

HP Service Oriented Architecture Manager

Integration Guide

Version: 2.51

Windows®, HP-UX, Linux



December 2007

© Copyright 2004-2007 Hewlett-Packard Development Company, L.P.

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

© Copyright 2004- 2007 Hewlett-Packard Development Company, L.P., all rights reserved.

Trademark Notices

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

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

UNIX® is a registered trademark of The Open Group

Support

You can visit the HP Software support web site at:

www.hp.com/go/hpsoftwaresupport

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

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

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

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

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

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

Table of Contents

Introduction	1-1
Document Overview	1-1
Audience	1-1
Integration with SOA Manager – An Overview	2-1
Integration Points	2-1
Prerequisites for Integration	2-2
Startup Tasks	2-2
HP SOA Manager	2-2
Web Services Integration	2-2
Database Integrations	2-3
HP SOA Manager Policy Enforcement Intermediary Group	2-3
Java API Integrations	2-3
Security Customization	2-4
Integration of SOA Manager with Other products	2-4
HP Select Access Integration	2-4
HP Operations Manager Integration	2-4
HP BPI Integration	2-5
Prerequisites for Integration	3-1
Startup Tasks for SOA Manager and Policy Enforcement Intermediary	3-1
StartupTask Interface	3-1
Configuration File (server.xml)	3-2
Manually Implementing Intermediary Web Services	3-3
Creating an Intermediary Web Service JAR	3-3
Writing the Intermediary Web Service Definition	3-3
Creating a Custom Handler	3-5
Adding Custom Handlers to an Intermediary Web Service Definition	3-7

Deploying an Intermediary Web Service Jar	3-7
Exposing Web Service Entities by Using SOA Manager.....	4-1
Use Case 1	4-3
Use Case 2	4-4
Use Case 3	4-4
Discovery Operations	4-4
Configuration Operations	4-5
Performance Operations	4-5
Alert Operations	4-5
Alert Severity Types	4-6
Audit Operations	4-7
EventPull Port Type	4-7
EventPush Port Type	4-7
Operations for Managing HP SOA Manager	4-8
Status Parameters	4-8
Business Service Operations	4-9
PEP Operations	4-9
Publishing the Model.....	4-10
Developing a Policy Enforcement Agent Using Southbound Interfaces.....	5-1
Managed Objects and Port Types.....	5-2
Service Managed Objects and Port Types.....	5-2
Managed Object Implementation Details for SOA Manager Agent.	5-3
SOA Manager Interfaces for Policy Enforcement Agent Groups	5-3
SOA Manager Interfaces for Services.....	5-5
Operation Details and Schema Used by the Operations.....	5-5
ManagedObjectIdentityPT:GetManagementWsdUrl()	5-6
ManagedObjectDiscoveryPT:GetSupportedRelationships()	5-6
ManagedObjectDiscoveryPT:GetRelationships()	5-6
ManagedObjectDiscoveryPT:GetSpecificRelationships()	5-6

PublishSubscribePT:GetSupportedEventTypes()	5-6
PublishSubscribePT:publishSubscribe()	5-7
PublishSubscribePT:CancelSubscription()	5-9
ManagedObjectConfigurationPT	5-9
ServiceConfigurationPT:GetOperationalWsdUrl (for Service MO)	5-10
Performance:Get (Only for Web Service Container)	5-10
Get Request Schema	5-12
Get Response Schema	5-12
Response Schema	5-12
Error Handling	5-13
PerformancePT and PerformanceExtPT: (Only for Service MO)	5-18
Steps to Create a New SOA Manager Policy Enforcement Agent	5-18
Upgrading SOA Manager Agent from 2.0 to 2.1	5-20
Upgrading the Agent by Implementing the Get operation	5-20
Upgrading the Agent by Implementing the Audit Trace Notification	5-20
Database Integration	6-1
Overview	6-1
Database Schema Reference	6-1
MESSAGE_TRACE Table	6-1
MESSAGE Table	6-3
Sample SLA Reports	6-4
Sample SLA Reports for Month of Nov 2004	6-5
Select Access Integration	7-1
Configuring Instructions	7-1
Security Customizations	7-1
Creating an XML Introspection Service	7-1
Role-based Security Access (Implementing Custom Authentication based on User Profile)	7-3
HP Operations Manager Integration	8-1
Overview	8-1
Conceptual Architecture	8-2
Installation and Configuration	8-2

Requirements.....	8-2
Software Requirements	8-3
Hardware Requirements	8-3
Patches	8-3
Kernel Parameters	8-4
Installing the Plug-in.....	8-5
Setting Environment Variables	8-5
Configuring the Plug-in.....	8-5
Creating a SOAM Message Group.....	8-6
Create a Message Group.....	8-7
Modify the Operator User Profile.....	8-7
Starting the Plug-in.....	8-7
Stopping the Plug-in.....	8-9
Installing the Java Console	8-9
Starting the Java Console	8-9
Uninstalling the Plug-in	8-10
Performing Standard Management Functions.....	8-10
Service Management	8-10
Starting the HP SOA Manager Web Interface.....	8-11
Alert Message Management	8-11
Acknowledging Alert Messages	8-11
Changing Message Properties.....	8-12
Using XPL Logging	8-12
Configuring Log Levels	8-13
Using JRE Properties File.....	8-13
Using the XPL Properties File.....	8-13
Troubleshooting.....	8-13
Startup Errors.....	8-13
Message Errors.....	8-14
HP BPI Integration	9-1

Integration Instructions.....	9-1
Configuration Instructions.....	9-2
On the HP BPI Server.....	9-2
On the SOA Manager Server.....	9-3
Prerequisites for HP BPI- SOA Manager Adapter Installation.....	9-3
Installing the HP BPI- SOA Manager Adapter.....	9-4
Stopping the SOA Manager Adapter.....	9-5
SOA Manager Adapter Log Files	9-5
Troubleshooting HP BPI Integration.....	9-5
HP SOA Systinet Integration	10-1
Mapping Artifacts- SOA Manager and Systinet.....	10-3
Publishing Artifacts from Systinet to Registry	10-3
Retrieving Business Services from the Registry Using SOA Manager	10-3
Mapping SOA Services Model to UDDI	11-1
Overview	11-1
Role of UDDI in the SOA.....	11-1
Technical Policy Mapping.....	11-2
Web Service Mappings	11-3
Appendix A Product Compatibility Matrix	A-1
Product Compatibility Matrix.....	A-1
Index	I-1

Introduction

The *Integration Guide* is intended for users who want to take advantage of the HP Service Oriented Architecture (SOA) Manager's integration possibilities. SOA Manager Integration is often performed to either customize how the SOA Manager works or to reuse the SOA Manager's assets within other enterprise applications.

Much attention has been given to ensure that the SOA Manager's architecture is flexible and easy to integrate with. The architecture is pluggable and uses popular industry standards such as Java and XML. More important, the SOA Manager is among the first distributed management software products to use a services-based architecture. For example, Web services are used to expose many parts of the SOA Manager's management model. This practice not only ensures standard and open integration possibilities, it clearly sets the SOA Manager within current and future SOA-based environments.

Document Overview

The *SOA Manager Integration Guide* provides instructions for performing integrations with the SOA Manager. The guide provides a brief contextual overview of the customization and integration possibilities. Generally, each chapter is dedicated to an integration possibility and is self contained. Therefore, the book need not be read in sequence.

Every effort has been made to explain general concepts. However, much of the content in this guide assumes that the user is already familiar with the SOA Manager and has a basic level of experience. If you are new to the SOA Manager product, it is recommended that you first become familiar with the product. A good starting point is the *SOA Manager User Guide* and the *SOA Manager Tutorials*. These and other documents are located in the distribution in the /Documentation directory.

Audience

The *Integration Guide* is primarily intended for solution architects, systems integrators, and application developers who are responsible for integrating and enabling management components in their environments.

Integration with SOA Manager – An Overview

Integration Points

SOA Manager provides several integration points. The integration points can facilitate the deployment of Web services management solution as well as provide an opportunity to repurpose some of the management data that is collected for Web services. In some cases, there is also the ability to customize the solution with management features that are not supported with the solution, but important to fulfill the management needs of a Web service deployment.

In general, the integration points have been organized into the categories listed below. This is purely an organizational scheme and does not suggest any functional boundaries within the product.

- SOA Manager includes the following integration points:
 - Database – This integration point is used to create custom audit reports or to include audit information within other management applications.
 - Web services interfaces – This integration point is used to create new (or augment existing) enterprise management applications by reusing the SOA Manager’s management data and management model. The data and model are exposed as management Web services.
- SOA Manager policy enforcement intermediary – The intermediary supports integration with Java APIs. This integration helps you create custom management handlers that address the specific Web service management requirements of an organization.
- Integration with other products – You can integrate HP SOA Manager with the following products:
 - HP Select Access
 - HP Operations Manager
 - HP Business Process Insight (BPI)

Prerequisites for Integration

These steps are used to manually create SOA Manager's assets as well as customize the HP SOA Manager and the policy enforcement intermediary or agent startup. The prerequisites for integration are as follows:

- Customize startup behaviors.
- Automate repetitive tasks.
- Save time when updating HP SOA Manager assets over multiple installations.
- Reuse current tool sets (IDE, Content Versioning System, and so on).
- Maintain current development processes.

Startup Tasks

HP SOA Manager policy enforcement intermediary group, and policy enforcement agent group are server processes that have the ability to execute user-defined Java code at startup. This is useful for initializing any required services when the processes are started or executing any type of process initialization code.

Integrators are responsible for creating the user-defined Java code as well as configuring each server to use the custom code. Configuration of startup classes is done in the servers' XML configuration file (`server.xml`).

HP SOA Manager

You can integrate with HP SOA Manager by using published Web services interfaces or databases.

Web Services Integration

The most common and robust method for integrating with HP SOA Manager is by using the SOA Manager's published Web services interfaces. The Web services are SOAP based Web services that are defined using WSDL. The Web services follow standard Web services management protocols.



Two Web services management specifications are: Web Services Distributed Management (WSDM) and Web Services for Management (WS-Management). At present, HP SOA Manager software utilizes the Web Services catalog, which is an HP authored precursor to the standard WS management protocols. HP SOA Manager software supports these specifications as they mature and stabilize.

Web services integrations allow system architects to leverage current management investments and provide a broader and more thorough view of the enterprise. In general, Web services integrations are used to:

- Link HP SOA Manager with other enterprise management products to create composite and custom applications
- Create and/or reuse current management consoles to display the SOA Manager's management data

- Create custom management consoles

Database Integrations

Web services audit trace messages are persisted to a central database. This information in conjunction with the model relationships which are also populated into the database (such as which Business Service contains which Web services) can be used to enable reporting and analytical applications such as SLA reporting, billing, non-repudiation, forecasting, and so on. Any application can potentially connect to the SOA Manager database and make use of the data. For example, the database can be used to create audit reports using packages like Crystal Reports.

Integrators are responsible for creating and maintaining database connections from within their applications. In addition, integrators are responsible for upgrading their applications as the SOA Manager's database schema changes.

HP SOA Manager Policy Enforcement Intermediary Group

The Broker Configurator tool is typically used to create intermediary groups, configure intermediary group management handlers, and deploy intermediary groups. These steps can become repetitive and time consuming depending on the number of intermediary groups that are being deployed. However, these tasks can be completed without using the Broker Configurator.

Integrators are responsible for creating the intermediary group definition file (a proprietary file written using XML), packaging the intermediary group as a Java Archive (jar file), and copying it to the appropriate directory on the policy enforcement intermediary or agent group. Depending on the requirements, some or all of these tasks can be automated.

Java API Integrations

Policy handlers implement management logic that is used to interpose visibility and controls on Web services. These handlers are inserted in the HTTP or SOAP pipeline that is responsible for processing request and response messages. Intermediary groups use a set of standard handlers and can also be configured to use a set of simple or advanced handlers. However, an organization may have special management requirements that are not covered by any of the provided handlers.

In such cases, the SOA Manager's Java API can be used to create custom handlers that can implement any management or processing logic that is required.

Integrators are responsible for:

- Creating the custom handler Java logic by extending a base class
- Compiling the handler
- Configuring the handler in the intermediary group definition file
- Packaging the handler in the intermediary group jar file

Security Customization

The SOA Manager's Java API integration point is used to create custom security implementations that allow an organization to enforce security policies that are not covered by the default security features provided by the policy enforcement intermediary. This includes custom authentication based on user profile, custom security handlers, and an XML Introspection service.

Integration of SOA Manager with Other products

The following sections include a brief overview of the integration features provided by HP SOA Manager with other HP products.

HP Select Access Integration

It is possible to customize the policy enforcement intermediary or agent group security features including the integration with Select Access. In general, there are two use cases that are covered in this chapter:

- Passing custom query parameters as part of a Select Access authorization query
- Checking personalization attributes that are returned as a result of an authentication query

Both use cases can be implemented either at the message level or the transport level. Typically, the customization occurs within the policy handler chain by either creating custom handlers or customizing existing handlers.

HP Operations Manager Integration

The HP Operations Manager for UNIX software provides a fully integrated management solution for networks, systems, databases, and applications found in heterogeneous distributed IT environments. This comprehensive product suite represents a complete set of tools enabling IT organizations to improve overall availability and reliability, maintain the highest degree of management flexibility, and establish management control over virtually all aspects of an enterprise environment.

The SOA Manager software is a distributed management solution that is used to manage enterprise applications that are deployed in SOA-based environments. These applications utilize SOA principals as well as Web services standards. A unique feature of the SOA Manager software is the ability to represent the manageability of assets in a virtual management model called a business service model and expose that management model through Web services-based interfaces.

The SOA Manager and HP Operations Manager integration gives HP Operations Manager the ability to manage the SOA Manager's management information together with other enterprise management data. The integration provides a means of improving the availability of enterprise resources as well as a comprehensive and centralized view of the health and well being of the resources.

The integration includes:

- The ability to view an SOA Manager service model using the HP Operations Manager Service Navigator

- The ability to view service model alert messages in the HP Operations Manager Message Browser
- The ability to acknowledge service model alerts using the HP Operations Manager Message Browser

HP BPI Integration

You can import SOA Manager business services within HP Business Process Insight (BPI), to monitor these services within your business flows. These business services are the representation of a number of related Web services and intermediary (brokered) services within SOA Manager; for example, they might represent an Order Status Query, which can comprise a number of Web services. This Order Status Query might need to be available between certain business hours and return responses within a given time. The availability of this service can be monitored by OVBPI as part of an over-arching business flow within your organization and you can then report on whether or not this service is available, and if the service is not available, what the impact is on your business.

Prerequisites for Integration

Before integration, you must make sure that you perform the following activities:

- Customize startup behaviors
- Automate repetitive tasks
- Save time when updating SOA Manager's assets over multiple installations
- Reuse current tool sets (IDE, Content Versioning System, and so on)
- Maintain current development processes

Startup Tasks for SOA Manager and Policy Enforcement Intermediary

HP SOA Manager and policy enforcement intermediary have the ability to execute user-defined Java code at startup. This is useful for initializing any required services when the process is started, or for executing any other type of process initialization code. To have either SOA Manager or intermediary execute startup code, a class must be created that implements the `com.hp.wsm.sn.server.StartupTask` interface. This startup class should be registered in the `service.xml` file of either HP SOA Manager or HP SOA Manager Intermediary. This file is present in the `/conf` directory.

StartupTask Interface

A startup class must implement the `com.hp.wsm.sn.server.StartupTask` interface in order to be executed. The `StartupTask` interface is as follows:

```
public interface StartupTask {
    public void startup(MipServer server) throws
        StartupTaskFailureException,
        LicenseException.LicenseRunTimeError,
        LicenseException.LicenseNotFound;
}
```


The class must be available on the classpath. When the server executes the startup task, it creates a new instance of the class and executes the startup method, passing in the current `com.hp.wsm.sn.server.MipServer` instance. The `MipServer` class is used for internal processing and is generally not used by user-defined startup tasks. The `MipServer` interface is not documented and is subject to change.

The startup class should throw a subclass of the `com.hp.wsm.sn.server.StartupTaskFailureException` class if a critical problem occurred and the server should shutdown.

Configuration File (server.xml)

The `server.xml` file is located in the `<install_dir>/conf/broker` and `<install_dir>/conf/networkservices` directories, respectively. Underneath the root `<server>` element, it has a `<startup>` element. This is the element in which the list of startup classes is contained. Each startup class should be specified within `<classname>` tags. For example:

```
<server>
  <startup>
    <classname>
      com.hp.wsm.sn.networkservices.NetworkServicesStartupTask
    </classname>
  </startup>
</server>
```

The startup tasks are executed in the order in which they appear in the startup list. User-defined startup tasks should follow the existing `com.hp.wsm.sn` startup tasks. While it is possible to put a startup task before the existing `NetworkServicesStartupTask` or `BrokerStartupTask`, it is not recommended.

Manually Implementing Intermediary Web Services

When using a policy enforcement intermediary-based deployment scenario, the Broker Configurator is typically used to create intermediary Web services. This includes configuring an intermediary Web service's management handlers. As an option, users can manually create intermediary Web services and configure their management handlers. Intermediary Web services are manually created and deployed when:

- You do not want to use the Broker Configurator.
- You want to develop and manage intermediary Web services using your own development environment (IDE, Content Versioning System, development processes).
- You want to create custom intermediary Web services.

The instructions in this section demonstrate how to manually create an intermediary Web service and deploy it to the Policy Enforcement Point (PEP).

Creating an Intermediary Web Service JAR

The artifacts of an intermediary Web service are packaged as a JAR file. A good method for learning about intermediary Web services is to create an intermediary Web service using the Broker Configurator, and then inspecting the intermediary Web service JAR file or using the files in the JAR as a template for creating your own intermediary Web services. When using the Broker Configurator, the JAR file is written to the `<install_dir>/conf/broker`.

The archive contains two files which can be manually created:

- `service.wsdl` – This file contains a service's definition without the address (endpoints) for the service. The service can be defined using either SOAP semantics, or can be created using straight XML.
- `service.xml` – This file contains a service's endpoints and also the management capabilities that will be interposed for the service.

Writing the Intermediary Web Service Definition

The intermediary Web service definition (`service.xml`) is an XML-based file and can be created using a text editor or an XML editor. Among other things, the definition contains a service's endpoints and references to any management handlers. The file can contain handlers that are included with the WSM solution, or can contain any custom handlers you create. The `service.xml` contains the following elements:

<service>

This is the root element of the intermediary Web service definition. The following attributes are included as part of the `<service>` element.

- `class`: Defines the class used to create the intermediary Web service. There are four possible classes that you can use:

- `com.hp.wsm.sn.router.xml.SimpleSoapServiceFactory`: This class is used to create an intermediary service for SOAP-based services as defined in the `service.wsdl` file. The intermediary Web service can contain a predefined set of handlers often referred to as simple handlers.
 - `com.hp.wsm.sn.router.xml.SimpleXmlServiceFactory`: This class is used to create an intermediary Web service for XML-based services as defined in the `service.wsdl` file. The intermediary Web service can contain a predefined set of handlers often referred to as simple handlers.
 - `com.hp.wsm.sn.router.xml.CustomSoapServiceFactory`: This class is used to create an intermediary service for SOAP-based services as defined in the `service.wsdl` file. The intermediary Web service can contain a broad set of handlers and can also contain any custom handlers you create.
 - `com.hp.wsm.sn.router.xml.CustomXmlServiceFactory`: This class is used to create an intermediary Web service for XML-based services as defined in the `service.wsdl` file. The intermediary Web service can contain a broad set of handlers and can also contain any custom handlers you create.
- `name`: The name of the intermediary Web service
 - `version`: A version for the intermediary Web service
 - `namespace`: A list of namespaces entered as *prefix=url*

<transport>

The <transport> element is contained within the <service> element and is used to define the transport layer used to access the intermediary Web service. It contains a reference to the transport provider as well as the http context that is used to access the intermediary Web service and whether the transport layer is secured.

The following example demonstrates the transport definition:

```
<transport
  provider="com.hp.wsm.sn.router.http.HttpTransportProvider">
  <ns1:property name="http.context" value="/financeServiceProxy"/>
  <ns1:property name="http.secure" value="false"/>
</transport>
```

<routing>

The <routing> element is contained within the <service> element and is used to bind the intermediary service to a Web service endpoint. The dispatcher is used to send the request to final endpoint. The following example demonstrates the routing definition:

```
<routing>
  <entry binding="ns2:FinanceServiceSoapSoapBinding">
    <ns3:endpoint
      type="com.hp.wsm.sn.common.dispatcher.
        endpoint.http.HttpEndpointProvider">
      <ns1:property name="priority" value="primary"/>
      <ns4:address>
        http://15.40.235.105:8080/axis/services/FinanceServiceSoap
      </ns4:address>
    </ns3:endpoint>
  </entry>
</routing>
```

Handler Definitions

Each handler that you want to include for the intermediary Web service must be referenced in the intermediary service definition and be contained within the <service> element. Any properties for the handler must be included as attributes. The following example demonstrates the audit handler definition:

```
<audit
  classname="com.hp.wsm.sn.router.xml.handlers.audit.
    AuditHandlerFactory"
  includeProfiling="true"
  payload-filter="ALL"
  payload-option="REQUEST-RESPONSE"
</audit>
```



A reference of all the handlers can be found in the *HP SOA Manager User Guide*.

Creating a Custom Handler

The following example demonstrates a custom handler:

```
Package com.mycompany.CustomHandler

import com.hp.wsm.sn.router.common.message.MessageServiceException;
import com.hp.wsm.sn.router.xml.XmlOperation;
import com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler;
import java.io.IOException;

public class CustomHandler extends BaseXmlHandler {
  public void onRequest(XmlOperation operation) throws
    MessageServiceException {
    try {
      // Custom code goes here.
    }
    catch (IOException e) {
      throw new MessageServiceException.RequestReadException(e);
    }
  }
}
```



An `onResponse` method is also available when processing a response message.

An operation can be retrieved by using `getOperation()` or `getOperationAsString()` on the `XmlOperation` object.

Accessing Attachments for SOAP with Attachments

The policy enforcement intermediary provides an API that can be used in custom handlers to access attachments that are part of a SOAP with Attachments (SwA) message. This API is part of the `com.hp.wsm.sn.router.xml.SoapOperation` class and is therefore only available for SOAP services.

To access the soap attachments of a service request in a custom intermediary Web service handler:

- 1 Create a Handler class which extends `BaseXmlHandler` and overwrites the methods:

```
public void onRequest(XmlOperation operation)
public void onResponse(XmlOperation operation)
```

- 2 In the `onRequest` and/or `onResponse` method, verify that the `XmlOperation` parameter is an instance of `SoapOperation`.

- 3 If it is a `SoapOperation`, the attachments can be accessed by casting the `XmlOperation` to a `SoapOperation` and calling:

```
public Attachments getRequestAttachments()
public Attachments getResponseAttachments()
```

The `com.hp.wsm.sn.router.soap.Attachments` class that is returned from these operations has methods to retrieve body parts by index:

```
public BodyPart get(int i)
```

Or by name:

```
public BodyPart getAttachmentByPartName(String partName)
```

The returned type is `javax.mail.BodyPart`. Below is an example of a handler that accesses attachments from a SwA message:

```
package ...

import com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler;
import com.hp.wsm.sn.router.xml.XmlOperation;
import com.hp.wsm.sn.router.xml.SoapOperation;
import com.hp.wsm.sn.router.common.message.
    MessageServiceException;
import com.hp.wsm.sn.router.soap.Attachments;

public class SoapAttachmentHandler extends BaseXmlHandler {
    public void onRequest(XmlOperation operation)
        throws MessageServiceException {
        // check that the operation is an instance of SoapOperation
        if (operation instanceof SoapOperation) {
            SoapOperation soapOp = (SoapOperation)operation;
            Attachments attachments =
                soapOp.getRequestAttachments();
        // add code to manipulate the request attachment parts.
        }
    }

    public void onResponse(XmlOperation operation)
        throws MessageServiceException.InternalError {
        // check that the operation is an instance of SoapOperation
        if (operation instanceof SoapOperation) {
            SoapOperation soapOp = (SoapOperation)operation;
            Attachments attachments =
                soapOp.getResponseAttachments();
        // add code to manipulate the response attachment parts.
    }
}
```

```

    }
  }
}

```

Adding Custom Handlers to an Intermediary Web Service Definition

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


To add a custom handler:

- 1 Uncompress `<install_dir>\conf\broker\<intermediary_web_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 (`xmlns:ns1="com.company"`).

- 5 Save and close `service.xml`.
- 6 Place the custom handler class and any dependent classes in the same directory as `service.xml`.
- 7 Re-jar the intermediary Web service including the custom handler class and any dependent classes.

Deploying an Intermediary Web Service Jar

Intermediary Web service JARs are placed in the `<install_dir>/conf/broker` directory. Any intermediary Web service JARs that are in this directory are automatically deployed by the policy enforcement intermediary. When the JAR is deployed, it is automatically extracted to `<install_dir>/conf/broker` and placed in a directory that is named using the IP address of the host computer.

 Never edit the extracted files as they will be overwritten when a new JAR is deployed.

Exposing Web Service Entities by Using SOA Manager

SOA Manager creates a unique Managed Object (MO) for every managed entity. A managed entity can be a business service, Policy Enforcement Point (PEP), web service, or a web service configuration that includes multiple instances of web services. SOA Manager exposes these entities as web services. You can use the northbound interfaces to extract the web services during an integration of SOA Manager with other products.

Each MO includes operations to expose the attributes, relationships, performance, events, and so on of a managed entity. Operations with similar functionalities are grouped into port types.

The following port types are available to expose the resources of an MO:

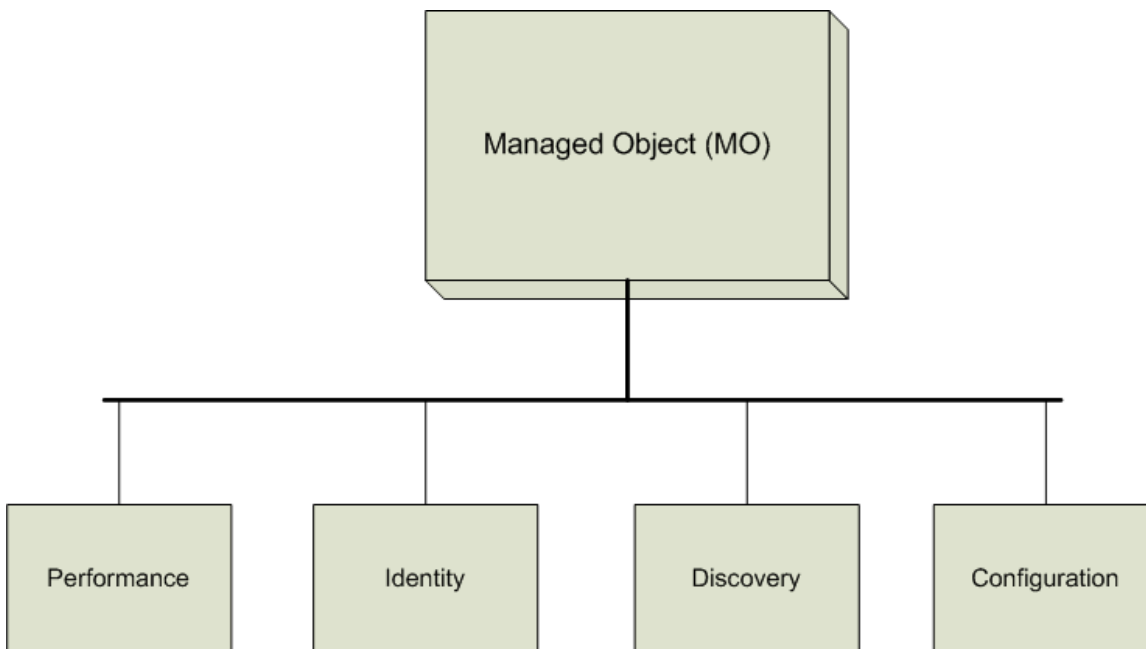
- **Identity:** This port type provides operations such as *GetManagementWsdUrl* that returns the management WSDL URL (identity) of the managed object. You can use this operation to retrieve the identity of any service or entity.
- **Discovery:** This port type provides operations such as *getRelationships* that returns the identity of the related managed objects and the type of relationship.
- **Performance:** This port type provides operations such as *QueryPerformanceData* to expose the performance metrics of a service. A service can be a business service or a Web service.
- **Configuration:** This port type provides operations such as *GetName*, *GetType*, *GetDescription*, and so on to retrieve the attributes of the type of configuration.

You can use the port types listed above and the corresponding operations with MOs that represent the following types of entities:

- SOA Manager
- Catalog services
- Business services
- PEPs

- Business content monitoring notification services

The following diagram represents the types of entities you can expose from a managed object.



In addition to the port types listed above, SOA Manager also provides alert and audit operations for all MOs. You can use these operations for reporting and alert management on the MOs. The following are the port types for alert and audit operations:

- Alert Service and Alert Source: These port types provide operations such as `queryAlerts`, `queryAlertProperties`, `deleteAlertsByIds`, `deleteAlerts`, and `getCurrentAlertSeverity` for alert management.
 - EventPull and EventPush: These port types provide operations to capture all trace information that you want to audit for Web service responses and requests. Trace information includes details about a Web service such as performance data, size, security, start and end points, success, failures, and so on.

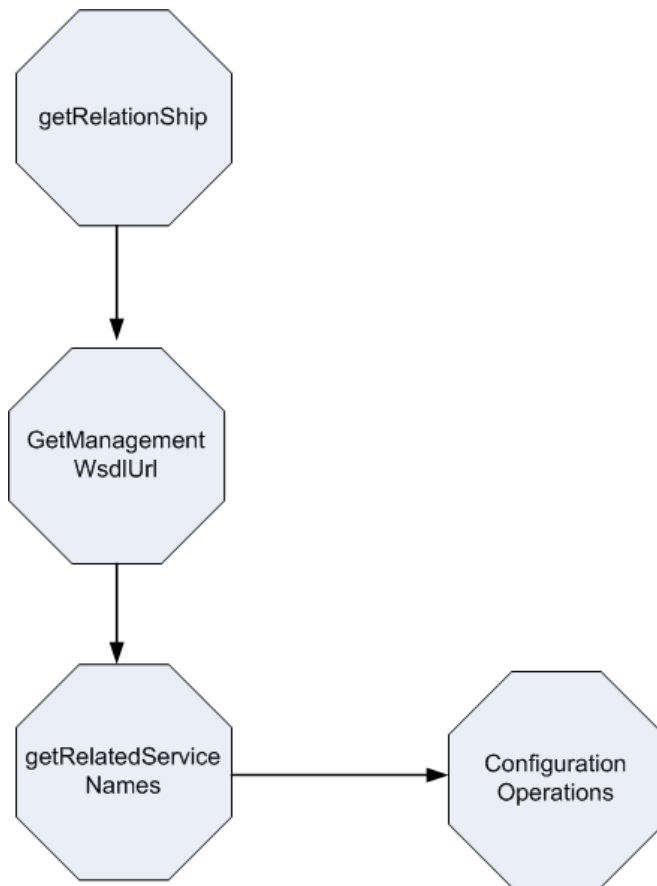
The subsequent chapters contain detailed information about the operations you can use to retrieve the entities of an MO. Following are some use cases that might help you in other product integration scenarios with SOA Manager:

- Use Case 1: Retrieving a list of business services, the attributes, and list of Web services related to the business service.
- Use Case 2: Retrieving performance metrics of a business service or a Web service.
- Use Case 3: Retrieving a list of events from an entity.

Use Case 1

To retrieve a list of business services, the attributes, and a list of web services related to the business service, you must do as follows:

- 1 Use the `getRelationship` operation to retrieve the WSDL of the MO that represents the business services and PEPs from the catalog service.
- 2 Use the `GetManagementWsdUrl` operation to retrieve the WSDL URL of the required business service. This WSDL URL represents the MO SOA Manager creates for each service.
- 3 Use the `getRelatedServiceNames` operation to retrieve a list of Web services related to the business service.
- 4 Use the operations listed in the *Configuration Operations* section to retrieve the attributes of the business service. The following figure represents the sequence of actions that you must perform.



Use Case 2

To retrieve performance metrics of an entity, you can use the *QueryPerformanceData* operation of the *ServicePerformanceSnapshotPT* (performance) port type. This operation returns the performance data from the last poll aggregated for the last six minutes, for every hour, and for a day.

Use Case 3

To retrieve a list of events from an entity, you can use the operations listed under the *EventPull* port type. These operations retrieve all the associated events with the MO from the HP SOA Manager. For example, the *GetSupportedEventTypes* operation returns the URIs for supported event types. For example, a sample output by this operation could be as follows:

The current events are
 Urn:MIPAlert
<http://schemas.hp.com/wsmf/2003/03/Foundation/Event>

Discovery Operations

The *ManagedObjectDiscoveryPT* (discovery) port type provides the *getRelationship* operation to retrieve the relationships and the types of relationships of a managed object with other managed objects. This operation returns the MO WSDL of the managed objects. The following table lists the type of relationships and the URIs that represent them.

Relationship Type	URI
ContainedIn	http://schemas.hp.com/wsmf/2003/03/Relations/ContainedIn
DependedUpon	http://schemas.hp.com/wsmf/2003/03/Relations/DependedUpon
Contains	http://schemas.hp.com/wsmf/2003/03/Relations/Contains
DependsOn	http://schemas.hp.com/wsmf/2003/03/Relations/DependsOn
CorrespondsTo	http://schemas.hp.com/wsmf/2003/03/Relations/CorrespondsTo

Refer to the following table for more information about the types of relationships returned when you use the *getRelationship* operation with different types of services.

Service Type	Returned Relationship Types
Business service	<ul style="list-style-type: none"> • ContainedIn* • Contains**
Web service intermediary	<ul style="list-style-type: none"> • ContainedIn* • Contains**

*Returns the catalog WSDL

**Returns the WSDL URL for the MO

Configuration Operations

The *ManagedObjectConfigurationPT* (configuration) port type provides the following operations to retrieve the attributes of a managed object:

- GetName
- GetType
- GetDescription
- GetOwner
- GetVendor
- GetResourceVersion
- GetManagedObjectVersion
- GetCreatedOn
- GetResourceHostName
- GetManagedObjectHostName

Performance Operations

The *ServicePerformanceSnapshotPT* port type provides the *QueryPerformanceData* operation to retrieve the performance data displayed in the HP SOA Manager web interface for a business service. The operation returns the following data for a business service:

- Availability status
- Uptime in milliseconds
- Aggregated data from the last poll
- Data for the past six minutes
- Aggregated data for one hour
- Aggregated data for a day

Alert Operations

SOA Manager used the *AlertService* and *AlertSource* port types to retrieve and manage the alerts associated with a managed object. These port types provide the following operations.

Operation	Description
deleteAlerts	This operation retrieves alerts by using the <i>queryAlerts</i> and <i>deleteAlertsByIds</i> operations. This operation converts all the retrieved alerts to the resolved state and deletes the alert.

deleteAlertsByIds	This operation changes the ID of an alert to the resolved state and deletes the alert. This operation generates an exception for an invalid alert ID and stops processing the rest of the alert IDs after encountering an invalid alert ID.
queryAlerts	This operation returns the alert details associated with an MO. This operation also returns the alerts based on the severity type specified with the operation. Refer to <i>Alert Severity Types</i> section for more information about alert severity types. This operation is similar to the deleteAlerts and deleteAlertsByIds operations, but it does not delete the alert.
sendAlert	This operation generates an alert by using the Alert Handle and sends the alert to the HP SOA Manager web interface.
queryAlertProperties	This operation retrieves the alert details based on the alert ID and creates a name-value pair and property of the alert and sends it to the HP SOA Manager web interface.
setOVMessageId	This operation adds an <i>AlertMetaData</i> for the alert in the SOA Manager database.
getCurrentAlertSeverity	This operation retrieves the alerts based on the severity type specified. For more information about the alert severity types, refer to the <i>Alerts Severity Types</i> section.

Alert Severity Types

The alert severity types and the URIs that represent the alert severity types are as follows.

Alert Severity Type	URI
Normal	http://schemas.hp.com/notiofication/severity/normal
Information	http://schemas.hp.com/notiofication/severity/info
Warning	http://schemas.hp.com/notiofication/severity/warning
Minor	http://schemas.hp.com/notiofication/severity/minor
Major	http://schemas.hp.com/notiofication/severity/major
Critical	http://schemas.hp.com/notiofication/severity/critical
Unknown	http://schemas.hp.com/notiofication/severity/unknown

You can specify the type of alert to retrieve by using the corresponding URI listed in the table.

Audit Operations

SOA Manager uses the EventPull and EventPush port types to retrieve all events from the HP SOA Manager and to subscribe for the list of event type notifications from the HP SOA Manager. The following sections include information in detail for these port types:

EventPull Port Type

You can use this port type to retrieve all the events from the HP SOA Manager. This port type provides the following operations:

- **GetSupportedEventTypes:** This operation returns the supported list of event types as URIs for the MO. The following table lists the event types and the URIs that represent the event types.

Event Type	URI
RelationshipsChanged	http://schemas.hp.com/wsmf/2003/03/Foundation/Event/RelationshipsChanged
MessageTraceNotification	http://schemas.hp.com/mip/2004/WsExecutionEnvironment/Event/MessageTraceNotification
All Events	http://schemas.hp.com/wsmf/2003/03/Foundation/Event

- **PullSubscribe:** You can use this operation to retrieve all the events from the HP SOA Manager.
- **CancelSubscription:** You can use this operation to cancel the retrieval of all the events from the HP SOA Manager.
- **GetEventsSinceId:** This operation returns the events that occurred since the event ID specified with the operation.
- **GetEventsSinceDate:** This operation returns the events that occurred since the date specified in the operation.
- **GetEventsRangeByDate:** This operations returns the events sorted according to date ranges. The operation returns the events based on the date specified.

EventPush Port Type

You can use this port type to subscribe for events and trace notifications from the HP SOA Manager. This port type provides the following operations:

- **GetSupportedEventTypes**
- **PublishSubscribe**
- **CancelSubscription**

You can use the *PublishSubscribe* operation to subscribe for notifications from the HP SOA Manager. You must provide the URI of the MO and the URI of the event type you require a subscription for, as the input parameters to this operation.

Operations for Managing HP SOA Manager

HP SOA Manager creates an MO that provides the following additional operations to manage HP SOA Manager.

Port Type	Operation	Operation Description
WseeConfigurationPT	GetConfigurationUrl	This operation returns the URL where you can configure the SOA Manager. This operation is currently not working.
ManagedObjectControlPT	SetStatus SetName	These operations allow you to specify the properties of the HP SOA Manager. Refer to the section <i>Status Parameters</i> for a list of URIs you can provide for these operations to specify the status.
ManagedObjectMonitoringPT	GetStatus	This operation retrieves the status of the HP SOA Manager.
WseeControlPT	Stop Start	These operations allow you to stop and start the HP SOA Manager. You cannot start the HP SOA Manager with this operation after you stop it.
WseeLoggingPT	getLogLevel getTrailingLogMessages setLogLevel SetLogLevelForCategory getLogLevelforCategory	These operations allow you to control the logging of the HP SOA Manager.
WseeMonitoringPT	getServices	This operation returns the services running on the Web services container. This operation is currently not working, so you can use the wsmf catalog WSDL to identify the services running on the Web service container.

Status Parameters

The following table lists the status type of the HP SOA Manager and the corresponding URIs that you can provide with the *SetStatus* and *SetName* operations.

Status Type	URI
Failed	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Failed
Inactive	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Inactive

Operational	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Operational
Starting	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Starting
Stopping	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Stopping
Unknown	http://schemas.hp.com/wsmf/2003/03/Foundation/Status/Unknown

Business Service Operations

The following table lists the additional operations that you can use to manage business services.

Port Type	Operation	Description
BusinessServiceUddiDiscovery	getRelatedServiceNames	This operation returns the names of the services related to the MO with the specified relationship type.
LifecyclePT	delete	This operation allows you to delete a business service.
ServiceMonitoringPolicyPT	GetPolicy	This operation returns the policies specified as the entry point for the business service.
	SetPolicy	This operation allows you to specify the policies to be set for the business service at entry point.

PEP Operations

SOA Manager provides the following additional operations that you can use to manage Policy Enforcement Point (PEP) services.

Port Type	Operation	Description
ItServiceResourcePT	getName	This operation returns the name of the PEP.
	unregister	This operation cancels the registration of the PEP service with the HP SOA Manager.
	getObjectNameString	This operation returns the value of the PEP service string as follows: http://soa.hp.com:5002/wsmf/services/Runtime\$service=WsmWseeItService\$IntermediaryWseeITService=Test?wsdl
ItServiceUddiDiscovery	getRelatedServiceNames	This operation returns the resource names of the services related to the PEP service. You can specify a specific type of relationship to retrieve services that have the specified relationship with the PEP service.
	findItResourceName	This operation returns the resource name of the PEP service.

	getSubType	This operation returns the type of the PEP service.
--	------------	---

Publishing the Model

You can publish the service model and the events associated with the model according to the RDF standards. SOA Manager provides the `RDFModelPublisher` port type to publish the service model. This port type provides the following operations:

- `getModel`: This operation exposes the SOA Manager model as an RDF model.
- `getCurrentModelIteration`: This operation returns the iteration number for the model. The iteration number is incremented when the model changes. You can also use this operation to provide the subscription ID of the client that uses the model. This helps in automatically extending the subscription expiry period if the client is found active.

Note: The iteration number is reset to one after the service model is restarted.

- `registerEventListener`: You can use this operation to register for notifications about the events that occur at a client that uses the model. You must provide the subscription ID of the client as the input for this operation.
- `unregisterEventListener`: You can use this operation to cancel the registration for the notifications about the events at a client. You must provide the subscription ID of the client as an input for this operation.

Refer to the *HP SOA Manager User Guide* for more information about publishing the model.

Developing a Policy Enforcement Agent Using Southbound Interfaces

SOA Manager monitors and manages Web services by using either of the following methods:

- Using a policy enforcement agent group: The agent intercepts the service requests at the policy enforcement agent group to provide management capabilities.
- Using a policy enforcement intermediary group: The intermediary intercepts the service requests and provides management capabilities

SOA Manager provides management capabilities for WebLogic 8.1 and .NET (IIS 6.0) agents. If the Web services run on these agents, you may not develop an agent. To monitor Web services running on other types of agents (such as Axis running on tomcat, SAP Netweaver server, IBM Websphere, and so on) using SOA Manager, you must develop an agent. You can contact HP sales or support for any existing agents developed by partners.

A SOA Manager agent discovers the Web services running on the policy enforcement agent group. The agent you design must perform the following minimal functionalities:

- Measure the performance
- Audit the request and response of the service
- Record the details of service invocation

You can also develop the agent to provide other management functionalities listed as follows:

- Deploy or remove deployment of Web services
- Enforce policies for security and business content monitoring

Currently the SOA Manager agent included with SOA Manager provides only the minimal functionalities.

HP SOA Manager communicates to the agent by using the web service technology. SOA Manager agent uses a unique Managed Object (MO) for each managed Web services agent to expose its attributes, relationships, performance, and events.

Managed Objects and Port Types

The agent creates policy enforcement agent MO when an agent is started. The agent considers the policy enforcement agent group as a resource and the services running on the agent group as separate resources.

The MO uses various port types to exposes the attributes, relationships, performance, and events related to a managed Web service. A port type is a collection of Web service operations that have similar functionalities. The default port types used by SOA Manager are classified as follows:

- Identity
- Discovery
- PublishSubscribe
- Event Pull
- Configuration
- Control

All the MOs must implement the port types listed above.

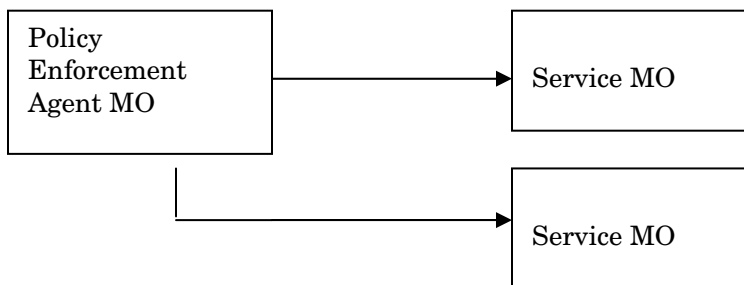
In addition, the policy enforcement agent MO can also implement the following port types:

- Web services container monitoring
- Web services container Control
- Web services container Logging
- Performance
- Deploy

Service Managed Objects and Port Types

SOA Manager agent uses a unique service MO to expose the attributes of the Web services running on a policy enforcement agent group.

The SOA Manager agent manages the life cycle of a service MO when a Web service is deployed, removed, installed, or uninstalled. The following figure illustrates that a policy enforcement agent MO can manage multiple service MOs.



Current HP SOA Manager uses only the Performance and Deploy port types.

In addition to the default port types, service MO also implements the following port types. Refer to section *Managed Objects and Port Types* for information about default port types.

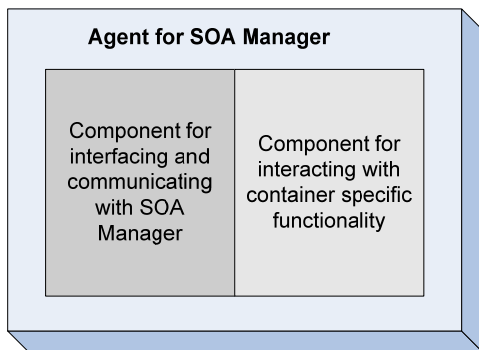
- Service monitoring
- Service control
- Service logging
- Service performance
- Service Performance Ext
- Service Security
- Audit

Managed Object Implementation Details for SOA Manager Agent.

HP SOA Manager manages services using the Web Services (WS) interfaces. You must include at least the following components when developing a SOA Manager policy enforcement agent:

- A component that manages the Web services container and the services
- A component that interfaces with the HP SOA Manager.

The figure below provides a high-level overview of SOA Manager policy enforcement agent.



While designing and developing an agent, you must separate (abstract) the WS interfaces layer from the management functionality layer. This abstraction allows changes to one of these layers without affecting the other layer. This approach helps you to port the agent even when the WS interface to the SOA Manager is updated to evolving standards in future.

The following section covers the different types of interfaces for different managed resources. The implementation details of the management functionalities for a container is specific to a container and is not discussed in this document. The document contains implementation details for the WS interfaces of the SOA Manager agent.

SOA Manager Interfaces for Policy Enforcement Agent Groups

The management interface for the policy enforcement agent group on which the agent is running is defined using the policy enforcement agent Managed Object (referred in the code as Web Services Execution Environment (WSEE)).

SOA Manager uses the following port types to interface with the Web services container:

- Identity port type: This port type uses the `GetManagementWsdUrl` operation to return the management Web Services Description Language (WSDL) URL for the managed object. You can use this port type to register the agent.
- Discovery port type: HP SOA Manager discovers the services running on the container using the `getRelationships` operation. HP SOA Manager invokes this operation periodically and discovers the services running on the policy enforcement agent group to display these services under policy enforcement points in the SOA Manager web interface. This operation returns the management WSDL for the service managed object (MO). A service managed object refers to a managed object created for each service discovered by HP SOA Manager.

NOTE: You must include both the identity and discovery port types when developing an agent.

- Managed Object Configuration port type: HP SOA Manager uses this port type to retrieve the attributes of the Web services container. HP SOA Manager uses operations such as `getName`, `getType`, `getDescription`, and so on to retrieve the attributes of the Web services container. HP SOA Manager used these attributes to display the configuration details of the Web services container object in the HP SOA Manager web interface. HP SOA Manager uses the following attributes of a Web services container:

- Name
- Type
- Resource Version
- Managed Object Version
- Managed Resource Host Name
- Managed Object Host Name
- Description
- Owner
- Vendor

NOTE: You must implement the `GetName`, `GetType`, and `GetDescription` operations to retrieve the attributes of a Web services container.

- Event Push Port type: HP SOA Manager uses an asynchronous mechanism to subscribe for the events from the SOA Manager agent. HP SOA Manager sets a publish subscribe relation between the agent and HP SOA Manager by using the `publishSubscribe` operation. Events from the SOA Manager agent can be one of the following:
- Change in relationship notification event: Changes in relationships between various managed objects such as web services container MOs, Service MOs, and so on.

- **Message trace notification event:** The SOA Manager agent uses message trace notification events to post the request and response messages for the WS operation. SOA Manager agent intercepts a message request and response and sends a copy of this to HP SOA Manager through this notification. Optionally, a SOA Manager agent can also send performance details of an operation to HP SOA Manager using a message trace notification. HP SOA Manager stores an audit message in the database and then updates the performance metrics accordingly.
 - **Performance port type:** HP SOA Manager server uses this port type to poll the agent periodically at intervals of one minute. This Get operation used by this port type exposes all the performance details of the services and operations. You must design this operation in such a way that it returns only the performance metrics collected since the last poll request from the HP SOA Manager. This is a new operation introduced from the SOA Manager version 2.1.

The SOA Manager agent must expose performance data using only one of the above mentioned port types (EventPush or Performance). Exposing performance metrics using more than one port type can result in duplication of data collection at the HP SOA Manager.

SOA Manager Interfaces for Services

HP SOA Manager creates one service MO for each of the services discovered on the Web services container. The following port types expose the service MOs to HP SOA Manager that helps in managing services.

- **Identity port type:** This port type uses the GetManagementWsdUrl operation to return the WSDL for a service.
- **Discovery port type:** This port type uses the getRelationship operation and returns the container on which the services are running.

NOTE: You must include both the identity and discovery port types when developing an agent.

- **Managed Object Configuration port type:** This port type retrieves all the attributes of a service. HP SOA Manager web interface uses the operations such as getName, getType, getDescription, getVersion, getManagedObjectHostName, and so on to retrieve attributes of services and displays them.
- **Service Configuration Port Type:** This port type uses the GetOperationalWsdUrl operation and returns the functional web service WSDL URL. HP SOA Manager uses this URL to discover the operations in the WS.

Operation Details and Schema Used by the Operations

This section provides the operation details and schema details for the mandatory operations for Web services container MO and service MO interfaces. Other operations are optional for the implementation of the SOA Manager agent.

You must implement the mandatory operations and make sure that the operations return correct responses. This is critical for HP SOA Manager to manage services correctly.

ManagedObjectIdentityPT:GetManagementWsdUri()

This operation returns the management WSDL URL for the managed object. HP SOA Manager loads the WSDL from the URL returned by this operation. HP SOA Manager then invokes various operations and manages the resource represented by the managed object.

For each service MO, the management WSDL URL is unique. The SOA Manager agent must have one service MO for every service in the container and each service MO must be uniquely identified by the management WSDL. The recommended approach is to use a template and generate the WSDL dynamically for the service MO.

ManagedObjectDiscoveryPT:GetSupportedRelationships()

This operation returns the list of supported relationship types. These relationship types are represented as Uniform Resource Identifiers (URIs). The table below lists the types of relationships and the corresponding URIs.

Relationship Type	URI
ContainedIn	http://schemas.hp.com/wsmf/2003/03/Relations/ContainedIn
DependedUpon	http://schemas.hp.com/wsmf/2003/03/Relations/DependedUpon
Contains	http://schemas.hp.com/wsmf/2003/03/Relations/Contains
DependsOn	http://schemas.hp.com/wsmf/2003/03/Relations/DependsOn
CorrespondsTo	http://schemas.hp.com/wsmf/2003/03/Relations/CorrespondsTo

ContainedIn and Contains are the only relationship types that HP SOA Manager currently handles. Therefore, you must make sure that agent returns these two relationship types.

ManagedObjectDiscoveryPT:GetRelationships()

This operation returns all the relationships and the types of relationships of a managed object with other managed objects. The values returned for the Web services container MO and service MOs are different.

The returned value contains two URIs as follows:

- Relationship type
- Management WSDL for the managed object

For the Web services container, the relationship type returned is Contains, which lists all the services contained in the Web services container. For the services, the relationship type returned is ContainedIn, which returns the web services container that contains the service.

ManagedObjectDiscoveryPT:GetSpecificRelationships()

This operation uses a relationship type URI as the input parameter and returns the list of the management WSDL URIs that contains the specified relationship.

PublishSubscribePT:GetSupportedEventTypes()

This operation returns the supported list of event types as URIs for the managed object.

The following table lists the values of URIs returned for the Web services container MO:

Event Type	URI
RelationshipsChanged	http://schemas.hp.com/wsmf/2003/03/Foundation/Event/RelationshipsChanged
MessageTraceNotification	http://schemas.hp.com/mip/2004/WsExecutionEnvironment/Event/MessageTraceNotification
BizMetricEvent *	http://schemas.hp.com/mip/2004/Service/Event/BizMetricEvent

* indicates that this event type is specific to service MOs. You can use this event type for business content monitoring.

SOA Manager agent uses message trace notification for uploading the audit message traces (request and response messages) for each operation. Optionally, you can also use message trace notification to send the performance data to HP SOA Manager from SOA Manager agent.

PublishSubscribePT:publishSubscribe()

HP SOA Manager uses this operation to subscribe for the notifications from the SOA Manager agent. HP SOA Manager subscribes for the message trace notifications and relationshipsChanged notification with the Web services container MO. If the service MO supports BizMetricEvent type, HP SOA Manager uses this operation to subscribe for the BizMetricEvent notifications.

The publishSubscribe operation uses the call back URL and expiry time as the input parameters. This operation returns the subscription ID which you can use to cancel the subscription, if required, later. Whenever the subscribed events occur at the container, the SOA Manager agent must send the notification to the callback URL.

You must design the SOA Manager agent in such a way that the agent handles duplicate subscriptions and duplicate notifications of the same event must not be sent to the call back URL. The SOA Manager agent must also track subscriptions and generate a unique ID for each subscription. A subscription must expire at the time specified in the input parameter and no notifications must be sent to HP SOA Manager after the subscription time expires. For a duplicate subscription, you must cancel the existing call back URL and use the new call back URL.

You can also design the SOA Manager agent to store and resend the notifications if the HP SOA Manager is not accessible at the time of the event.

The following section (in grey background) describes how to develop a client to send or publish notifications to HP SOA Manager.

The MessageTraceListType, used for the audit trace notification is defined in the mip-auditig.xsd schema. You can find this schema in <Install Dir>\conf\networkservices\wsmf\mip-auditing.xsd. <Install Dir> refers to the SOA Manager installation directory. In SOA Manager 2.1, optionally, you can use this event to send the performance metrics. Refer to the table below for information about using the schema to send audit traces and performance metrics.

The SOA Manager agent must collect and buffer the messages (request and response) and send these messages as a list using a single push to HP SOA Manager. The SOA Manager agent invokes only the Notify operation, which is bound to the call back URL given during the subscription. The SOA Manager agent must create the MessageTraceListType defined in the MIP auditing schema to send the messages. The following table lists MessageTrace types that you must use in audit trace notifications and performance metrics collection.

Message Trace Type	Description
Sequence ID	A unique string for each message
TraceSourceId	
TraceSourceRole	
ServiceName ^{1,2}	Name of the WS. This name must be the same as the name returned by the getName operation of the Configuration port type for the corresponding service MO.
ServiceVersion ^{1,2}	You must provide the Services functional URL to this message trace type. This URL must be the same as the address where the functional WS is bound.
PortType	Port type of the functional service.
Operation ¹	Operation name of the functional service for which the operation is audited.
Binding	Binding of the functional WS
ServiceType	Service type information
TransportType	Transport type information of the functional service.
ConsumerSecurityPrincipal	Security information from the consumer, if used.
ProviderSecurityPrincipal	Security information from the provider, if used.
SenderURI	Sender URI of the consumer
ReceiverURI	Receiver URI of the consumer
Duration ¹	Time in milliseconds (ms) for WS operation to complete.
Timestamp ¹	Time stamp of the operation invoked.
ErrorType	Error type information for the operation failure.
RequestSize	Size of the request message.
ResponseSize	Size of the response message.
RequestPayload	Request SOAP message
ResponsePayload	Response SOAP message
StoreInDB	Audit trace details to be stored in the

	database. This value can be false if audit trace notification is used only for performance metrics collection.
SecurityViolation	This message trace type is set to true if the failure is due to security violation.
ServiceManagementWsdUrl ^{1,2}	This message trace type contains the value of the management WSDL URL for corresponding services. This value must be the same as the value returned by the GetManagementWSDL URL in the service Configuration port type for the corresponding service MO.
ProfileTraceList	This message trace represents the service profile trace through various components. This is helpful in troubleshooting. For the WS intermediary, this message trace denotes the time spent in various handlers.

Note: Message trace types with “1” next to them indicate mandatory message trace types that you must provide if the audit trace is used for performance metrics collection.

Message trace types with “2” next to them indicate message trace types used to generate unique keys to aggregate the performance metrics in HP SOA Manager. Incorrect values for these message trace types prevent the display of performance metrics in HP SOA Manager web interface.

The SOA Manager agent must send the message trace list to the Notify operation defined in the WS-Events.wsdl (<Install Dir>\conf\networkservices\wsmf) and bound to the callback URL. The input for the operation is NotificationList type. The agent converts the MessageTraceList object to the NotificationType. For more information, refer to the EventProducerHelperUtil.java file which is bundled with sample agent code.

You must make sure that the SOA Manager agent implements a client for the Notify Operation and invokes the operation when the event occurs.

PublishSubscribePT:CancelSubscription()

HP SOA Manager uses this operation to cancel trace notification subscriptions. HP SOA Manager uses the subscription ID parameter returned by the PublishSubscribePT:publishSubscribe() operation to cancel the subscription.

ManagedObjectConfigurationPT

HP SOA Manager uses these operations to find the values of configuration attributes. These operations are mandatory for the implementation of policy enforcement agent MOs and service MOs.

- GetName → returns the name of the policy enforcement agent MO or service MO

- GetType → returns the type of policy enforcement agent MO. For example, for the WebLogic agent, the return value might be <http://wsm.agent.WLS>
- Get Description → returns the description for the policy enforcement agent MO or service MO
- GetResourceVersion -> returns the version of the managed resource
- GetManagedObjectVersion -> returns the version of the managed object
- getResourceHost -> returns the host name of the managed resource
- getManagedObjectHost -> returns the host name of the MO.

ServiceConfigurationPT:GetOperationalWsdUrl (for Service MO)

This operation returns the URI, which is the functional WSDL URL of Web service that is represented by service MO. HP SOA Manager uses this WSDL to discover the operations in the WS. The binding and address to which the service is bound is also used to generate the key to aggregate and store metrics. If the WSDL is invalid or if it does not include the binding information, SOA Manager cannot aggregate and store metrics. Therefore, this information is mandatory.

Performance:Get (Only for Web Service Container)

The SOA Manager agent uses this operation to retrieve performance metrics. The performance metrics list is given in the “Web Service Performance Metrics” section of the *SOA Manager User Guide*.

HP SOA Manager sends a request to the agent to collect all the above mentioned metrics. Performance metrics are collected by the HP SOA Manager for every minute (60000 ms) by default. The interval is configurable in <SOA Manager_install_directory>/networkservices/mipServer.xml using the com.hp.service.polling.interval parameter. <SOA Manager_install_directory> refers to the directory in which you have installed SOA Manager.

The SOA Manager agent must collect and maintain the performance data of WS operations. The performance data must be aggregated and maintained in windows of 30 seconds (30000 ms) for all the operations. Each window must have a unique window index number and this number must be incremented sequentially. The agent must maintain 60 such windows so that it can provide data for the last 30 minutes. The window size and the duration of the data maintained in memory must be configurable. The recommended window interval size is 30 seconds and the recommended duration to maintain the data in memory is 30 minutes.

HP SOA Manager provides window index number in the get operation. When the agent receives a request from the HP SOA Manager with a window index, it returns all the windows that have an index value greater than or equal to the requested index. When the HP SOA Manager starts or when agent is registered, HP SOA Manager requests with an index number 0. For the next performance metric collection request, HP SOA Manager increments the highest window number received in the previous response by 1 and uses this new number as index value.

The following figure shows a model for the performance metrics collection by HP SOA Manager discussed in this section.



An Example:

Let us assume that a Web services container has 2 Web services, WebServ1 and WebServ2. WebServ1 has Oper1 and Oper2 operations and WebServ2 has Oper3 operation.

The SOA Manager agent must collect the time taken for completing the operations Oper1, Oper2 and Oper3 when they are invoked. This can be done in any of the following ways:

- Intercepting through a handler, plug in, or a filter
- Collecting through some of the already available APIs provided by the Web services container

After the performance metrics are collected, the SOA Manager agent must aggregate metrics at operation level for a window at every 30 seconds. The agent must also implement a rolling window of size 60 to store 30 minutes of data at any point of time.

In each window, there must be separate aggregates for each operation, Oper1, Oper2, Oper3, and the performance metrics are stored to easily expose the metrics.

Get Request Schema

The request schema in this section contains a window index in element param0. The response for this request contains all the windows with an index value greater than or equal to the index value specified in the request.

```
<xsd:element name="Get">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="param0" type="xsd:long" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Get Response Schema

The response schema in this section contains windows with an index value greater than or equal to the index value specified in the request. The response also contains metrics for all the Web services managed by the SOA Manager policy enforcement agent. The Web service metrics are displayed based on the operations they perform.

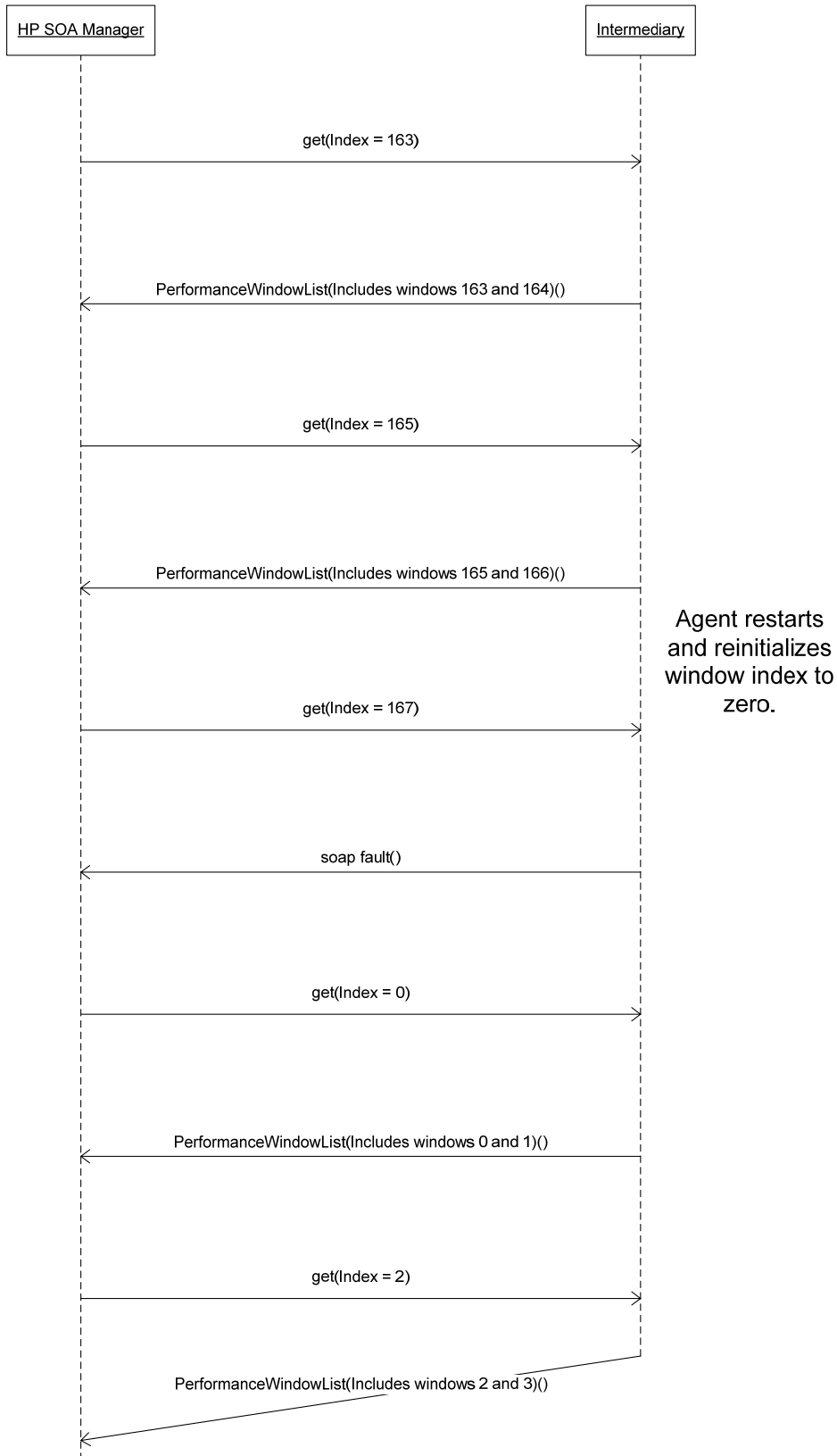
```
<xsd:element name="GetResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element
ref="PerformanceWindow.xsd:PerformanceWindowList"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Response Schema

The response schema can be found in <SOA Manager_install_dir>/conf/policy enforcement intermediary/wsmf/PerformanceWindow.xsd.

Error Handling

The following figure illustrates how SOA Manager agent handles errors when queried by using an incorrect or non-existent index value.



For more information, you can refer to the following sample XML schema for the scenario discussed in this section:

Request :

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:tns="http://openview.hp.com/xmlns/mip/2005/03/Wsee"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:Get>
      <tns:param0>0</tns:param0>
    </tns:Get>
  </soap:Body>
</soap:Envelope>
*****
```

Response :

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:Wsee="http://openview.hp.com/xmlns/mip/2005/03/Wsee"

  xmlns:PerformanceWindow.xsd="http://openview.hp.com/xmlns/soa/1/PerformanceWindow.xsd">
  <soapenv:Body>
    <Wsee:GetResponse>
      <PerformanceWindow.xsd:PerformanceWindowList>
        <PerformanceWindow.xsd:PerformanceWindow>

<PerformanceWindow.xsd:WindowIndex>0</PerformanceWindow.xsd:WindowIndex>
<PerformanceWindow.xsd:StartOfWindow>2006-11-03T17:09:36.160+05:30</PerformanceWindow.xsd:StartOfWindow>
<PerformanceWindow.xsd:EndOfWindow>2006-11-03T17:10:06.096+05:30</PerformanceWindow.xsd:EndOfWindow>
<PerformanceWindow.xsd:ServicePerformanceList>
  <PerformanceWindow.xsd:ServicePerformance>

<PerformanceWindow.xsd:ServiceNamespace>http://wsm.hp.com/Web1</PerformanceWindow.xsd:ServiceNamespace>
<PerformanceWindow.xsd:ServiceLocalName> WEB Service Name - Web1</PerformanceWindow.xsd:ServiceLocalName>
```



```
<PerformanceWindow.xsd:ServiceWsdUrl>Service WSDL URL -
http://machine.domain.com:port/web1?WSDL</PerformanceWindow.xsd:Service
WsdUrl>
<PerformanceWindow.xsd:OperationPerformanceList>
<PerformanceWindow.xsd:OperationPerformance>

<PerformanceWindow.xsd:OperationNamespace>http://wsm.hp.com/Web1</Perfo
rmanceWindow.xsd:OperationNamespace>

<PerformanceWindow.xsd:OperationLocalName>Op1</PerformanceWindow.xsd:Op
erationLocalName>
<PerformanceWindow.xsd:PerformanceMetrics>

<PerformanceWindow.xsd:SuccessMessageCount>0</PerformanceWindow.xsd:Suc
cessMessageCount>
<PerformanceWindow.xsd:SuccessAverageResponseTime>-
1</PerformanceWindow.xsd:SuccessAverageResponseTime>
<PerformanceWindow.xsd:SuccessMinimumResponseTime>-
1</PerformanceWindow.xsd:SuccessMinimumResponseTime>
<PerformanceWindow.xsd:SuccessMaximumResponseTime>-
1</PerformanceWindow.xsd:SuccessMaximumResponseTime>

<PerformanceWindow.xsd:FailureMessageCount>0</PerformanceWindow.xsd:Fai
lureMessageCount>
<PerformanceWindow.xsd:FailureAverageResponseTime>-
1</PerformanceWindow.xsd:FailureAverageResponseTime>
<PerformanceWindow.xsd:FailureMinimumResponseTime>-
1</PerformanceWindow.xsd:FailureMinimumResponseTime>
<PerformanceWindow.xsd:FailureMaximumResponseTime>-
1</PerformanceWindow.xsd:FailureMaximumResponseTime>

<PerformanceWindow.xsd:SecurityViolationMessageCount>0</PerformanceWind
ow.xsd:SecurityViolationMessageCount>
</PerformanceWindow.xsd:PerformanceMetrics>
</PerformanceWindow.xsd:OperationPerformance>
<PerformanceWindow.xsd:OperationPerformance>

<PerformanceWindow.xsd:OperationNamespace>http://wsm.hp.com/Web1</Perfo
rmanceWindow.xsd:OperationNamespace>

<PerformanceWindow.xsd:OperationLocalName>Op2</PerformanceWindow.xsd:Op
erationLocalName>
<PerformanceWindow.xsd:PerformanceMetrics>

<PerformanceWindow.xsd:SuccessMessageCount>0</PerformanceWindow.xsd:Suc
cessMessageCount>
<PerformanceWindow.xsd:SuccessAverageResponseTime>-
1</PerformanceWindow.xsd:SuccessAverageResponseTime>
<PerformanceWindow.xsd:SuccessMinimumResponseTime>-
1</PerformanceWindow.xsd:SuccessMinimumResponseTime>
```

```
<PerformanceWindow.xsd:SuccessMaximumResponseTime>-
1</PerformanceWindow.xsd:SuccessMaximumResponseTime>

<PerformanceWindow.xsd:FailureMessageCount>0</PerformanceWindow.xsd:FailureMessageCount>
<PerformanceWindow.xsd:FailureAverageResponseTime>-
1</PerformanceWindow.xsd:FailureAverageResponseTime>
<PerformanceWindow.xsd:FailureMinimumResponseTime>-
1</PerformanceWindow.xsd:FailureMinimumResponseTime>
<PerformanceWindow.xsd:FailureMaximumResponseTime>-
1</PerformanceWindow.xsd:FailureMaximumResponseTime>

-

<PerformanceWindow.xsd:SecurityViolationMessageCount>0</PerformanceWindow.xsd:SecurityViolationMessageCount>
</PerformanceWindow.xsd:PerformanceMetrics>
</PerformanceWindow.xsd:OperationPerformance>
</PerformanceWindow.xsd:OperationPerformanceList>
</PerformanceWindow.xsd:ServicePerformance>
<PerformanceWindow.xsd:ServicePerformance>

<PerformanceWindow.xsd:ServiceNamespace>http://wsm.hp.com/Web2</PerformanceWindow.xsd:ServiceNamespace>
<PerformanceWindow.xsd:ServiceLocalName> WEB Service Name -
Web2</PerformanceWindow.xsd:ServiceLocalName>
<PerformanceWindow.xsd:ServiceWsdUrl>Service WSDL URL -
http://machine.domain.com:port/web2?WSDL</PerformanceWindow.xsd:ServiceWsdUrl>
<PerformanceWindow.xsd:OperationPerformanceList>
<PerformanceWindow.xsd:OperationPerformance>

<PerformanceWindow.xsd:OperationNamespace>http://wsm.hp.com/Web2</PerformanceWindow.xsd:OperationNamespace>

<PerformanceWindow.xsd:OperationLocalName>Op3</PerformanceWindow.xsd:OperationLocalName>
<PerformanceWindow.xsd:PerformanceMetrics>

<PerformanceWindow.xsd:SuccessMessageCount>0</PerformanceWindow.xsd:SuccessMessageCount>

<PerformanceWindow.xsd:SuccessAverageResponseTime>-
1</PerformanceWindow.xsd:SuccessAverageResponseTime>

<PerformanceWindow.xsd:SuccessMinimumResponseTime>-
1</PerformanceWindow.xsd:SuccessMinimumResponseTime>

<PerformanceWindow.xsd:SuccessMaximumResponseTime>-
1</PerformanceWindow.xsd:SuccessMaximumResponseTime>
```

```
<PerformanceWindow.xsd:FailureMessageCount>0</PerformanceWindow.xsd:FailureMessageCount>

<PerformanceWindow.xsd:FailureAverageResponseTime>-1</PerformanceWindow.xsd:FailureAverageResponseTime>

<PerformanceWindow.xsd:FailureMinimumResponseTime>-1</PerformanceWindow.xsd:FailureMinimumResponseTime>

<PerformanceWindow.xsd:FailureMaximumResponseTime>-1</PerformanceWindow.xsd:FailureMaximumResponseTime>

<PerformanceWindow.xsd:SecurityViolationMessageCount>0</PerformanceWindow.xsd:SecurityViolationMessageCount>
</PerformanceWindow.xsd:PerformanceMetrics>
</PerformanceWindow.xsd:OperationPerformance>
</PerformanceWindow.xsd:OperationPerformanceList>
</PerformanceWindow.xsd:ServicePerformance>

</PerformanceWindow.xsd:ServicePerformanceList>
</PerformanceWindow.xsd:PerformanceWindow>
</PerformanceWindow.xsd:PerformanceWindowList>
  </Wsee:GetResponse>
</soapenv:Body>
```

PerformancePT and PerformanceExtPT: (Only for Service MO)

The SOA Manager agent uses these operations to return the performance data. These operations are deprecated in SOA Manager 2.1.

Steps to Create a New SOA Manager Policy Enforcement Agent

This section lists the steps to create an agent. This section does not describe the details of managing a container using an agent.

Consider the following situation: A web services container has two Web services, namely Web1 and Web 2. Web1 includes two operations, namely, op1 and op2 and Web2 includes an operation, namely, op3.

To create a new SOA Manager agent for this situation, perform the following steps:

- 1 Create a Web services container MO which must be instantiated during the agent startup. The Web services container MO must implement the identity, discovery, and configuration port types.
 - The identity port type on web services container MO returns the management WSDL URL for WSEE MO, for example, `http://<server name >/services/WseeMO?wsdl`

- The Web services container MO must discover all the services in the container and create the service MOs for each of the services. The `getRelationship` operation on discovery port type returns the Management WSDL URL for service MOs. For example:
http://<server name>:<port>/services/ServiceMO?wsdl&name=web1
http:// <server name>:<port>/services/ServiceMO?wsdl&name=web2
- The relationship for both these service MOs is `Contains`. You must provide the corresponding URI for the `Contains` relationship.

The Web services container MO must manage the service MO life cycle whenever services are added or deleted. The `getRelationship` operation must return the updated list of relationships. HP SOA Manager uses this operation to discover the services periodically.

HP SOA Manager web interface uses the Configuration port type operations such as `getName`, `GetResourceHostName`, `GetManagedObjectHostName`, `GetResourceVersion`, `GetManagedObjectVersion`, and so on to display the values of the agent attributes. The web services container MO must implement the corresponding operations in the configuration port type.

- 2 Create a service MO for each service. You must create service MOs at runtime whenever a service is added or removed. In addition, the attributes of a service MO must be discovered and only set at runtime. In the current situation, there are two Service MOs, each with its own attributes.

The identity port operation must return the WSDL URL for the Web1 and Web2 Web services as follows:

http://<server name>:<port>/services/ServiceMO?wsdl&name=web1
http:// <server name>:<port>/services/ServiceMO?wsdl&name=web2

The `getRelationship` operation on the discovery port type must return the `ContainedIn` relationship and provide the Web services container MO WSDL URL: `http://<server name >:<port>/services/WseeMO?wsdl`

HP SOA Manager Web interface displays the values returned by the Managed Object Configuration Port type operations such as `GetName`, `GetResourceHostName` and `GetManagedObjectHostName`.

The service Configuration Port operation `getOperationalWsdUrl` must return the URL of the functional web service WSDL, for example, `http://<server name>:<port>/services/Version?wsdl`

HP SOA Manager uses this functional WSDL URL to discover the operation names, binding information, and the address of the service.

Note: Make sure that the binding and address information in the functional WSDL URL are correct as SOA Manager uses this information to generate the unique key to store and aggregate the metrics.

- 3 Deploy the agent in the container and perform the necessary testing to verify whether the management interfaces are working correctly.
- 4 Register the agent by using the HP SOA Manager web interface as policy enforcement point with the type policy enforcement agent group. You can use the management WSDL URL for the policy enforcement agent MO to register the agent.

- 5 After registering the agent, HP SOA Manager discovers the WS and the operations are deployed accordingly. You can navigate through HP SOA Manager screens and view the services discovered and the operations deployed.
- 6 Implement the EventPushPortType operations such as publishSubscribe, cancelsubscribe, and so on to enable audit tracing. You must also implement a client to send notifications to the callback URL using the Notify operation. Whenever an operation is invoked, the SOA Manager agent must intercept the operation, collect the performance metrics, request or response payload and send the notifications to HP SOA Manager.
- 7 Make sure that the storeDB message trace type is set to TRUE in the message trace so that you can store message traces in the database and view audit traces in HP SOA Manager web interface as reports.
- 8 Implement the performance port type Get operation or overload the audit trace notification by using the publishSubscribe operation to collect performance metrics. This exposes the performance data from the SOA Manager agent. Refer to the corresponding sections for information on using the Get operation and audit trace notification. You must implement only one of these two approaches to expose the performance metrics.

Upgrading SOA Manager Agent from 2.0 to 2.1

You can upgrade a SOA Manager agent from version 2.0 to 2.1 by using one of the following methods:

- Implement the Get operation of the Performance port type
- Overload the audit trace notification by using the publishSubscribe operation in the event push port type

The following sections list the steps in detail for each of the above-mentioned methods.

Upgrading the Agent by Implementing the Get operation

Perform the following steps:

- 1 Implement the performance port type Get operation on the web services container MO.
- 2 Implement the publishSubscribe operation to enable audit tracing and sending notifications. You must use the new message trace schema provided with SOA Manager 2.1 for audit trace and notification implementation.

Upgrading the Agent by Implementing the Audit Trace Notification

To upgrade SOA Manager policy enforcement agent to 2.1 by implementing audit trace notification, you must implement the publishSubscribe operation.

You must make sure that you do not implement the Performance port type Get operation when using this method. You may choose not to remove the Service Performance and Service performanceExt port types if you already have an older version of the SOA Manager policy enforcement agent. This method reduces network bandwidth usage as HP SOA Manager does not send a request to the agent for receiving audit trace notifications.

Database Integration

This chapter describes how to create SLA audit reports using the SOA Manager Server's audit database. The chapter provides a brief overview of the integration architecture and includes a reference of the schema and data dictionary of the audit database. An example of an SLA audit report is also provided.

Overview

SOA Manager has the ability to capture audit information about Web service messages into a central audit database. When a policy enforcement agent group or a policy enforcement intermediary group is registered with HP SOA Manager, the Audit Service registers an event callback listener. Subsequently, the agent or the intermediary posts Audit Trace message lists back to the Audit Service at a configured interval. The Audit Service inserts these trace messages into the Audit database. This information in conjunction with the model relationships which are also populated into the database (such as which Business Service contains which Web services) can be used to enable reporting and analytical applications such as SLA reporting, billing, non-repudiation, forecasting, etc.

This chapter provides instructions for producing custom SLA reports based on the management data that is stored in the Audit database.

Database Schema Reference

There are two tables that are used to create SLA Audit reports:

- MESSAGE_TRACE
- MESSAGE

MESSAGE_TRACE Table

Name	Type	Description
------	------	-------------

Name	Type	Description
sequenceId	xsd:string	<p>An identifier that links together MessageTrace instances that represent hops between nodes (i.e. intermediaries and agents) for the same message call. This information is currently passed between nodes in the HTTP header SequenceHeader.</p> <p>If a node receives a message that has a SequenceHeader, it should use that value to populate the sequenceId field of the MessageTrace for that message.</p> <p>If a node receives a message that does not have a SequenceHeader, the node should generate a new globally unique sequenceId. It should use this value to populate the sequenceId field of the MessageTrace for that message.</p> <p>In either case, the node should also add a SequenceHeader HTTP header to any outgoing messages associated with this call and populate it with the appropriate sequenceId.</p>
traceSourceId	xsd:string	<p>A unique identifier that identifies the source of the message trace – the intermediary or agent that is sending the message trace to the CollectionService.</p> <p>The value is the host and port of the router/agent sending the trace.</p>
traceSourceRole	xsd:string	<p>A string indicating the role being played by the sender of this message trace with regards to the message being traced. Restricted to sender (if the agent is sending trace information regarding a message being sent to another service) or receiver (if the agent is sending trace information regarding a message that it has received).</p>
serviceName	xsd:string	<p>The name of the service being invoked.</p> <p>The value of this field is the name of the service element in the WSDL for this service.</p>
serviceVersion	xsd:string	<p>The version of the service being invoked.</p> <p>The value of this field is the namespace of the service element in the WSDL for this service.</p>
portType	xsd:string	The WSDL portType being invoked.
operation	xsd:string	The WSDL operation being invoked.
binding	xsd:string	The WSDL binding being invoked.

Name	Type	Description
serviceType	xsd:string	An indicator if the service is managed internally (proxy, agent) or is outside the service network. This field is restricted to the values <code>internal</code> and <code>external</code> .
transportType	xsd:string	The transport on which the message traveled. This is typically <code>http</code> or <code>jms</code> .
consumerSecurityPrincipal	xsd:string	The Principal invoking the service.
providerSecurityPrincipal	xsd:string	The Principal used by the services network on behalf of the consumer.
senderURI	xsd:string	A URI identifying the message source.
receiverURI	xsd:string	A URI identifying the message destination.
duration	xsd:long	The duration in milliseconds of the call from the time the request was received by this node to when it was completed.
timestamp	xsd:dateTime	The time at which the call was received. This should be in UTC format ('Z' extension) with Millisecond (.sss) precision.
errorType	xsd:string	An indicator for the type of error that occurred, if any. This field is restricted to the values <code>none</code> , <code>application</code> , <code>transport</code> , <code>timeout</code> , <code>marshalling</code> , and <code>soap fault</code> .
requestSize	xsd:int	The size of the request in bytes.
responseSize	xsd:int	The size of the response in bytes

MESSAGE Table

This table contains actual message payloads.

Name	Type	Description
sequenceId	xsd:string	The UUID representing this message. This ID gets generated at each agent and populated into Audit trace record and is used to co-relate audit traces to other kinds of alerts.
requestMessage	CLOB	The payload body of a request message.
responseMessage	CLOB	The payload body of a response message.

Sample SLA Reports

You can build SLA reports from the SOA Manager Audit database using a reporting package like Crystal Reports. Refer to the Audit database schema and data dictionary to understand the source of this information.

From: MM/DD/YY HH:MM:SS

To: MM/DD/YY HH:MM:SS

Compute over interval: MONTH / WEEK / DAY / HOUR

Consumer: <securityPrincipal>

Consumed Service: <serviceNS>:<serviceName>

Number of Requests made:

Number of Requests processed successfully:

Number of Requests failed:

%age Availability over interval: (Number of Requests processed successfully / Number of Requests made) X 100

Max Response time:

Average Response time:

Sample SLA Reports for Month of Nov 2004

From: 11/01/04 00:00:00

To: 11/30/04 00:00:00

Compute over : DAY

Consumer: joebob

Consumed Service: http://wsm.hp.com/finance:financeServiceProxy

Time Window	Number of Requests	Number of Successes	Number of failures	% Availability	Ave Response (ms)	Max Response (ms)
11/01/04 – 11/02/04	35	24	11	68.57	274	360
11/02/04 – 11/03/04	45	44	1	97.77	248	423
...						

Consumed Service: http://wsm.hp.com/mobile:mobileServiceProxy

Time Window	Number of Requests	Number of Successes	Number of failures	% Availability	Ave Response (ms)	Max Response (ms)
11/01/04 – 11/02/04	29	29	0	100	128	175
11/02/04- 11/03/04	13	13	0	100	110	155
...						

Select Access Integration

Refer to chapter *Integrating with Select Access* in the *HP SOA Manager User Guide* for detailed instructions about integrating Select Access with SOA Manager.

Configuring Instructions

Refer to the following sections to configure HP SOA Manager integration with HP Select Access.

Security Customizations

The SOA Manager's Java API integration point is used to create custom security implementations that allow an organization to enforce security policies that are not covered by HP SOA Manager policy enforcement intermediary's default security features. This includes custom authentication based on user profile, custom security handlers, and an XML Introspection service.

This chapter provides instructions for customizing the security features that are provided by the policy enforcement intermediary or agent groups.

Creating an XML Introspection Service

An XML Introspection service provides the ability to authorize services not only by user credentials but also by the request XML content, such as request XML message body or any XML documents. An XML Introspect service:

- Authenticates an XML service with HTTP Basic Authentication Only by using the Select Access Validator;
- Adds an XML message to service authorization context by a custom XML Introspection handler;
- Authorizes the service by sending the authenticated credential (which will be a SecureToken if returned by the Select Access Validator in the first authentication or the username and password credential) and the XML message in the authorization context to Select Access Validator.

Create an XML Introspection Handler Class

XML introspection is completed using a custom handler. Like all custom handlers, the handler must extend `BaseXmlHandler`. The handler is responsible for retrieving the XML document and applying security methods. The following example demonstrates an Introspection handler:

```
package com.hp.wsm.sn.common.security.auth.selectaccess;
import com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler;
import com.hp.wsm.sn.router.xml.XmlOperation;
import com.hp.wsm.sn.router.common.message.MessageServiceException;
import com.hp.wsm.sn.common.security.common.SecurityContext;
import com.hp.wsm.sn.common.security.common.AuthorizationContext;
import org.w3c.dom.Document;
import java.util.Map;

public class XmlIntrospectionHandler extends BaseXmlHandler {
    /**
     * Handle service request.
     * @param operation
     * @throws MessageServiceException
     */
    public void onRequest(XmlOperation operation) throws
        MessageServiceException {

        super.onRequest(operation);
        // find the request XML Document
        final Document requestDoc = operation.getRequestDocument();
        final SecurityContext securityContext =
            operation.getSecurityContext();
        final AuthorizationContext authorizationContext =
            securityContext.getAuthorizationContext();
        Map params = authorizationContext.getServiceParameters();

        // put an XML document into authorization parameter list with
        // name "XML" which will be picked up by SelectAccess Security
        // Provider when service is being authorized against
        // SelectAccess validator.

        params.put("XML", requestDoc);
    }
}
```

Create an XML Introspection Intermediary Web Service

The steps in this section create an intermediary Web service and add the XML Introspection handler to the intermediary Web service. If you are not familiar with custom intermediary service and adding custom handlers to an intermediary service, see the *HP SOA Manager User Guide*.

To create an XML introspection intermediary web service:

- 1 Open the Broker Configurator console, and create an XML Service. For example: `XmlIntrospectionService`.
- 2 When defining the service, select **Basic Authentication** and select the check box for **Authentication Only** in the Inbound Transport Authorization section.



Ensure that you have completed the steps for integrating Select Access and setting up basic authentication only for user authentication with the Intermediary.

- 3 Select any features as needed and click **finish**.
- 4 Convert the XML service to a Custom XML service and from the Edit Custom Service screen, select **Service Security Inbound Handler** from the Add a new handler drop-down list.
- 5 Click the **OK** to save the intermediary service.
- 6 Find and extract the service JAR file under `<install_dir>/conf/broker` (`XmlIntrospectionService.jar`).
- 7 Edit the `service.xml` file and add the XML Introspection handler to the handler list before `ServiceSecurityInboundHandler`. For example:


```
<handler classname="com.hp.wsm.sn.router.xml.handlers.
LogHandler" />

<handler classname="com.hp.wsm.sn.router.xml.handlers.inbound.
XmlContractHandler" />

<handler classname="com.hp.wsm.sn.router.xml.handlers.outbound.
XmlDispatchHandler" />

<handler classname= "com.hp.wsm.sn.common.security.auth.selectaccess.
XmlIntrospectionHandler" />

<handler classname="com.hp.wsm.sn.router.xml.handlers.inbound.
ServiceSecurityInboundHandler" />
```
- 8 Re-package the service JAR including the XML Introspection handler. The handler must have the same fully qualified name as entered in the `service.xml` file.
- 9 Undeploy and deploy the intermediary service for the changes to take effect.

Role-based Security Access (Implementing Custom Authentication based on User Profile)

It is possible to customize the WSM Intermediary Agent's security features including the integration with Select Access. In general, there are two use cases that are covered:

- Passing custom query parameters as part of a Select Access authorization query
- Checking personalization attributes that are returned as a result of an authentication query

Both use cases can be implemented either at the message level or the transport level. Typically, the customization occurs within the handler chain by either creating custom handlers or customizing existing handlers.

Customizing the RealmAuthorizationHandler

The `RealmAuthorizationHandler` interface offers callback methods that are used to add custom authorization policies when using HTTP transport level security. See the *HP SOA Manager User Guide* for instructions on setting up transport-level security..

There are four steps discussed in this section:

- Create a class that implements the `RealmAuthorizationHandler` interface.
- Add your implementation class to the `<install_dir>/lib/ext` directory.
- Configure the implementation class in the Intermediary's `MipServer.xml` configuration file.
- Verify the integration.

Create an Implementation Class

Create a class that implements the `RealmAuthorizationHandler` interface. Details about the interface are provided below, as well as an example.

Interface: RealmAuthorizationHandler

```
com.hp.wsm.sn.common.security.auth.jetty.RealmAuthorizationHandler
```

Methods

Method	Description
<code>getContextParameters</code>	<p>Context parameters are passed to security provider as context to an authorization request. This method returns string name value pairs representing context parameters.</p> <pre>public java.util.Properties getContextParameters (com.hp.wsm.sn.common.security.auth.ServiceRequest serviceRequest, org.mortbay.http.HttpServletRequest httpRequest)</pre> <ul style="list-style-type: none"> • <code>serviceRequest</code> – only the service name will be known at the transport level. • <code>httpRequest</code> – the current http request.

Method	Description
onAuthorize	<p>The security provider has successfully authorized the subject. Implement any custom authorization rules here, such as checking subject attributes. This method returns <code>true</code> if the subject is authorized to access the service.</p> <pre>public boolean onAuthorize (javax.security.auth.Subject subject, com.hp.wsm.sn.common.security.auth.ServiceRequest serviceRequest, org.mortbay.http.HttpServletRequest httpRequest)</pre> <ul style="list-style-type: none"> • <code>subject</code> - the authorized subject. • <code>serviceRequest</code> - only the service name will be known at the transport level. • <code>httpRequest</code> - the current http request.

Example

```
package com.hp.wsm.examples.auth;
import com.hp.wsm.sn.common.security.auth.jetty.
    RealmAuthorizationHandler;
import com.hp.wsm.sn.common.security.auth.ServiceRequest;
import com.hp.wsm.sn.common.security.auth.
    SubjectAttributesCredential;
import java.util.Properties;
import java.util.Set;
import org.mortbay.http.HttpServletRequest;
import javax.security.auth.Subject;

public class ExampleRealmAuthenticationHandler implements
    RealmAuthorizationHandler {
    public Properties getContextParameters(ServiceRequest serviceReq,
        HttpServletRequest request) {
        Properties context = new Properties();
        context.setProperty("contentLength",
            String.valueOf(request.getContentLength()));
        return context;
    }

    public boolean onAuthorize(Subject subject, ServiceRequest
        serviceReq, HttpServletRequest request) {
        Set attributeCredentials = subject.getPublicCredentials
            (SubjectAttributesCredential.class);
        if (!attributeCredentials.isEmpty()) {
            SubjectAttributesCredential attributes =
                (SubjectAttributesCredential)
                    attributeCredentials.iterator().next();
            String roles = attributes.getSubjectAttributes()
                .getProperty("groups");
            if (roles != null) {
                return roles.indexOf("gold") > -1;
            }
        }
        return false;
    }
}
```

Add the Implementation Class

The implementation class must be located by the Security Provider at runtime. Typically, this means creating a JAR that contains the implementation class and adding it to the `<install_dir>/lib/ext` directory. This ensures that the class is located in the classpath of the Intermediary Agent.

Configuring the Policy Enforcement Intermediary

The policy enforcement intermediary must be configured to use the implementation class at runtime. The configuration entry is added in the policy enforcement intermediary's `mipServer.xml` file.

To configure the implementation, you must do as follows:

- 1 Use a text editor to open `<install_dir>/conf/broker/mipServer.xml`.
- 2 Add the property "com.hp.mip.security.realmAuthHandler" and set it to the fully qualified implementation class name. For instance, using the above example, the following entry would be used:

```
<entry name=com.hp.mip.security.realmAuthHandler>
  com.hp.wsm.examples.auth.ExampleAuthorizationHandler
</entry>
```

- 3 Save and close the File.

Verify the Integration

To verify the integration, you must create a custom intermediary Web service that is configured to use transport-level security.

Creating Custom Security Handlers

When using message-level security, custom handlers can be created to pass custom query parameters and check personalization attributes. The handlers are included as part of a custom XML intermediary service's handler chain.

In this scenario two handlers are created. A context handler is used to setup context information before an authorization call (i.e., to Select Access) and an authorization handler is used to check any returned attributes. An example of these handlers is provided below. They are named `ExampleContextHandler` and `ExampleAuthorizationHandler` respectively.

There are two steps discussed in this section:

- Create the security handlers.
- Add the handlers to a custom intermediary service definition

Create the Handlers

Create two handlers that extend `com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler`. The handlers use the method `AuthorizationContext` available from the `XmlOperation` method that is passed into the `onRequest` method for an XML Handler. An example of both handlers is provided below:

ExampleContextHandler

```
package com.hp.mip.test.auth;

import com.hp.wsm.sn.router.common.message.MessageServiceException;
import com.hp.wsm.sn.router.xml.XmlOperation;
import com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler;

import java.io.IOException;

public class ExampleContextHandler extends BaseXmlHandler {

    public void onRequest(XmlOperation operation) throws
        MessageServiceException {
        super.onRequest(operation);
        try {
            int length = operation.getAuditRequestData().getBytes().length;
            operation.getSecurityContext().getAuthorizationContext()
                .getContextProperties().setProperty("contentLength",
                    String.valueOf(length));
        }
        catch (IOException e) {
            throw new MessageServiceException.RequestReadException(e);
        }
    }
}
```

ExampleAuthorizationHandler

```

package com.hp.mip.test.auth;

import
com.hp.wsm.sn.common.security.auth.SubjectAttributesCredential;

import com.hp.wsm.sn.router.common.message.MessageServiceException;
import com.hp.wsm.sn.router.xml.XmlOperation;
import com.hp.wsm.sn.router.xml.handlers.BaseXmlHandler;

import javax.security.auth.Subject;
import java.util.Set;

public class ExampleAuthorizationHandler extends BaseXmlHandler {

    public void onRequest(XmlOperation operation) throws
        MessageServiceException {
        Subject subject = operation.getSecurityContext().getSubject();
        Set attributeCredentials =
subject.getPublicCredentials(SubjectAttributesCredential.class);
        String email = null;
        if (!attributeCredentials.isEmpty()) {
            SubjectAttributesCredential attributes =
                (SubjectAttributesCredential)
                    attributeCredentials.iterator().next();
            email =
                attributes.getSubjectAttributes().getProperty("email");
        }
        if (email == null || email.indexOf("hp.com") == -1) {
            throw new
                MessageServiceException.BadRequest.NotAuthorized("Subject is
                not from hp!", null);
        }
    }
}

```

Add the Handlers to a Custom Intermediary Web Service

The custom security handlers must be added to a custom intermediary Web service that is configured to use message-level security. In particular, the custom intermediary Web service must contain the handler `ServiceSecurityInboundHandler`.

When adding the handlers to the handler chain, the context handler (`ExampleContextHandler`) must be located before `ServiceSecurityInboundHandler` and the authorization handler (`ExampleAuthorizationHandler`) must be located after `ServiceSecurityInboundHandler` in the handler chain.

HP Operations Manager Integration

Overview

The HP Operations Manager software provides a fully integrated management solution for networks, systems, databases, and applications found in heterogeneous distributed IT environments. This comprehensive product suite represents a complete set of tools enabling IT organizations to improve overall availability and reliability, maintain the highest degree of management flexibility, and establish management control over virtually all aspects of an enterprise environment.

The SOA Manager software is a distributed management solution that is used to manage enterprise applications that are deployed in SOA-based environments. These applications utilize SOA principals as well as Web services standards. A unique feature of the SOA Manager software is the ability to represent the manageability of assets in a virtual management model called a business service model and expose that management model through Web services-based interfaces.

The SOA Manager Operations Manager integration gives Operations Manager the ability to manage the SOA Manager's management information together with other enterprise management data. The integration provides a means of improving the availability of enterprise resources as well as a comprehensive and centralized view of the health and well being of the resources.

The integration includes:

- The ability to view an SOA Manager service model using the HP Operations Manager Service Navigator
- The ability to view service model alert messages in the HP Operations Manager Message Browser
- The ability to acknowledge service model alerts using the HP Operations Manager Message Browser

Conceptual Architecture

The SOA Manager Operations Manager Plug-in is the main component of the integration. The Plug-in is installed on the HP Operations Manager Management Server and communicates with the SOA Manager's SOA Manager Server using a standard Web services based management protocol.

The Plug-in contains a Web service client that is used to get management information from the SOA Manager Server. This information includes the service model (business services, policy enforcement points, and so on...) and also includes any service model alerts.

In particular, two Web services are published by the SOA Manager Server and used by the plug-in:

- `RDFModelPublisher` – This Web service is used to get the SOA Manager's service model that is published as an RDF-based (Resource Description Framework) model.
- `AlertService` – This Web service is used to get and acknowledge alert messages.

The Plug-in then transforms the management information into an Operations Manager format so that it can be viewed using the HP Operations Manager Console. The Plug-in utilizes several HP Operations Manager tools to present the management information. These tools include:

- `opcservice` – This tool is used to update the Service Navigator's service map.
- `opcmsg` – This tool is used to update the Message Browser to include all alert messages.
- `opcmack` – This tool is used to synchronize messages that have been acknowledged in either the SOA Manager's web interface or the HP Operations Manager Message Browser.

Installation and Configuration

This chapter provides instructions for installing and configuring the SOA Manager Operations Manager Plug-in. If you are not familiar with HP Operations Manager, you should consult the HP Operations Manager documentation when completing the instructions in this chapter. In addition, it is recommended that a development-time installation of HP Operations Manager be used to test the SOA Manager Operations Manager Integration before installing and running on a production installation of HP Operations Manager.

Requirements

This section details the hardware and software requirements for installing and running the SOA Manager Operations Manager Plug-in. Make sure the requirements are met before beginning the installation.



Before installing the SOA Manager Operations Manager Plug-in, make sure SOA Manager is installed and operational.

Software Requirements

The following table provides a list of software that is required to install and run the SOA Manager Operations Manager Plug-in.

Table 8-1: SOA Manager Operations Manager Integration Required Software

Products	Platforms
SOA Manager 2.50	Windows: Microsoft Windows 2003 Server Unix: HP-UX PA-RISC 11.11i, HP-UX Itanium 11.23 and Red Hat Linux Advanced Server 3.0
Operations Manager 8.0 or above	HP-UX 11.23, Solaris 2.9/5.9/9.0, Solaris 2.8/5.8/8.0
JavaConsole A.08.00 (UNIX version included with HP Operations Manager)	HP-UX 11.x, Windows 2K / NT
JDK 1.4.x	HP-UX 11.23, Solaris 2.9/5.9/9.0, Solaris 2.8/5.8/8.0

Hardware Requirements

The following minimum hardware is required to install and run the SOA Manager Operations Manager Plug-in.

Table 8-2: Hardware Requirements

Requirement	Minimum
RAM	512 MB
Disk Space	50 MB

Patches

It is important that all recommended patches for Operations Manager and Java are installed prior to running the SOA Manager Operations Manager Plug-in. To find the recommended HP Operations Manager patches, go to <http://support.openview.hp.com/patches>. In the browse by product version section, select **operations for UNIX** and click the >> button. To find the recommended Java patches for HP-UX, go to <http://www.hp.com/products1/unix/java>. From the left side of the screen, click **Patches**. To find the recommended Java patches for Solaris, go to <http://sunsolve.sun.com/pubpatch> for the latest J2SE patch cluster.



Check for updated patches every few months.

Kernel Parameters

Verify that the kernel parameters meet the recommended settings to run a Java program. You can use the HPjconfig tool to determine the recommended kernel parameter settings. To download the HPjconfig tool, go to <http://www.hp.com/products1/unix/java/java2/hpjconfig>.

Installing the Plug-in

This section includes steps for installing the SOA Manager Operations Manager Plug-in on HP-UX and Solaris. The installation is performed on an Operations Manager management server. In addition, you must be logged onto the management server as root when completing the instructions in this section.

To install the SOA Manager Operations Manager plug-in:

- 1 From the SOA Manager distribution, copy `Addons/ovo/soam-ovou-hpux.depot`, `Addons/ovo/soam-ovou-hpux-itanium.depot` or `Addons/ovo/soam-ovou-solaris.depot` to a location on the management server.

- 2 From a command prompt, run:

```
swinstall -x reinstall=true -s <depot_file> \*
```

The depot is installed to `/opt/soam-ovou`. This location is referred to as `plugin_install_dir` throughout this documentation.

Setting Environment Variables

The Plug-in requires the following environment variables to be set on the HP Operations Manager management server:

- `JAVA_HOME`: Set this variable to the JDK 1.4.x installation directory.
- For HP-UX PA-RISC, set `SHLIB_PATH` to `/opt/OV/lib` or wherever `libopcdb.sl` and `libopcsv_r.sl` are located.
- For HP-UX Itanium, set `SHLIB_PATH` to `/opt/OV/lib/hpux32` or wherever `libopcdb.so` and `libopcsv_r.so` are located.
- For Solaris, set `LD_LIBRARY_PATH` to `/opt/OV/lib` or wherever `libopcdb.so` and `libopcsv_r.so` are located.

Configuring the Plug-in

This section includes instructions for configuring the Plug-in. The Plug-in is configured using the `<plugin_install_dir>/soam-ovou.properties` file. The properties file is a text file and can be edited using a text editor.

The properties file contains default values which may or may not be valid for your HP Operations Manager and SOA Manager installation. The default values must be changed before starting the Plug-in.

To configure the Plug-in:

- 1 Using a text editor, open `<plugin_install_dir>/soam-ovou.properties`.
- 2 Configure the following properties:

- **NS_BASE_URL:** Enter the URL of the SOA Manager Server. The Plug-in uses this URL to connect to the SOA Manager Server. For example:

```
http://<host>:5002
```

Replace *<host>* with either the IP address or the fully qualified DNS name of the computer where the SOA Manager Server is installed. Port 5002 is the default port used to connect to the SOA Manager Server. Change the port number if a different port is being used.
- **HP Operations Manager_ACK_OPERATOR:** Enter a valid HP Operations Manager operator username. This user name is used to connect to the HP Operations Manager server in order to synchronize acknowledged messages between the SOA Manager and HP Operations Manager Message Browser.
- **HP Operations Manager_ACK_PASSWORD:** Run the `<plugin_install_dir>/encrypt-pwd.sh` script to encrypt the HP Operations Manager operator password. The script prompts for the password and returns an encrypted password. Enter the returned encrypted password as the value of this property.
- **SERVICE_MAP_OPERATORS:** Enter a comma separated list of valid HP Operations Manager operator user names that are to be granted access to the service map.
- **OPCSERVICE:** Enter the path to the HP Operations Manager `opcservice` command line tool. The default path where this tool is located is typically `/opt/OV/bin/`. This tool is required to run the Plug-in.
- **OPCMACK:** Enter the path to the HP Operations Manager `opcmack` command line tool. The default path where this tool is located is typically `/opt/OV/bin/`. This tool is required to run the Plug-in.
- **OPCMSG:** Enter the path to the HP Operations Manager `opcmsg` command line tool. The default path where this tool is located is typically `/opt/OV/bin/`. This tool is required to run the Plug-in.
- **LISTENER_PORT:** Enter a port that is currently not being used. The port is used to listen for model change events.

3 Save and close `soam-ovou.properties`.

Creating a SOAM Message Group

The plug-in is configured to group all messages as part of the SOAM message group. This message group must be created. If the SOAM message group is not created and set to a valid operator name, the messages are sent to the Misc (miscellaneous) message group.

The following tasks are used to create a message group:

- Create a Message Group
- Modify the Operator User Profile

Create a Message Group

To create a custom message group:

- 1 From any HP Operations Manager window, open the Message Group Bank window by selecting **Window | Message Group Bank**. The HP Operations Manager Message Group Bank window displays.
- 2 From the HP Operations Manager Message Group Bank window, select **Actions | Message Groups | Add...** The Add Message Group dialog box displays.
- 3 From the Add Message Group dialog box, enter *SOAM* as the name and enter a description.
- 4 Click **OK**. The HP Operations Manager Message Group Bank window displays and the new message group is listed.
- 5 Close the Message Group Bank window.

Modify the Operator User Profile

A message group must be assigned to a valid operator profile. In these instructions, the *opc_adm* operator is modified to include the *SOAM* message group. If you are using a different operator profile, use that operator instead.

To modify the *opc_adm* user profile:

- 1 From any HP Operations Manager window, open the User Bank window by selecting **Window | User Bank**. The HP Operations Manager User Bank window displays.
- 2 From the HP Operations Manager User Bank window, right-click **opc_adm** and select **Modify**. The Modify User: *opc_adm* dialog box displays.
- 3 From the dialog box, select **Responsibilities**. The Responsibilities for Operator [*opc_adm*] dialog box displays.
- 4 From the Message Groups column, scroll to locate the *SOAM* message group.
- 5 Click the Node Group check boxes to assign the *SOAM* message group to different node groups.
- 6 Click **Close**.
- 7 From the Modify User: *opc_adm* dialog box, click **OK**.
- 8 Close the HP Operations Manager User Profile Bank window.

Starting the Plug-in

The Plug-in is started using a shell script. The script is located in the *<plugin_install_dir>*. Before starting the Plug-in, make sure that both the SOA Manager Server and the HP Operations Manager Management Server are started.

To start the plug-in:

- 1 From a command prompt, change directories to *<plugin_install_dir>*.

- 2 Run `start-soam-ovou.sh`. The Plug-in may take several minutes to start.

Stopping the Plug-in

The Plug-in is stopped using a shell script. The script is located in the `<plugin_install_dir>`.

To stop the plug-in:

- 1 From a command prompt, change directories to `<plugin_install_dir>`.
- 2 Run `stop-soam-ovou.sh`. The shutdown is complete when the following message displays in the output:

```
SOA Manager HP Operations Manager plugin stopped
```

Installing the Java Console

HP Operations Manager for UNIX includes a Java-based operator's console that is used to monitor an HP Operations Manager managed environment. In regards to the SOA Manager Operations Manager integration, the Java console is used to:

- View the status of the SOAM services.
- Perform impact analysis and root cause analysis for SOAM services.
- View and acknowledge SOAM messages.

The Java Console can run on any system where a JRE is installed. However, the instructions in this section are specific to installing the Java Console on Windows. See the HP Operations Manager documentation or the Java Console's Online Help (once you have installed the Java console on Windows) for instructions on additional platform installation instructions.

To install the Java console:

- 1 Ftp `/opt/OV/www/htdocs/ito_op/ITO_JAVA.exe` from the HP Operations Manager management server to the PC where you are installing the console.
- 2 Execute `ITO_JAVA.exe`. An install shield displays and guides you through the installation process.


Starting the Java Console

To start the Java Console:

- 1 Double-click the Java Console icon on the desktop.
- 2 Enter the following information in the HP Operations Manager Operations Login dialog box (if a different operator is configured, use that operator name instead of `opc_adm`):

— **User Name:** `opc_adm`

— **Password:** `<opc_adm password>` default password is `OpC_adm`

- **Management Server:** The full DNS name of the HP Operations Manager management server system
 - 3 Click **OK**. The Java Console starts and displays information for your HP Operations Manager environment.
 - 4 From the Object Pane, expand the **Services** node to view SOAM services.
 - 5 From the Object Pane, expand the **Message Groups** node to view the SOAM message group.
-  If you are new to the Java console, see the HP Operations Manager documentation or the Java Console's Online Help.

Uninstalling the Plug-in

To uninstall the SOA Manager Operations Manager Plug-in:

- 1 From a command prompt, run:

```
swremove SOAMSPI
```
- 2 Delete `<plugin_install_dir>`.
- 3 Delete the `/tmp/soa-map.xml` file.

Performing Standard Management Functions

This chapter provides some tasks that are typically performed when using the SOA Manager Operations Manager Integration. In particular, the following sections are included:

- Service Management
- Alert Message Management
- Using XPL Logging

Service Management

Service management is achieved using the Service Navigator that is included in the HP Operations Manager Java Console. During runtime, the SOA Manager Operations Manager Integration automatically discovers SOA Manager managed resources and represents them as a Service Map. Any deployment changes that occur in the SOA Manager environment are dynamically synchronized with the Service Map.

The Service Navigator also allows detailed management of the resources that are presented in the Service Map. The detailed management includes browsing the managed resource hierarchy, their relationships, attributes, metrics, and invocation of methods that are exposed by the resources.

Instructions for installing and starting the Java Console (including the Service Navigator) are located in Chapter 2. In addition, Online Help is available for the Java Console.

To view the service map:

- 1 Start and log in to the Java Console.
- 2 From the Object Pane, expand the **Services** node to view SOAM business services.
- 3 Right-click a business service and select **Show Graph**. The service graph displays and shows the business service model.

Starting the HP SOA Manager Web Interface

The HP SOA Manager web interface can be started from the HP Operations Manager Java Console and is a convenient way to troubleshoot business services from the console.

To start the HP SOA Manager web interface console from the HP Operations Manager Java Console:

- 1 From the Object Pane, expand the **Services** node to view SOAM business services.
- 2 Right-click a business service and select **Start --> SOA Manager GUI**. The HP SOA Manager web interface is displayed in a new Workspace window.

Alert Message Management

All alert messages generated through SOA Manager are captured and displayed in the Java Console's message browser.

To view SOA Manager alert messages:

- 1 Start and log in to the Java Console.
- 2 From the Object Pane, expand the **Message Groups** node to view the SOAM message group.
- 3 Select the SOAM message group. The message browser displays and lists all SOA Manager alert messages.
- 4 Right-click a message and select **Properties**. The Message Properties dialog box displays and lists additional details about the alert message.

Acknowledging Alert Messages

Alert messages are acknowledged using standard HP Operations Manager Console procedures and can also be acknowledged in the SOA Manager HP SOA Manager web interface. Alerts that are acknowledged in the HP Operations Manager console are automatically removed from the alert list in the HP SOA Manager web interface console and vice versa.

To acknowledge an alert message in the HP Operations Manager console, right-click on a message and select **Acknowledge**. The alert message is removed from the list of alert messages. Check the HP SOA Manager web interface Console to verify that the alert message has been removed from the SOA Manager Server as well.

Changing Message Properties

SOA Manager alert messages are organized in HP Operations Manager by the message group name (this field is named `MsgGrp`) and the Application name. The default values of these fields are respectively `SOAM` and `SOA Manager`. A custom message group and application name can be configured using the Plug-in property file.

To change message properties:

- 1 Stop the Plug-in if it is currently started.
- 2 Using a text editor, open `<plugin_install_dir>/soam-ovou.properties`.
- 3 Configure the following properties:
 - **MSG_APPLICATION**: Enter an application name. The name displays in the Application field.
 - **MSG_GROUP**: Enter a message group name. The name displays in the `MsgGrp` field and on the Object pane. The Message group must be a valid HP Operations Manager message group. See “Creating a Message Group” in Chapter 2. If the name entered is not a valid message group name, the messages are displayed as Misc (miscellaneous) messages.
- 4 Save and close `soam-ovou.properties`.
- 5 Start the Plug-in.
- 6 Start (or refresh) the HP Operations Manager Console and verify that the Application and `MsgGrp` fields are updated.

Using XPL Logging

The Plug-in uses HP Operations Manager Cross Platform (XPL) logging. Log messages are displayed in the terminal window where the Plug-in is started and written to the `log` subdirectory of the HP Operations Manager data directory (typically `/var/opt/OV`). The log file name has the format:

soam-ovou[unique].sequence.locale

For example:

`soam-ovou0.0.en_US`

This file is the first plug-in log file created for the US English locale.

XPL creates a log file for an English locale and a second file for your system’s locale if it is different from English.

XPL 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.

Configuring Log Levels

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

Using JRE Properties File

You can change the log level for Plug-in by editing the `logging.properties` file in the `JRE /lib` directory. For example:

```
com.hp.ov.mip.mip.rdf.client.RDFModelClient.level = INFO
```

This sets the log level to INFO. You must restart the Plug-in for the change to take effect.

Using the XPL Properties File

You can change the log level for the Plug-in by editing the `<plugin_install_dir>/xpllogging.properties` file. For example:

```
com.hp.ov.mip.mip.rdf.client.RDFModelClient.level = INFO
```

This sets the log level to INFO. You must restart the Plug-in for the change to take effect.

Troubleshooting

This chapter provides common troubleshooting tasks when using the SOA Manager Operations Manager Plug-in. In addition, refer to the *SOA Manager Release Notes* for the latest information about the SOA Manager Operations Manager Integration.

Startup Errors

The Plug-in does not start and the following log message displays:

```
Error starting event listener: java.net.BindException: Address
already in use
```

Solution:

This error occurs when either another instance of the Plug-in is currently started or when a different process is already using the port. Use the `<plugin_install_dir>/stop-soam-ovou.sh` to stop the Plug-in and then restart the Plug-in.

If the problem still occurs, use the `ps` command to manually kill the `soam-ovou` process or any other process that is using the port.

The plug-in starts with the following error:

```
Unexpected Exception when trying to fetch model from Network
Services: AxisFault

    faultCode:
{http://schemas.xmlsoap.org/soap/envelope/}Server.userException
    faultSubcode:
    faultString: java.net.ConnectException: Connection refused
    faultActor:
    faultNode:
```

Solution:

This error occurs when the URL for the Network Service server is invalid or a connection to the server failed.

- 1 Stop the plug-in.
- 2 Open `<plugin_install_dir>/soam-ovou.properties` and verify that the `NS_BASE_URL` is set to a valid host and port.
- 3 Save and close the properties file.
- 4 Start the Plug-in.

Message Errors

SOA Manager messages display in the `Misc` message group in the Java Console Message Browser.

Solution:

- 1 Stop the plug-in.
- 2 Open `<plugin_install_dir>/soam-ovou.properties`
- 3 Verify that the `MSG_GROUP` property is set to a valid HP Operations Manager message group and that the message group is assigned to the operator. See the “Creating a SOAM Message Group” section in Chapter 2.
- 4 Save and close the properties file.
- 5 Start the Plug-in.

HP BPI Integration

HP Business Process Insight (BPI) is one of the HP Software products that you can integrate with SOA Manager. You can configure instances of the HP BPI SOA Manager adapter to integrate with your SOA Manager installations. You can then configure HPBPI to link SOA Manager business events to the business flows that you define. HP SOA Manager enables you to manage your service oriented architecture (SOA) resources to ensure their reliability and optimize their performance. The combination of HP SOA Manager and HP Business Process Insight provides you with the ability to monitor the health and performance of services running within a Service Oriented Architecture. This chapter provides an overview of the integration, configuring, and troubleshooting instructions when integrating HP BPI with HP SOA Manager. Refer to the HP Business Process Insight (BPI) manuals for more information about detailed instructions and conceptual information regarding HP BPI.

Integration Instructions

The HP BPI-SOA Manager integration works as follows:

- Within the OVBPI Modeler, you are able to specify the SOA Manager Business Services on which your business flow depends.
- You then deploy your flow to the HP BPI Business Impact Engine.
- As these Business Services change state, this status information is collected from SOA Manager and your HP BPI flow reflects any resultant business impact.

For SOA Manager to be able to communicate with HP BPI you need to install and run the OVBPI SOA Manager Adapter component. This SOA Manager Adapter component can be installed on any machine that has access to your network, but typically you would install the SOA Manager Adapter component on the SOA Manager server. Refer to the BPI Installation Guide for more details.

Let's consider the architecture where you have installed HP BPI and the SOA Manager adapter on different servers where:

- You install and configure the HP BPI SOA Manager Adapter to talk to your SOA Manager installation.

- Within the HP BPI Administration Console, you configure a connection to your HP BPI SOA Manager Adapter.
- You make sure the HP BPI Service Adapters component is running, as this is the component that handles the BPI side of the SOA Manager connection. This communication uses the OpenView Web Services interface.
- When defining a flow within the HP BPI Modeler you specify the names of the SOA Manager Business Services that you wish to use. The HP BPI Modeler calls the HP BPI Service Adapters component (which, in turn, calls the HP BPI SOA Manager Adapter) to determine whether these services exist as Business Services within SOA Manager.
- You deploy your flow to the HP BPI Business Impact Engine.
- The BPI Service Adapters component polls the HP BPI SOA Manager Adapter to get status details about the SOA Manager Business Services. Any status changes for these Business Services are passed through to the HP BPI Business Impact Engine.

Configuration Instructions

Configuring the HP BPI SOA Manager integration requires some set up on both the SOA Manager server and the HP BPI server. Even if you have installed HP BPI on the same server as SOA Manager, these steps are still required:

On the HP BPI Server

You configure the SOA Manager connection details using the HP BPI Administration Console:

- 1 Select **Operational Service Sources** from the Navigation tree in the Administration Console. The right-hand pane shows a list of Operational Service Sources.
- 2 From the right-hand pane, select the **Add** button to add a new Service Source for the SOA Manager adapter instance that you have created. You are presented with the Service Oriented Architecture Manager Source Properties dialog box.
- 3 Enter values for the properties of the SOA Manager Service Source. The properties are fully described in the OpenView Business Process Insight Administration Guide:
 - Service Source Name
 - Description (Optional)
 - Product Name, which is set as Service Oriented Architecture Manager. The interface allows for additional Service Sources to be added in future versions of BPI.
 - Host name
 - Port
 - Status Event Poll Interval In addition, you can configure a Web Proxy for your Web Services connection if you have on

- 4 Click the **OK** button when you modifications are complete.
- 5 Make sure that the **Enabled** check box is selected in the column next to the new service source entry.
- 6 Click the **Apply** button to apply your changes
- 7 Move to the Component Status screen and stop and restart all the BPI components. The configuration is now complete and you can access SOA Manager business services from HP BPI.

On the SOA Manager Server

You need to install the HP BPI SOA Manager Adapter and configure it to talk to your SOA Manager installation. Once the BPI SOA Manager Adapter is installed and running, it is available to HP BPI.

Prerequisites for HP BPI- SOA Manager Adapter Installation

The SOA Manager adapter needs be installed only if you want to use SOA Manager as a source of operational events. The adapter can be installed on any Windows machine and configured to access the machine where SOA Manager is running.

You install the SOA Manager Adapter files from the zip archive provided on the HP BPI distribution media.



You can configure and start only one instance of the SOA Manager adapter for each installation. Starting multiple instances of the adapter from the same installation can have unpredictable results.

If you want to run multiple instances of the adapter on one machine, you can achieve this by installing the adapter multiple times and starting one adapter instance from each installation.

You need the following information available to install and set up the adapter on the SOA Manager machine.

Information	Notes
The name of the SOA Manager Web Service that contains the catalog of web services that it exposes.	<p>This is the SOA Manager Web service that starts with the following string: WsmfServiceCatalog...</p> <p>The SOA Manager Web services are listed at: http://hostname:port/wsmf/services where:</p> <ul style="list-style-type: none"> • hostname, is the host name of the machine where SOA Manager is running. This is described as the next parameter in this table. • port, is the port number used by SOA Manager to publish its Web Services. This is described more fully later in this table.

Hostname of the machine where SOA Manager is running	If no hostname is specified, a value of local host is assumed.
Port number used by SOA Manager to publish its Web Services	If no port number is specified, a value of 5002 is used.
Port Number used to publish the SOA Manager adapter as a Web Service. This is the Axis port number as installed with the adapter.	If no port number is specified, a value of 18097 is used.

Installing the HP BPI- SOA Manager Adapter

Complete the following steps to install the adapter:

- 1 Locate the zip archive file on the HP SOA Manager distribution media:

```
cd-root\i386\soam-adaptor.zip
```

- 2 Copy the zip archive file to the machine where you want to install the SOA Manager adapter.
- 3 Create a new directory for the SOA Manager Adapter zip archive file.
- 4 Unpack the zip archive file into the directory that you have created in the previous step.
- 5 Set the path for the Java Home directory for the adapter in the environment variable SOAMADAPTER_JAVA_HOME. Open a Command Window and enter:

```
set SOAMADAPTER_JAVA_HOME=java-install-dir
```

where java-install-dir is the location of your Java installation, for

```
example: c:\program files\java\jdk1.5.0_08
```

- 6 From a Windows Command Window, locate the following script to configure and start the adapter:
- 7 From the directory where the script is located, run the following script in a new Command Window to configure and start the adapter as follows:

```
runAdapter.bat
```

```
start runAdapter -csoa-svs-catalog -h hostname -swsvs-port -aadport
```

where:

```
-c soa-svs-catalog (required parameter)
```

- c takes a parameter `soa-svs-catalog`, which is the name of the SOA Manager Web Services Catalog. This is the catalog name that you identified starting with the string `WsmfServiceCatalog`.
- h `hostname` (optional parameter)
- h takes a parameter, which is the fully qualified host name of the machine where SOA Manager is running. This parameter is optional and if it is not specified, a value of `localhost` is assumed.
- s `wsvs-port` (optional parameter)
- s takes a parameter, which is the port number used by SOA Manager to publish its Web services. This parameter is optional and if it is not specified a value of `5002` is used.
- a `adport` (optional parameter)
- a takes a parameter, which is the port number used by Axis to publish the SOA Manager Adapter as a Web Service. This parameter is optional and if it is not specified, a value of `18097` is used.

An instance of the SOA Manager Adapter is now installed, configured and started. Do not close the Command Window where you started the adapter, or you shut down this adapter instance.

You also need to configure the adapter as an operational service source for HP BPI. You do this using the Administration Console as described in the OpenView Business Process Insight Administration Guide.

You can check that the adapter is configured and running at any time using the following URL:

`http://hostname:18097/axis/services/SOAMAdapter?wsdl` where `hostname` is the name of the machine where the adapter is installed. If the browser returns an error page, then the adapter is not running.

Stopping the SOA Manager Adapter

You stop the adapter using `CTRL/C` in the Command Window where you started the adapter.

SOA Manager Adapter Log Files

The SOA Manager Adapter logs errors and warnings in the following log file:

`adapter-install-dir/data/log`

Troubleshooting HP BPI Integration

If the BPI Dashboard returns an error similar to the following when you attempt to link to a SOA Manager-defined Service, it is likely to be because you are using a version of the Business Process Dashboard that does not recognize the Service definition:

File: `dash1-1_error.gif`

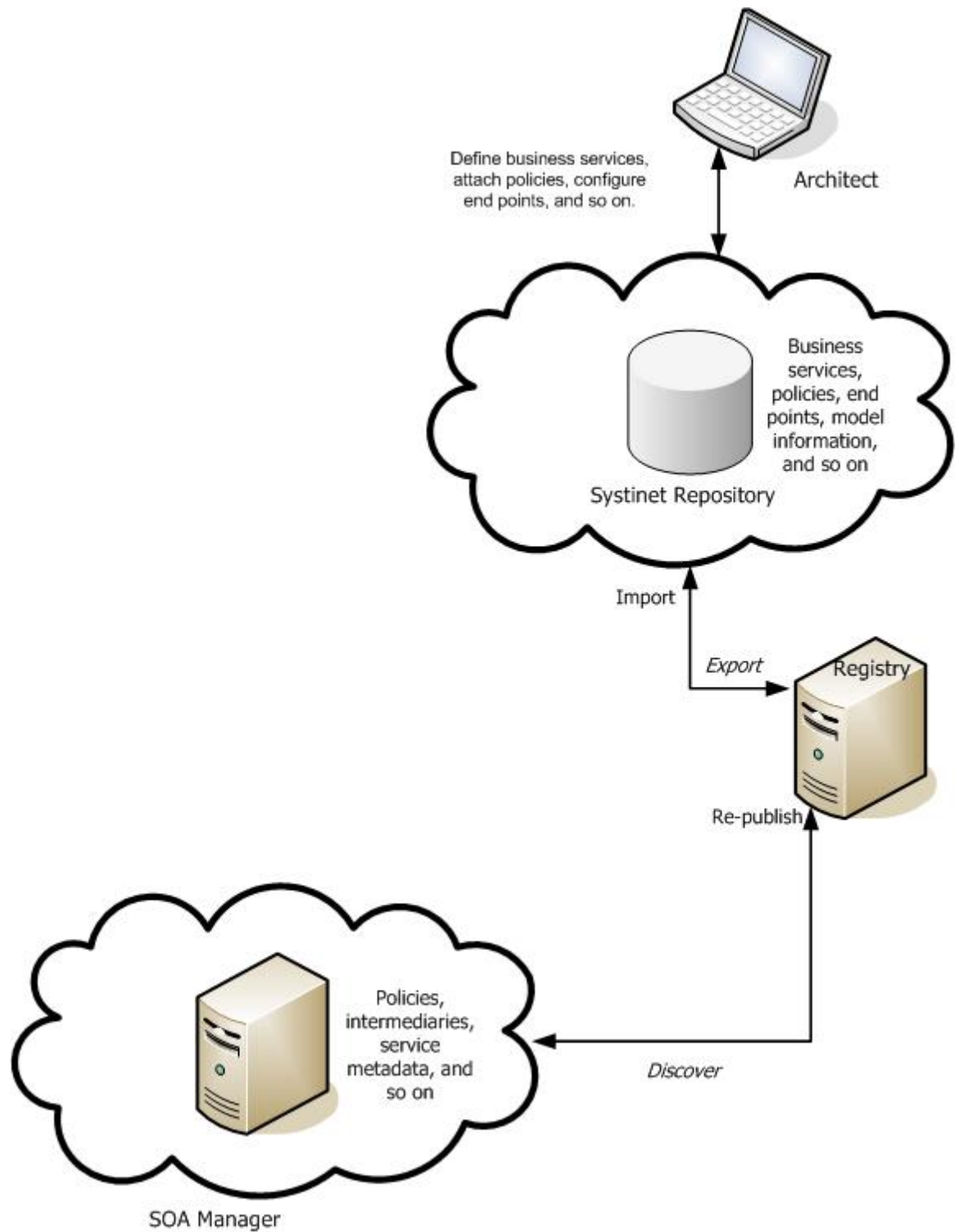
In this example, you are running a Business Process Dashboard based on OVBPI version 01.01. The integration with SOA Manager was introduced in OVBPI version 2.10. As a result, the OVBPI Dashboard version 01.01 does not understand the SOA Manager Service and is not able to render it.

If you want to show SOA Manager Services, you need to upgrade your Business Process Dashboard to version 02.10.

HP SOA Systinet Integration

HP SOA Manager 2.51 together with HP SOA Systinet 2.52, known as the HP SOA solution provides a complete solution across the lifecycle of SOA and allows to achieve end-to-end SOA governance. The HP SOA solution offers you a complete integrated SOA governance environment. HP provides end-to-end design-time and run-time governance by integrating HP SOA Manager with HP SOA Systinet using GIF (Governance Interoperability Framework). The GIF is designed to provide a standards-based approach to publishing and discovering business services information in a SOA across multiple vendor products. GIF improves SOA visibility, governance, and lifecycle management. During a design phase, as an architect, you can define your business services, attach policies to the service, maintain information about the service model, end points, and so on (collectively known as artifacts) using HP SOA Systinet. HP SOA Systinet maintains all this information in a location known as the repository. Systinet publishes the artifacts to a registry (UDDIv3 compliant).

You can use HP SOA Manager to implement run time governance and management of the SOA environment. HP SOA Manager allows you to receive information about published artifacts from the registry. You can use SOA Manager to associate more policies to the published Web service, retrieve service metadata, associate Intermediaries to the Web service to bring the service under management and then publish the updated Web service back to the registry. The figure below illustrates this cycle:



Service artifacts exchange between HP SOA Manager and HP SOA Systinet

- Configure the location of the Systinet registry in SOA Manager
- Publish the runtime policies to Systinet
- Model the services in the Systinet by defining the business service, implementation service and policy information.

- Discovery of services from Systinet by SOA Manager
- Provision the discovered service to bring under management (configure last mile for policies, specify policy enforcement point location, saving and or deployment to policy enforcement point)
- Publish proxy access point and changed policy association to Systinet
- View Service Metrics from Systinet

Mapping Artifacts- SOA Manager and Systinet

The following sections provide you the conceptual information to understand the integration scenario. HP SOA Systinet business service is represented as `hpss:business` service and the business service in registry is represented as `uddi:business` service in the following sections.

The `hpss:business` service contains the following details:

- The implementation services that contain:
 - Definition of the service
 - Associated policies
- Endpoints (known as binding templates in Systinet)

Publishing Artifacts from Systinet to Registry

When you publish a business service from Systinet to the Registry, the Registry maps and maintains information about the `hpss:business` service as follows:

- TModel corresponding to the `hpss:business` service
- `uddi:business` service corresponding to the `hpss:implementation` service
- Policies associations as `uddi:keyed` references in `uddi:business` services category bag
- Endpoint information as a `uddi:binding` templates for the `uddi:business` service

Retrieving Business Services from the Registry Using SOA Manager

You can configure SOA Manager to receive subscription notifications about services that are published to the registry from Systinet. The mapping of the registry artifacts to SOA Manager artifacts is as follows:

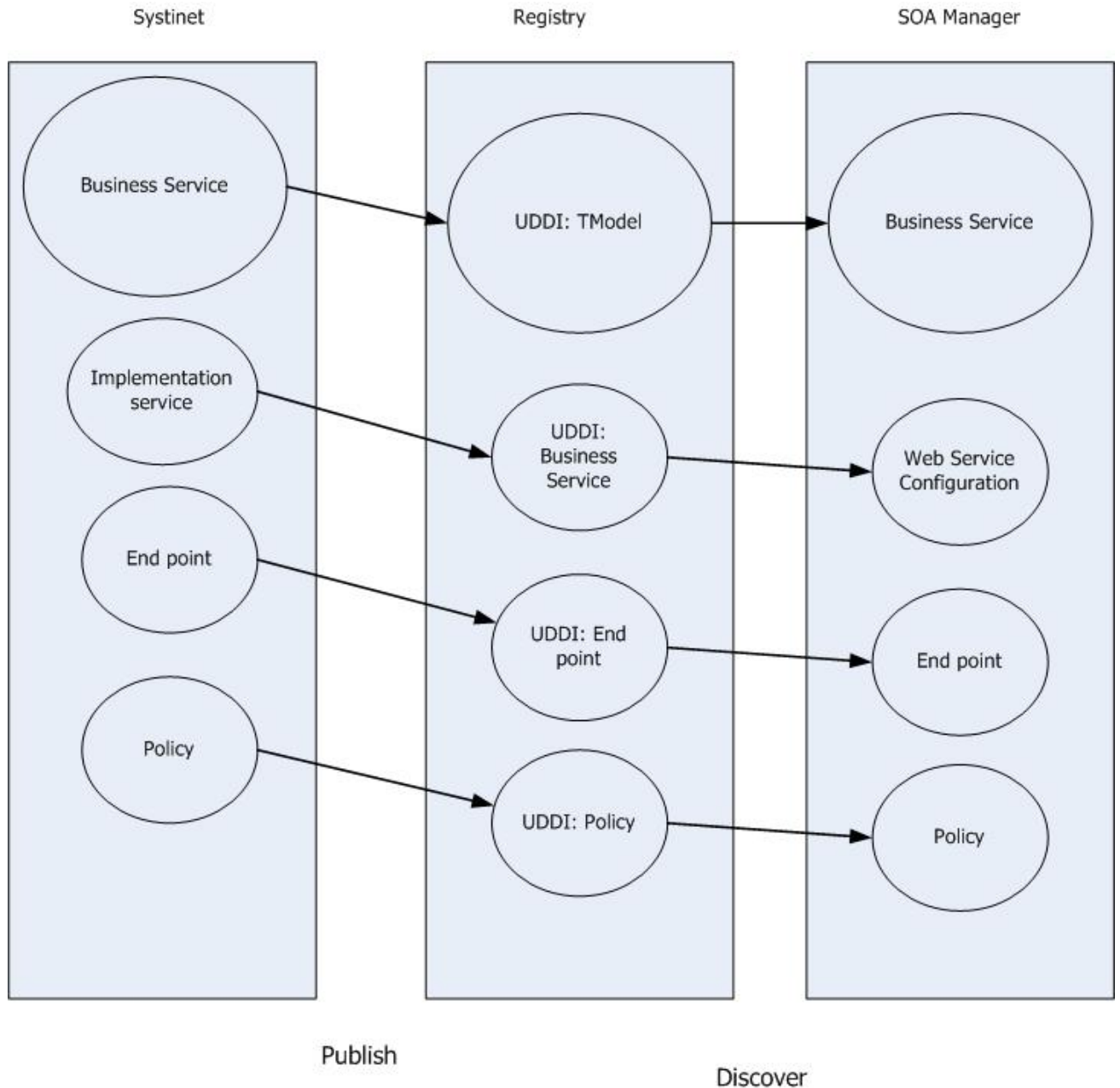
- TModel corresponding to the `hpss:business` service maps to `soam:business` service configuration
- `uddi:business` service maps to `soam:webservice` configuration
- Policies associated to the `uddi:business` service maps to policies associated to `soam:web` service

Note: End points are not directly mapped to SOA manager artifacts, but during provisioning, you can select the binding templates from the registry that you want to use for provisioning the service.

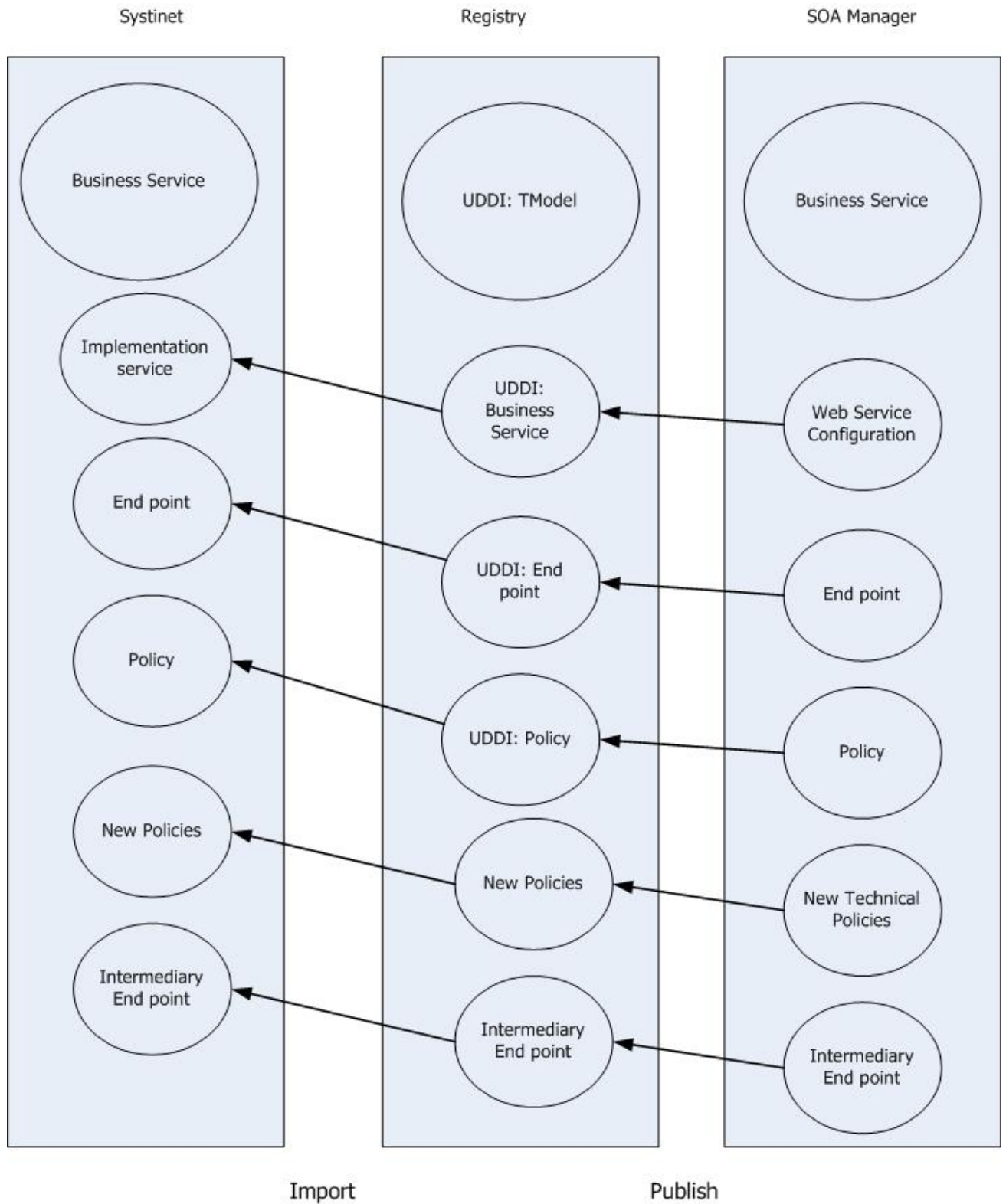
This sends the business service to SOA Manager along with the information about the policies, and endpoints.

You must provision the service retrieved from the registry using SOA Manager. You can associate additional policies and deploy the service to a policy enforcement intermediary before publishing the service back to the registry. The policy enforcement intermediary acts as the proxy for the end point that was configured initially for the business service from Systinet. This improves governance and re-usability of policies. Importing Artifacts to Systinet

The figure below illustrates how the artifacts are published from Systinet to the registry and then imported by SOA Manager.



The figure below illustrates how an imported service from the registry is updated with additional artifacts and then republished back to the registry. Systinet then imports the service from the registry.



Mapping SOA Services Model to UDDI

The SOA Manager's assets that are published to a UDDI registry can be leveraged by any application that can integrate with UDDI. This section provides a reference of how the assets are mapped in UDDI.

In particular, this section includes mappings for:

- Web services
- Technical policies

SOA Manager works with only Systinet UDDIv3 registry. You must synchronize the Systinet Registry taxonomies with the HP SOA Systinet before configuring the registry settings in SOA Manager.

Define **Managed Endpoint** represents the service managed by a Web Service Management System. It represents a proxy of the Functional Endpoint.

Functional Endpoint represents the service exposing functionality managed by a WSMS. The service is deployed on an application server.

Overview

When you publish a business service, the web services contained in that business service along with the policies associated to it are published to the registry.

Role of UDDI in the SOA

SOA implementations use UDDI as a system of record. The positioning of the role of registry technology (such as UDDI) in an SOA has evolved over the past few years since its inception. Originally, the registry was conceived as a central discovery point for design-time and run-time reuse.

However, most recent thinking in this direction is that if the registry is used as the only way to offer and discover Web services in the SOA, it provides an excellent control point to achieve Governance during various stages of development, deployment, and runtime management. Business stakeholders and enterprise architects define various policies that must be adhered to in the enterprise SOA. These policies are captured and attached to various entities in UDDI.

The UDDI Registry is used to achieve the following:

- **Reuse** – capture meta-data about Web services as well as other technology assets so that effective search capabilities from various environments may be written against the registry.
- **Policy Definition** – capture various policy definitions that provide the ability for a business person or enterprise architect to mandate policies on various entities.
- **Capture SOA Environment Model** – capture various entities participating in an SOA; their relationships; and some meaningful subset of state information about these entities. This information is used to create a standardized/normalized information store that is used to support various IT governance processes.
- **Integration** – the agreement of various participants in the SOA to settle upon ontologies that are linked together in the UDDI Registry creates very good potential for different participants to consume and populate information out of UDDI for their own narrow domains. The registry then provides a central store to create the ultimate spider that links together all this information.

Technical Policy Mapping

Technical policies defined in SOA Manager are published to UDDI registry according to : Web Services Policy Attachment, Version 1.2.

According to this specification, reusable policy expressions are registered in UDDI registry as distinct TModels. SOA Manager publishes all policies except routing and load balancing policy as distinct TModels. An example TModel is as follows:

```
<tModel tModelKey="uddi:469fef70-ac75-11dc-a342-3b4e75e1a340"
deleted="false" xmlns="urn:uddi-org:api_v3">
  <name>SchemaPolicy</name>
  <description>SchemaPolicy</description>
  <overviewDoc>
    <description>WS-Policy Expression</description>

<overviewURL>http://nt11812.asiapacific.hpqcorp.net:5002/bse_refresh/
PolicyExpression.jsp?cwcPopup=true&policy=SchemaPolicy</overviewU
RL>

  </overviewDoc>
  <categoryBag>
    <keyedReference
tModelKey="uddi:schemas.xmlsoap.org:policytypes:2003_03"
keyName="policy" keyValue
```



```

<keyedReference
tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
keyName="Policy Expression for SchemaPolicy"
keyValue="http://nt11812.asiapacific.hpqcorp.net:5002/bse_refresh/PolicyExpression.jsp?cwcPopup=true&policy=KiranSchemaPolicy"/>

    <keyedReference
tModelKey="uddi:systinet.com:soa:model:taxonomies:policyTypes"
keyName="Runtime"
keyValue="uddi:systinet.com:soa:model:taxonomies:policyTypes:runtime"
/>

    <keyedReference
tModelKey="uddi:systinet.com:soa:model:taxonomies:associatedApplication"
keyName="Other"
keyValue="uddi:systinet.com:soa:model:taxonomies:associatedApplication:other"/>

</categoryBag>
</tModel

```

Apart from the taxonomies mentioned in the WS Policy Attachment Specification, additional taxonomies are published to indicate the policy type as runtime and associated application as not 'Policy Manager'

Web Service Mappings

When a web service is published to registry, the information published to registry depends on whether the web service was originally created in SOA Manager or Systinet.

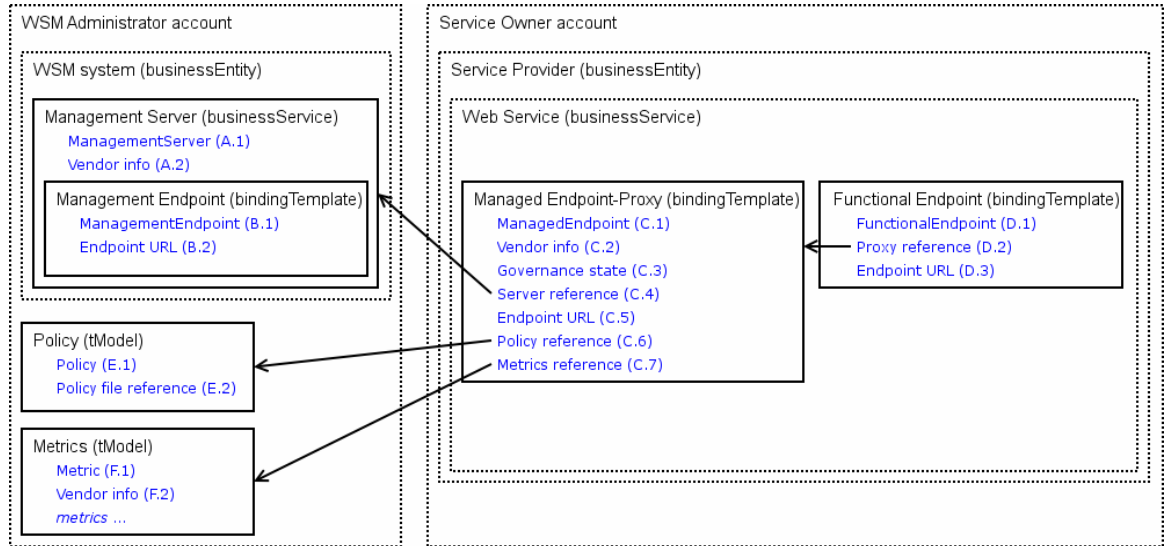
For web services created in SOA Manager initially, the following is published :

The WSDL for the proxy web service. The WSDL is published according to: *Using WSDL in a UDDI Registry, Version 2.0.2 – Technical Note*. The following UDDI entries are created:

- UDDI business service for the WSDL service
- UDDI binding template for the WSDL port
- UDDI tModel for each binding and port type
- The WSDL for the proxy Web service. The Web service WSDL is published to the UDDI registry if it does not exist in the registry already. It is published according to the technical note cited above.
- Relationships. The relationships are published as keyed references in the UDDI business service category bag. The following types of relationships are published:
 - **Reusable Policies** – A UDDI business service contains a keyed reference to indicate the reusable policies attached to the web services. Local Policy Reference taxonomy as mentioned in WS Policy Attachment specification is used to indicate this relationship
 - **Non Reusable Policies** – A UDDI business service contains a keyed reference to indicate the web service specific policies attached to it. Remote Policy Reference taxonomy as mentioned in WS Policy Attachment specification is used to indicate this relationship.

NOTE: For discovered services from systinet, the UDDI business service representing the web service is expected to be already registered in registry. Only the policy association relationship and the binding template information is updated in registry

For discovered services from systinet, the managed endpoint is published as per GIF. The figure shown below explains the mapping,



While publishing the managed endpoint following steps are followed by SOA Manager,

- The functional service's bindingTemplate (BT1) is copied to a new bindingTemplate (BT2) contained by the same businessService
- bindingTemplate (BT2) is updated with a reference to bindingTemplate (BT1) using "uddi:systinet.com:management:proxy-reference" taxonomy
- BT1's accessPoint is updated with the proxy endpoint
- Both bindingTemplates are updated with additional categorizations

An example is shown below:

```
<businessService
  serviceKey="uddi:example.com:myService:bs "
  businessKey="...">
  <name>My service</name>
  <bindingTemplates>
    <bindingTemplate
      bindingKey="uddi:example.com:myService:bt "
      serviceKey="uddi:example.com:myService:bs">
      <accessPoint URLType="http">http://example.com/myServiceProxy</accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo>
          ...
        </tModelInstanceInfo>
        ...
      </tModelInstanceDetails>
      <categoryBag>
        <keyedReference
```

```

    tModelKey="uddi:systinet.com:management:system"
    keyName="Management System"
    keyValue="HP SOA Manager" />
  <keyedReference
    tModelKey="uddi:systinet.com:management:type"
    keyName="Management entity type"
    keyValue="managedEndpoint" />
</keyedReference>
  <keyedReference
    tModelKey="uddi:systinet.com:management:state"
    keyName="Governance state"
    keyValue="managed" />

  <keyedReference
    tModelKey="uddi:systinet.com:management:url"
    keyName="URL from AccessPoint"
    keyValue="http://example.com/myServiceProxy" />
</categoryBag>
</bindingTemplate>

<bindingTemplate
  bindingKey="uddi:example.com:myService:functionalEndpoint"
  serviceKey="uddi:example.com:myService:bs">
  <accessPoint URLType="http">http://example.com/myService</accessPoint>
  <tModelInstanceDetails>
    ...
  </tModelInstanceDetails>
  <categoryBag>
    <keyedReference
      tModelKey="uddi:systinet.com:management:system"
      keyName="Management System"
      keyValue="HP SOA Manager" />
    <keyedReference
      tModelKey="uddi:systinet.com:management:type"
      keyName="Management entity type"
      keyValue="functionalEndpoint" />
    <keyedReference
      tModelKey="uddi:systinet.com:management:proxy-reference"
      keyName="Proxy reference"
      keyValue="uddi:example.com:myService:bt" />
  </categoryBag>
</bindingTemplate>
</bindingTemplates>
<categoryBag>

```

```

<
  <keyedReference
    tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
    keyName="Policy Expression for myServiceProxyRoutePolicy1"
    keyValue="https://soamanagerhost:5003/bse_refresh/PolicyExpression.jsp?cwcPopup=true&policy=myServiceProxyRoutePolicy1" />

  <keyedReference
    tModelKey="uddi:schemas.xmlsoap.org:localpolicyreference:2003_03"
    keyName="SchemaValidation"
    keyValue="uddi:a8666990-ac99-11dc-9cdd-a32db2519cdc" />

  <keyedReference
    tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
    keyName="Policy Expression for myServiceProxyLoadBalancingPolicy"
    keyValue="https://soamanagerhost:5003/bse_refresh/PolicyExpression.jsp?cwcPopup=true&policy=myServiceProxyLoadBalancingPolicy" />

```

```

</categoryBag>
</businessService>

```

For services created in SOA Manager the proxy binding template is published according to: *Using WSDL in a UDDI Registry, Version 2.0.2 – Technical Note*

An example is shown below,

```

<businessService serviceKey="uddi:37bd5bd0-ac9a-11dc-9cdd-a32db2519cdc"
businessKey="uddi:systinet.com:demo:hr" xmlns="urn:uddi-org:api_v3">
  <name>myServiceProxy</name>
  <bindingTemplates>
    <bindingTemplate bindingKey="uddi:37e83c60-ac9a-11dc-9cdd-a32db2519cdc"
serviceKey="uddi:37bd5bd0-ac9a-11dc-9cdd-a32db2519cdc">
      <accessPoint
useType="http">http://soamanagerhost:9032/myServiceProxy/myServiceSoapSoapBindin
g</accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo tModelKey="uddi:37783c80-ac9a-11dc-9cdd-a32db2519cdc">
          <instanceDetails>
            <instanceParms>myServiceSoapPort</instanceParms>
          </instanceDetails>
        </tModelInstanceInfo>
        <tModelInstanceInfo tModelKey="uddi:83814670-a944-11dc-bc79-f05f9301bc77"/>
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
  <categoryBag>
    <keyedReference tModelKey="uddi:uddi.org:wSDL:types" keyName="WSDL Entity
type" keyValue="service"/>
    <keyedReference tModelKey="uddi:uddi.org:xml:namespace" keyName="XML
namespace" keyValue="http://wsm.hp.com/myService"/>
    <keyedReference tModelKey="uddi:uddi.org:xml:localName" keyName="XML local
name" keyValue="myServiceProxy"/>
    <keyedReference
tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
keyName="Policy Expression for myServiceProxyRoutePolicy1"
keyValue="https://nttiju.asiapacific.hpqcorp.net:5003/bse_refresh/PolicyExpression.jsp?c
wcPopup=true&amp;policy=myServiceProxyRoutePolicy1"/>

```

```
<keyedReference tModelKey="uddi:schemas.xmlsoap.org:localpolicyreference:2003_03"  
keyName="SchemaValidation" keyValue="uddi:a8666990-ac99-11dc-9cdd-  
a32db2519cdc"/>
```

```
<keyedReference  
tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"  
keyName="Policy Expression for myServiceProxyLoadBalancingPolicy"  
keyValue="https://soamanagerhost:5003/bse_refresh/PolicyExpression.jsp?cwcPopup=tru  
e&amp;policy=myServiceProxyLoadBalancingPolicy"/>
```

```
</categoryBag>
```

```
</businessService
```



Appendix A Product Compatibility Matrix

Product Compatibility Matrix

The following table lists the HP Software products compatible for an integration with HP SOA Manager. The table also lists the versions compatible for the integration with HP SOA Manager.

Product	Version
HP Business Process Insight	2.10
HP Operations Manager for UNIX	8.1, 8.2
HP Select Access	6.1, 6.2
HP SOA Systinet	2.52 (hotfix)

A

- attachments, 3-5
- audit
 - architecture, 0-1
 - database, 0-1
 - messages, 0-1
 - service, 0-1
- audit integration, 0-1
- authentication, 0-4
 - basic, 0-1, 0-2
- authorization, 0-1, 0-4

B

- BaseXmlHandler interface, 3-5
- billing, 0-1
- Broker Configurator, 3-3
- brokered services
 - add custom handler, 3-7
 - definition, 3-3
 - jar, 3-3
 - manually implement, 3-3
 - xml introspection, 0-2
- BSE, 2-3
- business services, 2-3
 - UDDI mappings, 0-2

C

- contact distinguished name, 0-16
- contact email, 0-18
- contact name, 0-14
- contact relationship, 0-19
- custom policy handler, 3-5
 - add, 3-7
- custom security, 0-4
 - handlers, 0-6

D

- database

- message table, 0-3
- message trace table, 0-1
- schema, 0-1
- database integrations, 2-3
- dependency relationships UDDI mappings, 0-10
- document overview, 1-1

E

- event management, 0-11

F

- forecasting, 0-1

G

- general integration, 2-2

H

- handler. *See* policy handler
- hardware requirements, 0-4
- HP Operations Manager, 0-3
- HPjconfig tool, 0-4

I

- installation
 - plugin, 0-5
- interfaces
 - BaseXmlHandler, 3-5, 0-2, 0-6
 - RealmAuthoizationHandler, 0-4
 - startup tasks, 3-1
- introspection service, 0-1

J

- Java API integrations, 2-3
- Java console
 - installing, 0-9
 - starting, 0-9

K

- kernel parameters, 0-4

L

- logging, 0-12
 - levels, 0-13

M

- management console
 - Java console, 0-9
- management integration
 - database, 2-3
 - general, 2-2, 3-1
 - Java API, 2-3, 2-4
 - web services, 2-2
- management server, 0-5
- mappings, 0-2, 0-10, 0-14, 0-16, 0-18, 0-19
- message group
 - custom, 0-6
- message table, 0-3
- message trace table, 0-1

N

- non-repudiation, 0-1

O

- organizational contact information UDDI mappings, 0-14

P

- patches
 - software, 0-4
- plugin
 - starting, 0-7
- plug-in
 - installation HP-UX and SOLARIS, 0-5
- policy handler
 - add custom, 3-7
 - custom, 2-3, 3-5
 - custom security, 0-6
 - definition, 3-5
 - xml introspection, 0-2

R

- RealmAuthoizationHandler interface, 0-4
- reports, 2-3, 3-1, 0-4, 0-5
- requirements
 - hardware, 0-4

- SOAMI installation, 0-3
- software, 0-3
- software patches, 0-4

S

- security
 - custom, 0-4
 - custom handlers, 0-6
- security customization, 2-4
- Select Access, 2-4, 0-1, 0-6
- server.xml, 3-2
- service management, 0-10
- service security inbound handler, 0-3
- service.wsdl, 3-3
- service.xml, 3-3
- SLA
 - reports, 3-1
 - reports example, 0-4, 0-5
- SOAMI
 - hardware requirements, 0-4
 - installation requirements, 0-3
 - software requirements, 0-3
 - troubleshooting, 0-14
- SOAP attachments, 3-5
- software requirements, 0-3
 - patches, 0-4
- starting
 - plug-in, 0-7
- startup, 2-2, 3-1

T

- tModel, 0-10, 0-12, 0-14, 0-18, 0-19
- trace messages, 0-1
- troubleshooting, 0-14

U

- UDDI
 - data mapping reference, 0-1
- UDDI mappings
 - business services, 0-2
 - dependency relationships, 0-10
 - organizational contact information, 0-14

W

- web services
 - integrations, 2-2
- WSDL, 0-2, 0-5, 0-6, 0-8

X

- xml introspection service, 0-1
- XPL, 0-12
 - configure, 0-12