configured Key Store.

If you have a test certificate or a certificate issued by an authority which is a commonly known public authority, you will need to ensure that the Certificate is installed on all client trust stores in order for the connection to be created. In addition, the CA root certificate needs to be installed in the server Trust Store as well.

Index