

HP SOA Systinet

Software Version: 3.00

Demo Guide

Document Release Date: June 2008
Software Release Date: June 2008



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.

Third-Party Web Sites

HP provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. HP makes no representations or warranties whatsoever as to site content or availability.

Copyright Notices

© Copyright 2003-2008 Hewlett-Packard Development Company, L.P.

Trademark Notices

Java™ is a US trademark of Sun Microsystems, Inc. Microsoft®, Windows® and Windows XP® are U.S. registered trademarks of Microsoft Corporation. IBM®, AIX® and WebSphere® are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries. BEA® and WebLogic® are registered trademarks of BEA Systems, Inc.

Contents

About This Guide.	5
How This Guide is Organized.	5
Document Conventions.	7
Documentation Updates.	8
Support.	9
1 REST API Demo.	11
2 RSS Demo.	13
3 SDM Demo.	17
4 Contract Demo.	21
5 Reporting Demo.	25
6 Validation and Report Rendering Demo.	27
7 Java Assertion Demo.	29
Creating the Assertion Validator.	29
Applying the Extension.	31
Creating and Deploying the Assertion.	31
Testing the Assertion Validator.	32

About This Guide

Welcome to HP SOA Systinet, the foundation of Service Oriented Architecture, providing an enterprise with a single place to organize, understand, and manage information in its SOA. The standards-based architecture of SOA Systinet maximizes interoperability with other SOA products.

- ▶ HP Software controls access to components of SOA Systinet with a license. This document describes the full functionality of SOA Systinet including licensed components. If your license does not include these licensed components, their features are not available.

How This Guide is Organized

This guide describes the demos provided with SOA Systinet. Running the demos is a good way to learn about SOA Systinet capabilities.



The demos require the following users to exist in the user store:

- Username `demouser` with password `changeit`
- Username `demoapprover` with password `changeit`

The users are created automatically by the installation if you are using JBoss with the JBoss user store.

- [Chapter 1, REST API Demo](#)

Demonstrates basic interaction with the REST API.

- [Chapter 2, RSS Demo](#)

Demonstrates the use of RSS by SOA Systinet to provide up to date repository information on the dashboard.

- [Chapter 3, SDM Demo](#)

Shows how to use *SDM* and its Java client to model *SOA*.

- [Chapter 4, Contract Demo](#)

Creates a provider, consumer and contract.

- [Chapter 5, Reporting Demo](#)

Demonstrates how to create tasks and perform them via the http interface.

- [Chapter 6, Validation and Report Rendering Demo](#)

Validates a resource against a business policy.

- [Chapter 7, Java Assertion Demo](#)

Shows how to create an assertion validator and an assertion and apply them as extensions to Policy Manager.

Each demo is in a directory in `SOA_HOME/demos/`.

Document Conventions

This document uses the following typographical conventions:

run.bat make	Script name or other executable command plus mandatory arguments.
<code>--help</code>	Command-line option.
either or	Choice of arguments.
<i>replace_value</i>	Command-line argument that should be replaced with an actual value.
{arg1 arg2}	Choice between two command-line arguments where one or the other is mandatory.
<code>rmdir /S /Q System32</code>	User input.
<code>C:\System.ini</code>	Filenames, directory names, paths and package names.
<code>a.append(b);</code>	Program source code.
<code>server.Version</code>	Inline Java class name.
<code>getVersion()</code>	Inline Java method name.
Shift+N	Combination of keystrokes.
Service View	Label, word, or phrase in a GUI window, often clickable.
OK	Button in a user interface.
New→Service	Menu option.

Documentation Updates

This guide's title page contains the following identifying information:

- Software version number, which indicates the software version.
- Document release date, which changes each time the document is updated.
- Software release date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support Web site at:

<http://www.hp.com/go/hpsoftwaresupport>

HP Software Support Online 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:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

1 REST API Demo

The purpose of this demo is to introduce the REST Java client and to show how to interact with SOA Systinet using this client. The basic operations CREATE, UPDATE, DELETE, UNDELETE, PURGE and GET are demonstrated.

The demo creates a collection and stores this to a collection sample XML document (taxonomy). After that the document is updated, deleted, undeleted and finally it is purged.

You can find the demo source code in: `SOA_HOME\demos\rest\api\src`

To run the REST API demo:

- 1 Ensure that the demo is properly configured and SOA Systinet is running.
- 2 Change your working directory to: `SOA_HOME\demos\rest\api`
- 3 To get help, execute: **run**
- 4 To build the demo, execute: **run make**
- 5 To run the demo, execute: **run publish**

The output of this demo resembles the following:

```
Publishing...
Probing secure port using 'https://localhost:8843/soa/systinet/platform/restBasic/repository/'...
...secure port OK
Running demo...
Creating demo collection...
  Successfully created!
Creating taxonomy...
  Successfully created:
https://localhost:8843/soa/systinet/platform/restBasic/repository/demo/demoTaxonomy.xml
Getting taxonomy...
  Successfully obtained:
  https://localhost:8843/soa/systinet/platform/restBasic/repository/demo/demoTaxonomy.xml
```

```
with root element
  {http://systinet.com/uddi/taxonomy/v3/5.0}taxonomy
Updating taxonomy...
  Successfully updated: https://localhost:8843/soa/systinet/platform/restBasic/repository/
demo/demoTaxonomy.xml?revision=2
Deleting taxonomy...
  Successfully deleted: https://localhost:8843/soa/systinet/platform/restBasic/repository/
demo/demoTaxonomy.xml?revision=2
Undeleting taxonomy...
  Successfully undeleted: https://localhost:8843/soa/systinet/platform/restBasic/repository/
demo/demoTaxonomy.xml?revision=2
Purging taxonomy...
  Successfully purged!
Demo successfully finished!
```

- 6 To unpublish the demo, execute: **run unpublish**
- 7 To rebuild the demo, execute **run clean** to delete the classes directory and **run make** to rebuild the demo classes.

2 RSS Demo

The purpose of this demo is to show how SOA Systinet uses RSS to provide updated information in Dashboard portlets.

The demo outputs the RSS view of the business collection before and after the publication of a new service.

This feature allows users to access the RSS view of repository data which can be syndicated by the SOA Systinet dashboard UI or any feed reader.

You can find the demo source code in: `SOA_HOME\demos\rest\rss\src`

To run the RSS demo:

- 1 Ensure that the demo is properly configured and SOA Systinet is running.
- 2 Change your working directory to: `SOA_HOME\demos\rest\rss`
- 3 To get help, execute: **run**
- 4 To build the demo, execute: **run make**
- 5 To run the demo, execute: **run publish**

The output of this demo resembles the following:

```
Publishing...
Running demo...
Get the RSS feed before creating a new BusinessServiceArtifact...
Result:
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/
rss/1.0/modules/content/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#
" xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/" version="2.0">
<channel>
<title>Collection businessServiceArtifacts</title>
<link>http://localhost:8080/soa/systinet/platform/restBasic/repository/bus
```

```

inessServiceArtifacts/?view</link>
<description>Feed for collection: businessServiceArtifacts</description>
<pubDate>Mon, 30 Apr 2007 18:54:50 GMT</pubDate>
<dc:creator>systinet:admin</dc:creator>
<dc:date>2007-04-30T18:54:50Z</dc:date>
<item>
<title>FTP Business Service</title>
<link>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/FTPServiceBs?view</link>
<description>Logical service describing FTP Service in business terms.<
/description>
<pubDate>Thu, 10 May 2007 10:04:35 GMT</pubDate>
<guid>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/FTPServiceBs?view</guid>
<dc:creator>demouser</dc:creator>
<dc:date>2007-05-10T10:04:35Z</dc:date>
</item>
</channel>
</rss>
Creating BusinesServiceArtifact...
Successfully created: http://localhost:8080/soa/systinet/platform/restBasic/rep
ository/businessServiceArtifacts/RssDemoBusinessService
Get RSS feed after update...
Result:
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/
rss/1.0/modules/content/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#
" xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/" version="2.0">
<channel>
<title>Collection businessServiceArtifacts</title>
<link>http://localhost:8080/soa/systinet/platform/restBasic/repository/bus
inessServiceArtifacts/?view</link>
<description>Feed for collection: businessServiceArtifacts</description>
<pubDate>Mon, 30 Apr 2007 18:54:50 GMT</pubDate>
<dc:creator>systinet:admin</dc:creator>
<dc:date>2007-04-30T18:54:50Z</dc:date>
<item>
<title>RssDemoBusinessService</title>
<link>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/RssDemoBusinessService?view</link>
<description>Business service published by REST rss demo</description>
<pubDate>Thu, 10 May 2007 13:17:47 GMT</pubDate>
<guid>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/RssDemoBusinessService?view</guid>
<dc:creator>demouser</dc:creator>
<dc:date>2007-05-10T13:17:47Z</dc:date>
</item>

```

```
<item>
<title>FTP Business Service</title>
<link>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/FTPServiceBs?view</link>
<description>Logical service describing FTP Service in business terms.<
/description>
<pubDate>Thu, 10 May 2007 10:04:35 GMT</pubDate>
<guid>http://localhost:8080/soa/systinet/platform/restBasic/repository/
businessServiceArtifacts/FTPServiceBs?view</guid>
<dc:creator>demouser</dc:creator>
<dc:date>2007-05-10T10:04:35Z</dc:date>
</item>
</channel>
</rss>
Demo successfully finished!
```

- 6 To unpublish the demo, execute: **run unpublish**
- 7 To rebuild the demo, execute **run clean** to delete the classes directory and **run make** to rebuild the demo classes.

The RSS output is the view available from the context menu action **RSS View** when viewing the business service collection from the **Tools** tab.

- Use http://localhost:8080/soa/systinet/platform/restBasic/repository/businessServiceArtifacts/rs&cs_artifactType=uncomsystinetsoamdeartifactssoabusinessService to navigate directly to the RSS view of the business service collection.



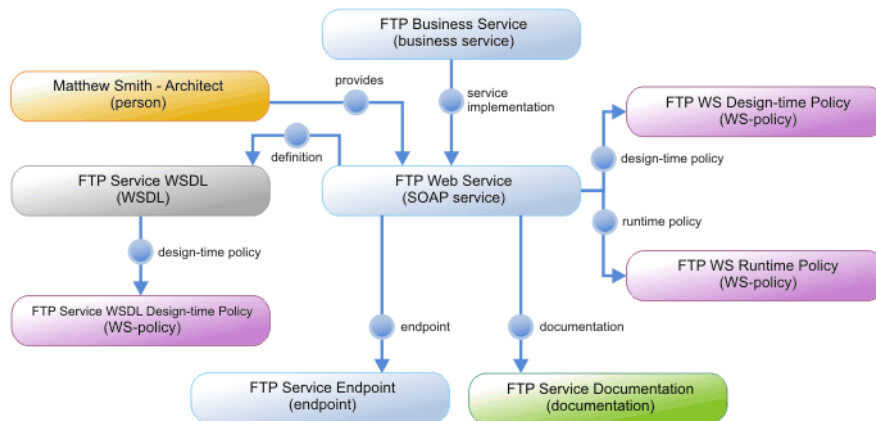
Change the hostname, port and context in the URL according to your installation settings.

- The RSS view url is the basis of the **Services** content feed portlet on the dashboard.

3 SDM Demo

The purpose of this demo is to show how to use the SDM Java API to interact with SOA Systinet REST http interface. The example publishes artifacts relating to a Business Service artifact. The business service described enables use of the FTP protocol via a standard web service interface.

Figure 1. SDM Demo Artifacts



These artifacts are published by the demo:

- **FTP Business Service:** The Business Service artifact represents a logical service and contains descriptive information about the service in business terms. Technical information can be found in business service implementations.
- **Matthew Smith Architect.** SDM distinguishes two types of contacts – *persons* and *organizational units*. Matthew Smith is represented by a Person artifact. The fact that he is responsible for the *FTP service* in the SOA system is expressed using the `provides` relationship.

- **FTP Web Service.** There is only one implementation of the *FTP Business Service* in the demo. It is represented by the SOAP Service artifact and both artifacts are associated using the **service** relationship.
- **FTP Service Endpoint.** The actual implementation of the service is linked to the FTP Web Service artifact via the endpoint artifact. This contains the address of the implementation.
- **FTP Service WSDL.** Technical details of the SOAP Service artifact are described not only in its metadata descriptor but also using references to other artifacts. The data of the WSDL artifact is a WSDL document describing the *FTP Web Service*. It is attached to the SOAP Service artifact using the **definition** relationship.
- **FTP Service WSDL Design-time policy.** SDM allows the attachment of policies describing various aspect of the artifacts directly to them using relationships. Such policies may contain assertions describing compliance, change management preferences or other information.
- **FTP WS Runtime Policy.** Other policy information mentioned in the previous paragraph may for example specify limits on the number of requests to be issued against the FTP Web Service.
- **FTP Service Documentation.** The SDM allows a number of documents to be attached to a web service in a Documentation artifact. These typically describe the web service and related artifacts.

You can find the demo source code in: `SOA_HOME\demos\sdm\ftpservice\src`

To run the SDM demo:

- 1 Ensure SOA Systinet is running and that the demo environment and properties are correct for your installation.
- 2 Change your working directory to: `SOA_HOME\demos\sdm\ftpservice`
- 3 To get help, execute: **run**
- 4 To build the demo, execute: **run make**
- 5 To run the demo, execute: **run publish**

The output of this demo resembles the following:

```
Publishing...
Publishing FTP Service SDM model...
```

```

Going to publish WS-Policies...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/ws
Policies/ftpServiceWsRuntimePolicy.xml
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/ws
Policies/ftpServiceWsDesignTimePolicy.xml
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/ws
Policies/ftpServiceWsdDesignTimePolicy.xml
Going to publish WSDL...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/ws
dls/ftpService.wsdl
Going to publish Endpoint...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/en
dpointArtifacts/FTPServiceEndpoint
Going to publish documentation...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/do
cumentation/ftpServiceDocumentation.pdf
Going to publish Web Service...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/se
rviceArtifacts/FTPServiceWs
Going to publish contact...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/co
ntactArtifacts/MatthewSmithArchitect
Going to publish Business Service...
Published: http://localhost:8080/soa/systinet/platform/restBasic/repository/bu
sinessServiceArtifacts/FTPServiceBs
Getting Web Service: http://localhost:8080/soa/systinet/platform/restBasic/repos
itory/serviceArtifacts/FTPServiceWs
FTP Service Web Service descriptor:

<a:webServiceArtifact ...>
  ...
</a:webServiceArtifact>
Getting WSDL:
https://localhost.localdomain:8843/soa/systinet/platform/restBasic/repository/wsdl
s/ftpService.wsdl
FTP Service WSDL descriptor:

<a:wsdlArtifact ...>
  ...
</a:wsdlArtifact>
FTP Service Endpoint descriptor:

<a:endpointArtifact ...>
  ...
</a:endpointArtifact>
FTP Service successfully published!

```


- 6 To unpublish the demo, execute: **run unpublish**

You can then publish again. To unpublish and publish in a single step, execute **run republish**


This is useful if you want to experiment by making changes to the demo.

- 7 To rebuild the demo, execute **run clean** to delete the classes directory and **run make** to rebuild the demo classes.

Once the demo data is published, you can view the published artifacts using either the web UI or the http interface:

 Please change the hostname, port, and context in the URL according to your installation settings.

- Use <http://localhost:8080/soa/web/service-catalog/services/businessServiceArtifact/viewBusinessServiceArtifact?document=FTPServiceBs&collection=businessServiceArtifacts> to check the published FTP Business Service. You may follow the relationship to its implementation (FTP Web Service) and other artifacts.
- The XML representation of the business service SDM descriptor can be obtained using <http://localhost:8080/soa/systinet/platform/rest/repository/businessServiceArtifacts/FTPServiceBs?desc>
- If you are reviewing an artifact via the http interface, you may return to the UI using the link <http://localhost:8080/soa/systinet/platform/rest/repository/businessServiceArtifacts/FTPServiceBs?view>

 Note the `view` parameter.

4 Contract Demo

The purpose of this demo is to show the creation a contract between a service provider and a service consumer.

- 1 The first step is to create a business service artifact – the provider. The business service `ready for consumption` property is set to `true` to indicate that the service can be consumed. Along with the business service a service level objective (SLO) artifact is also published describing conditions under which the service can be consumed.
- 2 In the second step a consumer, Daniel Johnson, is created. He is represented by a *person* artifact. The consumer also creates a *request* for a consumption artifact that references the provider, the preferred SLO and the consumer.

3 Figure 2. Contract Overview



Finally the contract artifact is created. The consumer reviews the request for consumption and if they are satisfied, they create a contract. The contract is approved by the creation of the relationship from the artifact representing the provider (business service) to a particular revision of the contract.

You can find the demo source code in: `SOA_HOME\demos\contractmgr\simplecontract\src`

To run the contract demo:

- 1 Ensure that the demo is properly configured and SOA Systinet is running.
- 2 Change your working directory to: `SOA_HOME\demos\contractmgr\simplecontract`
- 3 To get help, execute: **run**
- 4 To build the demo, execute: **run make**
- 5 To run the first demo step – *provider*, execute: **run provider**

The output of this step resembles the following:

```
Preparing provider ...
Preparing Provider in the following steps:
1. Trying to publish Provider: Postal Address Verification and Correction Business Service
   Published!
2. The provider is trying to publish the Platinum SLO
   Published!
3. Attaching SLO to provider
   Attached!

Prepare provider summary

The following artifacts have been published:
* provider at:
  https://localhost:8843/soa/systinet/platform/restBasic/repository/businessServiceArtifacts/
PostalAddressVerificationBusiness
Service
* SLO at:

https://localhost:8843/soa/systinet/platform/restBasic/repository/sloArtifacts/PostalAddressVerificationSlo
```

- 6 To run the second demo step – *consumer*, execute: **run consumer**

The output of this step resembles the following:

```
Preparing consumer ...
Checking provider's artifacts:
  SLO found!

Preparing Consumer in the following steps:
1. Finding Person artifact representing consumer
```

```
Found: contactArtifacts/DanielConsumerJohnson
2. Trying to publish a Contract request
   Published!

Prepare consumer summary

The following artifact has been published:
* contract request at:
  https://localhost:8843/soa/systinet/platform/restBasic/repository/contractRequestArtifacts/
  DanielJohnsonContractRequest
```

7 To run the third demo step – *contract*, execute: **run contract**

The output of this step resembles the following:

```
Preparing contract ...
Checking provider's artifacts:
  SLO found!

Checking consumer's artifacts:
  Contract request found!

1. Provider is trying to publish the contract:
  Using Usage Plan: usagePlanArtifacts/PostalAddressVerificationUsagePlan?revision=1
  Using Contract request: contractRequestArtifacts/DanielJohnsonContractRequest?revision=1
  Published!
2. Provider is confirming Contract contractArtifacts/
  PostalAddressVerificationAndDanielJohnsonContract?revision=1 ...
  Contract confirmed by provider:
    contractArtifacts/PostalAddressVerificationAndDanielJohnsonContract?revision=1
```

8 To unpublish the demo, execute: **run unpublish**

9 To rebuild the demo, execute **run clean** to delete the classes directory and **run make** to rebuild the demo classes.

Once the demo is published, you may review the published artifacts using both the web UI and the http interface:



Change the hostname, port and context in the URL according to your installation settings.

- Access the Services tab at <http://localhost:8080/soa/web/service-catalog/sm/homepage>. Login as [demoapprover/changeit](#) (provider participant of the demo) or [demouser/changeit](#) (consumer participant of the demo).
- Use <https://localhost:8443/soa/web/service-catalog/services/contractArtifact/viewContractArtifact?document=PostalAddressVerificationAndDanielJohnsonContract&collection=/contractArtifacts> to review the contract artifact.
- Use <http://localhost:8080/soa/systinet/platform/rest/repository/contractArtifacts/PostalAddressVerificationAndDanielJohnsonContract?desc> to review the XML representation of the contract.

5 Reporting Demo

The purpose of this demo is to show how to create a reporting framework task and how to perform that task via the http interface.

The demo creates a Task artifact. The task specifies the use of an included selector, that every document stored in the WSDL collection should be validated against the WSDL 1.1 XML schema. The action is performed by referencing the reporting framework tool to be invoked.

The task execution is initiated by invoking the execute operation on the Task artifact using the http client. As a result, the execute call obtains the URL of the index report. The index report in turn references sub-reports (one for each document that was validated).

Polling is used in order to determine the status of the index report. Once the index report is closed and all the documents are validated the demo finishes.



Make sure that [Chapter 3, SDM Demo](#) is published.

You can find the demo source code in: `SOA_HOME\demos\reporting\task\src`

To run the reporting demo:

- 1 Make sure that the demo is properly configured and SOA Systinet is running.
- 2 Change your working directory to: `SOA_HOME\demos\reporting\task`
- 3 To get help, execute: **run**
- 4 To build the demo, execute: **run make**
- 5 To run the demo, execute: **run publish**



The output of this demo resembles the following:

```
Publishing...
Probing secure port using 'https://localhost:8843/soa/systinet/platform/restBasic/repository/'...
...secure port OK
Running demo...
Be sure that FTP Service demo is published!
Creating task...
  Published:
https://localhost:8843/soa/systinet/platform/restBasic/repository/taskArtifacts/ReportingDemoTask
Executing task asynchronously...
  Created report: https://localhost:8843/soa/systinet/platform/restBasic/repository/reportArtifacts/1

Checking report status:
  Sub-reports count: 0
  Sub-reports count: 0
  Sub-reports count: 0
  Sub-reports count: 1
  Sub-reports count: 1
  Sub-reports count: 1
Task execution finished with status: Ok
Demo successfully finished!
```

- 6 To unpublish the demo, execute: **run unpublish**
- 7 To delete the classes directory, execute: **run clean**.

Once the demo is published, you may also perform the task directly from the browser by accessing the following link:

-  Change the hostname, port, and context in the URL according to your installation settings.
- <https://localhost:port/context/systinet/platform/restBasic/repository/taskArtifacts/ReportingDemoTask?execute>
-  Log in as [demouser/changeit](#).

The task is published as the *WSDL collection change management task*.

6 Validation and Report Rendering Demo

This demo shows how to use the Policy Manager REST API to validate a resource and render the output as PDF. The demo utilizes the classes `ValidationClient` and `ReportRenderingClient` respectively. See the Javadoc for full descriptions of these classes.

Before running this demo, please run the SDM demo, described in [Chapter 3, SDM Demo](#), in order to deploy the FTP service WSDL.

In this demo, you will learn how to:

- Publish a technical policy
- Publish a business policy using this technical policy
- Use this business policy to validate a document
- View and remove report
- Render the report as a PDF file

You can find the demo source code in `SOA_HOME\demos\policymgr\validation\src`.

To run the validation demo:

- 1 Ensure that SOA Systinet is running.
- 2 Run the SDM demo as described in [Chapter 3, SDM Demo](#). Be certain to publish the FTP service from that demo.
- 3 Open a command prompt at `SOA_HOME\demos\policymgr\validation`.
- 4 Enter **run make** to compile the demo source code.

- 5 Enter **run run** to publish policies and run the validation. A link to the HTML report page is printed to the console. In addition, the report is rendered as PDF in `SOA_HOME\demos\policymgr\validation\policy_report.pdf`.

7 Java Assertion Demo

This demo shows how to create and use a custom assertion validator. You will learn how to:

- Create a custom assertion validator.
- Apply a custom assertion validator into Policy Manager as an extension.
- Create an assertion in Assertion Editor (a feature of HP SOA Systinet Workbench) based on our validator.
- Publish the assertion to the Platform repository.

You can find the demo sources in `SOA_HOME\demos\policymgr\assertionvalidator`. It contains:

- An Eclipse project for developing a custom assertion validator (in the `validator` folder).
- An Assertion Editor project for developing a sample assertion (in the `assertion` folder).
- Demo data for testing our assertion validator (in the `demodata` folder).

The demo is divided into separate procedures, described in the following sections:

Creating the Assertion Validator

A sample Eclipse project is available in `SOA_HOME\demos\policymgr\assertionvalidator\validator`. This project can be imported into Eclipse as an existing project. It has the following structure:

- `src/`
A directory containing the Java sources.
- `lib/`
A directory containing external libraries used by assertion validator.

- `resources/extension.xml`

A Policy Manager extension definition file.

- `build.xml`

An Ant build file to build extension containing the validators.

The `src` folder contains a sample implementation of an assertion validator:

`mycompany.validator.demo.ServiceNameValidator`. This validator applies to WSDL documents and checks whether the name of services defined in WSDL starts with the words "dummy," "test," "sample," or "example."

The assertion validator class must implement both methods of the interface

`org.syntinet.policy.validation.AssertionValidator`:

- `QName getDialect()` — Must return a `QName` which will be used in assertions to invoke this validator. Our example uses the `QName {http://mycompany/validation}ServiceNameValidator`.
- `void validate(ValidationListener listener, SourceCollection sources, SourceType sourceType, ValidatedAssertion[] assertions, ValidationContext context)` — This is the validation method which is called when a source must be validated against an assertion using this validator. Our sample validator parses the XML content of the WSDL document and checks each service name (under XPath `/wsdl:definitions/wsdl:service/@name`) to see whether it starts with the unwanted words or not.



For more information about Policy Manager's interfaces see the Javadoc.

To build an extension from your assertion implementation, you need an extension definition file, which is available at `resources/extension.xml`. It contains a unique identifier (attribute `uri`) which identifies the extension, the extension name (element name), and the list of contained assertion validators (elements `assertion-validator`):

```
<?xml version="1.0"?>
<extension version="1.0" uri="mycompany.ext.demo">
  <name>Demo Assertion Validators</name>
  <assertion-validator class="mycompany.validator.demo.ServiceNameValidator" />
</extension>
```

To build the extension you can use Eclipse to start an ANT build using build.xml or use the following procedure:

To build an extension for the assertion validator:

- 1 Change your working directory to `SOA_HOME\demos\policymgr\assertionvalidator`.
- 2 To get help run `run.bat` or `run.sh`.
- 3 Build the extension with the command **run.bat|sh make**.
- 4 Check the created extension in
`SOA_HOME\demos\policymgr\assertionvalidator\validator\dist\mycompany.ext.demo.jar`.

Applying the Extension

After you create `mycompany.ext.demo.jar`, apply it to SOA Systinet as an extension.

To apply the extension to SOA Systinet:

- 1 Make sure the SOA Systinet server is not running.
- 2 Copy `SOA_HOME\demos\policymgr\assertionvalidator\validator\dist\mycompany.ext.demo.jar` into `SOA_HOME\extensions`.
- 3 Execute `SOA_HOME\bin\setup` and select the **Apply Extensions** scenario.
- 4 Start SOA Systinet.

For details, see "Applying Extensions" in the *HP SOA Systinet Administration Guide*.

Creating and Deploying the Assertion

There is a sample project for Assertion Editor in `SOA_HOME\demos\policymgr\assertionvalidator\assertion`.

You can use this sample project to create your own assertions. It already contains a demo assertion named `WSDLServiceNameIsNotDummy`. Browse this assertion to see that it is applicable to WSDL Documents and implements the `ServiceNameValidator` created in [Creating the Assertion Validator on page 29](#) (linked through `SOA_HOME\demos\policymgr\assertionvalidator\validator\dist\mycompany.ext.demo.jar`).

The XML definition of the assertion is automatically filled out according to the definition in assertion validator (in the method `getDialect()`):

```
<my:ServiceNameValidator xmlns:my="http://mycompany/validation"
  xmlns:pm="http://systinet.com/2005/10/soa/policy"/>
```

After creating the assertion, deploy it. Or deploy the existing `WSDLServiceNameIsNotDummy` assertion.

To publish an assertion to SOA Systinet:

- 1 Open Assertion Editor.
- 2 Open the File menu and select **Import**→**Existing Projects into Workspace** and open the demo project.
- 3 In the Server Explorer, open the context menu and define a **New Server** pointing to your SOA Systinet server.
- 4 In the Project Explorer, open the Project context menu and select **Properties**→**SOA Systinet Server**, and select the server you defined in [Step 3](#).
- 5 Publish the demo assertion from the Project Explorer.

Open the `WSDLServiceNameIsNotDummy.asr` context menu and select **SOA Systinet Server**→**Publish Assertion**.

For more details, see the *HP SOA Systinet Assertion Editor Guide*.

Testing the Assertion Validator

In the previous sections you created and deployed an assertion validator and an assertion. The next step is to use this new assertion in a validation for a business testcase. To do this, you need to create:

- A *Technical Policy* that contains our demo assertion.
- A *Business Policy* that is applicable to *Business Services*, has a *Lifecycle Stage* set to *Production* and contains our technical policy
- Sample Business Services (one to succeed and one to fail).

The business test case is to find all business services in the Production stage that have a WSDL service name starting with Demo, Test, etc. The test data are already prepared for this demo so you do not have to create them manually. Use the following steps to import them:

Importing Demo Data to SOA Systinet

- 1 Make sure the SOA Systinet server is not running.
- 2 Run **SOA_HOME\bin\repositoryimport.bat -z SOA_HOME\demos\policymgr\assertionvalidator\demodata\demodata.zip**.
- 3 Start the SOA Systinet server.

If the demo data is successfully imported, you can start the sample validation.

In the Policies tab of the SOA Systinet UI, view the detail page for the demo business policy (Production Business Service Name Policies) and click **Validate Compliance**.

SOA Systinet displays a report that contains one business service which succeeded (OkFtpBusinessService) and one which failed (TestFtpBusinessService).