

HP SOA Systinet

Software Version: 3.10

Developer Guide

Document Release Date: January 2009
Software Release Date: January 2009



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 2003-2009 Hewlett-Packard Development Company, L.P.

Contents

| | |
|--|-----------|
| About this Guide. | 5 |
| In this Guide. | 5 |
| Document Conventions. | 7 |
| Documentation Updates. | 8 |
| Support. | 9 |
| 1 IDE Integration. | 11 |
| WSIL Report – IBM RAD and Eclipse. | 11 |
| Microsoft Visual Studio. | 12 |
| 2 REST Interface. | 15 |
| Atom-Based REST Interface. | 15 |
| Proprietary REST Interface. | 39 |
| Proprietary REST Client. | 75 |
| 3 SDM Client. | 81 |
| Basic Principles. | 82 |
| SDM Client Package. | 86 |
| 4 Technical Security. | 87 |
| SOA Systinet Overview. | 87 |
| Users and Groups. | 88 |
| Transport Security. | 90 |
| Authentication. | 91 |
| Resource ACL. | 91 |
| WEB Security. | 93 |
| Platform Services. | 93 |
| Reporting Services. | 94 |
| Policy Manager Services. | 94 |

| | | |
|---|---|-----|
| 5 | RSS | 95 |
| | Kinds of RSS Feed | 95 |
| | Syndication Syntax | 95 |
| | Subscriptions over RSS | 96 |
| 6 | Custom Source Parsers | 97 |
| 7 | Custom Validation Handlers | 101 |
| 8 | Validation Client | 103 |
| | Downloading Policies and Assertions (sync) | 103 |
| | Local Validations (validate) | 104 |
| | Validating Against Policy On Server (server-validate) | 107 |
| | Rendering Output from XML Reports (render) | 108 |

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.

In this Guide

SOA Systinet Developer Guide describes additional features and methods to enable developers to better interact with SOA Systinet.

It contains the following chapters:

- [Chapter 1, IDE Integration](#)
How to integrate SOA Systinet with IDEs.
- [Chapter 2, REST Interface](#)
A guide to the REST Interfaces.
- [Chapter 3, SDM Client](#)
Using the SDM Client.
- [Chapter 4, Technical Security](#)
A technical overview of SOA Systinet from the security point of view.
- [Chapter 5, RSS](#)

The RSS format used in SOA Systinet.

- [Chapter 6, Custom Source Parsers](#)

How to write your own source parser.

- [Chapter 7, Custom Validation Handlers](#)

How to write your own validation handler.

- [Chapter 8, Validation Client](#)

A command-line tool for policy compliance validation.

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>java -jar hpsystinet.jar</code> | User input. |
| <code>C:\System.ini</code> | File names, 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 logon page.

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

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 IDE Integration

This chapter explains how to allow IDEs to access the SOA Systinet repository.

It contains the following sections:

- [WSIL Report – IBM RAD and Eclipse on page 11](#)

How to use the WSIL query include with SOA Systinet to add it to an IDE.

- [Microsoft Visual Studio on page 12](#)

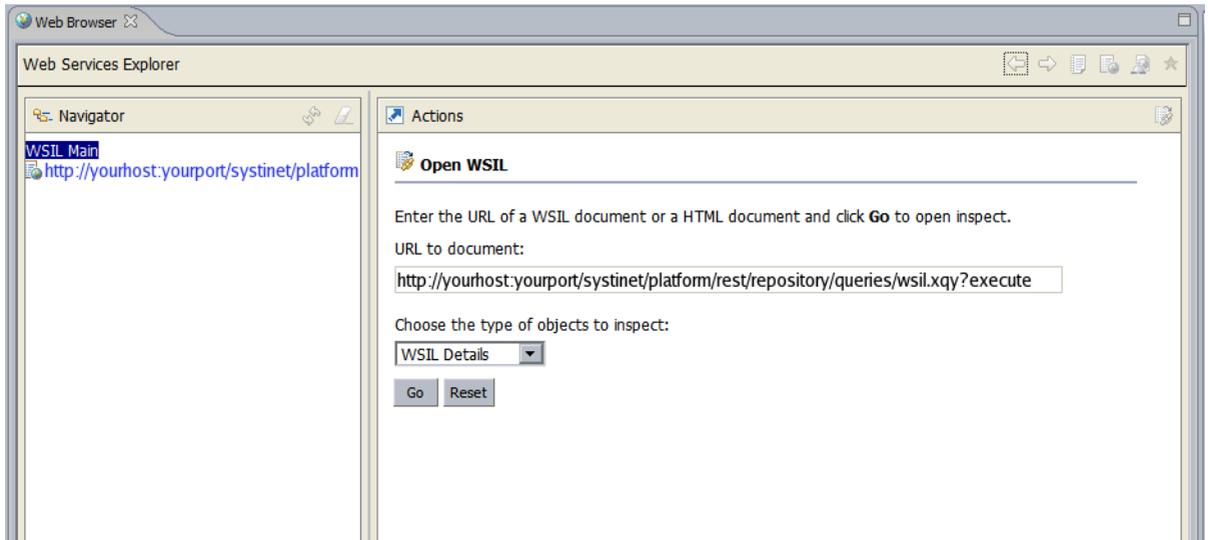
How to add SOA Systinet as a Web Reference in MS Visual Studio.

WSIL Report – IBM RAD and Eclipse

A WSIL (Web Service Inspection Language) dynamic query is included to make it easy for IDEs, like IBM RAD, to leverage the SOA Systinet repository. This query provides a list of all web services and their WSDLs and is used by RAD to create a service proxy. You can access this query from the Tools tab menu **Generate WSIL Document**, or at the referenced location

`http://yourhost:yourport/soa/systinet/platform/restBasic/service/system/wsil`

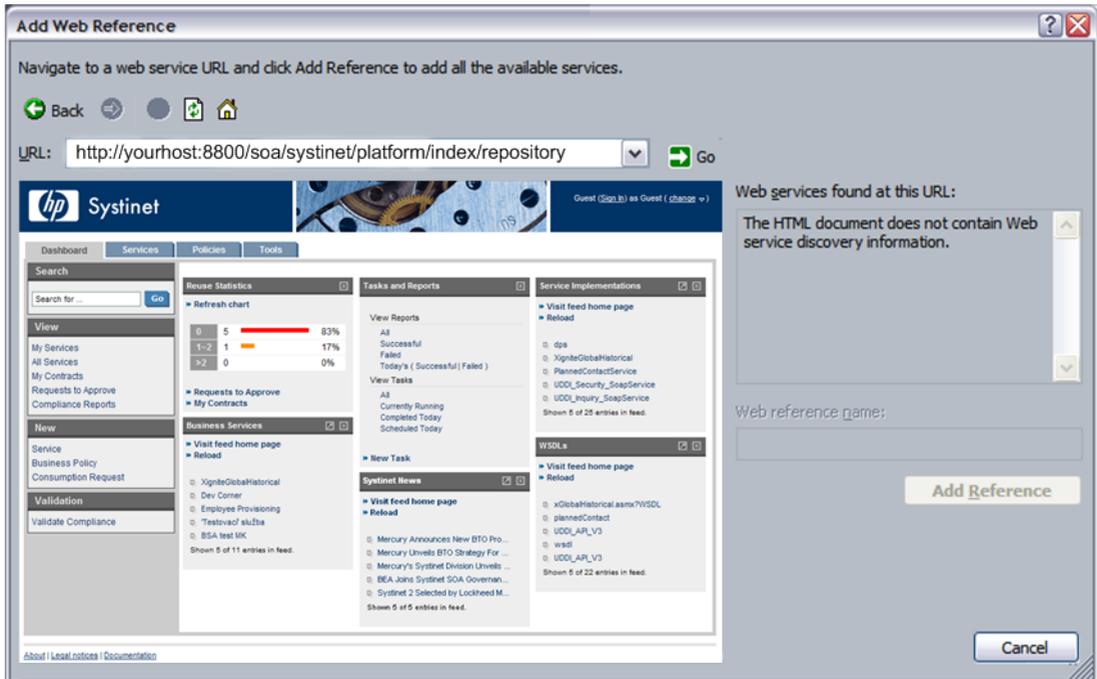
Launch IBM RAD 6.0's Web Services Explorer, and enter the WSIL report URL (the page that is generated by the WSIL link of **Search**).



From there, you will be able to access the services' WSDL documents.

Microsoft Visual Studio

The **Add Web Reference** facility of Microsoft Visual Studio's Solution Explorer is fully supported.



Enter the URL of your SOA Systinet installation (for example, `http://yourserver:8080/soa/`) to access SOA Systinet within Microsoft Visual Studio.

Notice the instructions from Microsoft Visual Studio at the top. In this case, you are navigating to a WSDL file stored in SOA Systinet. On the right, you can see that Microsoft Visual Studio does not recognize web service discovery information on the current page.

To find the service you are looking for, see the Full Text Search section in the HP Systinet User Guide.

Select the WSDL artifact for the service.

From this page you can access the WSDL document by clicking **Cached version**. At this point, Microsoft Visual Studio's Solution Explorer recognizes that the document accessed is WSDL. You can now click **Add Reference** to read the web service definition(s) into Microsoft Visual Studio.

2 REST Interface

SOA Systinet utilizes two REST interfaces in this release as an intermediate measure.

- An ATOM-based REST interface, based on *HP Symphony SDK*.



The ATOM-based interface is partially implemented in this release and utilizes only READ/GET functionality.

- A proprietary REST interface, developed by Systinet.

The interfaces are described in the following sections:

- [Atom-Based REST Interface on page 15](#)
- [Proprietary REST Interface on page 39](#)
- [Proprietary REST Client on page 75](#)

Atom-Based REST Interface

SOA Systinet uses an ATOM-based REST interface, developed using HP Symphony SDK.

For more information about Symphony SDK, see the product documentation.



In this release, the ATOM-based REST interface only uses READ functionality.

The SOA Systinet platform service document can be accessed using the following URL:

`http://hostname:port/context/platform/rest`

Hostname, port, and context are set during installation. For example, if you used the default settings and installed to your local machine, use the following URL:

```
http://localhost:8080/soa/platform/rest
```

If set up during installation, an HTTPS secure endpoint is available which requires credentials to access.

A default secure endpoint uses the following URL:

```
https://localhost:8443/soa/platform/rest
```



Use `restSecure` instead of `rest` if you are using HTTP basic authentication.

The service document consists of workspaces, which in turn contains feeds made up of entries, as shown in [Example 1 on page 17](#).

Example 1: Platform Service Document

```
<?xml version="1.0" encoding="UTF-8"?>
<app:service xml:base="http://localhost:8080/soa/platform/rest/" xmlns:app="http://www.w3.org/2007/app">
  <app:workspace>
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SDM collections</atom:title>
    <app:collection href="./artifact/reportArtifact">
      <app:accept/>
      <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Collection of Reports</atom:title>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:artifactTypes:_artifactType"/>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:reportTypes:reportType"/>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:reportCategories:reportCategory"/>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:reportStatus:reportStatus"/>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:reportResultCodes:reportResultCode"/>
      <app:categories href="./category-document/
        uddi:systinet.com:soa:model:taxonomies:associatedApplication:associatedApplication"/>
    </app:collection>
    ...
  </app:workspace>
  <app:workspace>
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Publishing Locations</atom:title>
    <app:collection href="./location">
      <app:accept/>
    </app:collection>
  </app:workspace>
  <app:workspace>
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">System Information</atom:title>
    <app:collection href="./system">
      <app:accept/>
    </app:collection>
  </app:workspace>
</app:service>
```

The interface is described in the following sections:

- [Workspaces on page 18](#)
- [Feeds on page 19](#)

- [Entries on page 27](#)
- [Category Documents on page 38](#)

Workspaces

The platform service document consists of the following workspaces:

- [SDM Collections Workspace on page 18](#)

The SDM workspace reflects the structure of the SOA Definition Model (SDM) and defines feeds for the collections in the SOA Systinet repository.

- [Publishing Locations Workspace on page 19](#)

The locations workspace reflects the structure of attached data content in SOA Systinet created by the publisher.

- [System Collections Workspace on page 19](#)

The system workspace contains system information used by SOA Systinet.

SDM Collections Workspace

The SDM collections workspace contains a collection for each artifact type in the SOA Definition Model (SDM) for which an instance can be created within its artifact hierarchy.



HP SOA Systinet Customization Editor can be used to modify the SDM, so your configuration may vary from specific examples in this documentation.

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

Each collection in the workspace consists of the following:

- `<app:collection href="/artifact/artifactType">`

The reference defines the URL used for the feed for that particular artifact type collection.

For details, see [Artifact Collection Feeds on page 20](#).

- `<app:categories href="/category-documents/taxonomy">`

Categories can occur in feed entries and some feed readers can perform filtering according to these categories.

Publishing Locations Workspace

The publishing locations workspace consists of a single collection. This collection is an atom feed made up of entries where the entry can be one of the following types:

- Subcollection
- Resource

The subcollections and resources reflect content uploaded to SOA Systinet using its publication feature.

For more details, see "Publishing Service Infrastructure from Definition Documents" in the *HP SOA Systinet User Guide*.

This location is available as a feed.

For details, see [Publishing Location Feeds on page 25](#).

System Collections Workspace

The system collections workspace contains a single collection. This collection contains information about the running system.

Feeds

You can access the content of the repository using feeds.

- [Artifact Collection Feeds on page 20](#)
- [Publishing Location Feeds on page 25](#)
- [Artifact History Feed on page 27](#)

Artifact Collection Feeds

Every artifact type collection in the SDM is accessible as a feed.

Use the reference defined in the SDM collections workspace to access a collection feed.

For example, the WSDL collection feed is accessed with URL:

```
http://localhost:port/context/platform/rest/artifact/wsdArtifact
```

Example 2: WSDL Collection Feed

```
<feed xml:base="http://localhost:8080/soa/platform/rest/artifact/wsdArtifact"
xmlns="http://www.w3.org/2005/Atom">
  <id>hp-soa:atom:feed:sdm:wsdlArtifact</id>
  <updated>2008-04-29T13:59:47.240+02:00</updated>
  <title type="text" xml:lang="en">Collection of WSDLs</title>
  <link href="artifactBase" type="application/atom+xml;type=feed"
    rel="related-super" title="parent sdm feed"/>
  <link href="wsdlArtifact?start-index=1&page-size=50" type="application/atom+xml;type=feed" rel="self"
    title="feed self"/>
  <generator>HP SOA Systinet</generator>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>50</opensearch:itemsPerPage>
  <entry>
    <id>f726b881-9763-4a9f-8977-1992dd9e2b95</id>
    <updated>2008-04-29T12:10:22.981+02:00</updated>
    <title type="text" xml:lang="en">account_portType.wsdl</title>
    ..
  </entry>
  ..
</feed>
```

Each artifact type collection feed consists of the following descriptors:

| Descriptors | Description |
|-------------|--------------------------|
| id | The feed identification. |
| updated | The last update time. |
| title | The name of the feed. |

| Descriptors | Description |
|-------------------------|--|
| link | <p>A set of links with the following link types indicated by the <code>rel</code> attribute:</p> <ul style="list-style-type: none"> related-super <p>Links to collection feeds for super artifacts in the inheritance category.</p> <ul style="list-style-type: none"> related-sub <p>Links to collection feeds for descendant artifact types.</p> |
| entry | The set of entries in the feed. For more details, see Artifact Atom Entries on page 28 . |
| opensearch:startIndex | Starting point for the feed relative to index entries. The first indexed item is 1. |
| opensearch:itemsPerPage | Number of items per page. |

You can modify the output of the feed as described in the following sections:

- [Filtering Feeds on page 22](#)
- [Viewing Entry Content in Feeds on page 22](#)
- [Property Based Searching on page 23](#)
- [Feed Ordering on page 24](#)
- [Feed Paging on page 25](#)

You can also combine these output methods.

Separate each term with `&`.

For example, to get artifacts 10-79 which contain `policy` in the description, ordered primarily by their name in descending order and then by description in ascending order, and displaying properties defined in `artifactBase`, use the following URL:

```
http://host:port/context/platform/rest/artifact/artifactBase?p.description=policy&start-index=10&page-size=70&order-by=name-,description&inline-content
```

Filtering Feeds

Feeds are presented in the REST interface as a set of equivalent collections.

Examples of feeds include:

- `http://localhost:port/context/platform/rest/artifact/implementationArtifact`
- `http://localhost:port/context/platform/rest/artifact/xmlServiceArtifact`
- `http://localhost:port/context/platform/rest/artifact/webServiceArtifact`
- `http://localhost:port/context/platform/rest/artifact/businessServiceArtifact`
- `http://localhost:port/context/platform/rest/artifact/wsdlArtifact`

Viewed in this way, the feeds form a flat structure. However, there are established relationships between feeds in terms of an inheritance hierarchy.

The root of the hierarchy is `http://localhost:port/context/platform/rest/artifact/artifactBase`.

You can use abstract artifact type feeds to obtain all artifact types lower in the hierarchy. For example, the `implementationArtifact` feed contains all SOAP service, XML service, and web application artifacts.

The relationships between feeds are realized via `related-sub` and `related-super` links.

Viewing Entry Content in Feeds

You can use feeds to obtain multiple artifact entry content as well.

Add `?inline-content` to the collection feed URL to obtain the full content for each entry in the feed.



The properties displayed in the content for an entry are determined by the artifact type used in the feed URL. Properties specific to an artifact type lower in the hierarchy are not displayed. However relational properties are always displayed regardless the artifact type used in the feed URL.

Property Based Searching

You can search for specific artifacts in a feed with property based filtering. You can filter by any property type regardless of its type and cardinality, but the elementary conditions are always primitive values. The filtering property must be present in the artifact type defining the feed.

The property must be one of the following elementary types:

- text
- integer
- bigInteger
- date
- double
- boolean
- uuid

To view the permitted property names for a particular artifact feed, you can examine the SDM with URL:

```
http://host:port/context/platform/rest/system/model.
```

If you want to filter by a compound property (for example, category property which has 3 compounds: taxonomyUri, name, value) you must use dot notation. For example to search by compound `val` (value) of property `criticality` on `businessServiceArtifact` use the following URL:

```
http://host:port/soa/platform/rest/artifact/businessServiceArtifact?p.criticality.val=uddi:systinet.com:soa:model:taxonomies:impactLevel:high
```

Only business services artifacts with high criticality are listed.

For text property filtering, operator `case-insensitive-equals` is used, but can explicitly use wildcards. To find all service artifact with `svc` in their name submit the following URL:

```
http://host:port/soa/platform/rest/artifact/businessServiceArtifact?p.name=*svc*
```

The following wildcards are supported:

- * for zero or more arbitrary characters.
- _ for exactly one arbitrary character.



SOA Systinet does not support explicit boolean operators but there is an implicit AND for conditions on different properties and an implicit OR on conditions on the same property.

The following examples show various ways to use property searching:

- Artifacts with a name starting with `service` and a description containing `assertion`:

```
http://host:port/context/platform/rest/artifact/artifactBase?p.name=service*&p.description=*assertion*
```

- Artifacts with a name containing either starting with `service` or containing `assertion`:

```
http://host:port/context/platform/rest/artifact/artifactBase?p.name=service*&p.name=*assertion*
```

- Deleted artifacts only.

```
http://host:port/context/platform/rest/artifact/artifactBase?p._deleted=true
```

Feed Ordering

By default, entries in feeds are ordered by their `atom:updated` element.

Add `?order-by=` to the collection feed URL to change the order.

- Entries ordered by name (ascending):

```
http://host:port/context/platform/rest/artifact/artifactBase?order-by=name
```

- Entries ordered by name (descending):

```
http://host:port/context/platform/rest/artifact/artifactBase?order-by=name-
```

- Entries ordered by name (descending), then description (ascending):

```
http://host:port/context/platform/rest/artifact/artifactBase?order-by=name-,description
```

You can also use properties for ordering with the same conditions as for searching.

For details, see [Property Based Searching on page 23](#).

Feed Paging

You can also control the feed paging.

- The first ten entries:

```
http://host:port/context/platform/rest/artifact/artifactBase?page-size=10
```

- Entries 10-19 (inclusive):

```
http://host:port/context/platform/rest/artifact/artifactBase?page-size=10&start-index=10
```



The default number of entries is 50 and the maximum number of entries is 500.

Publishing Location Feeds

The location feed enables you to browse the attached data content in the repository.

SOA Systinet adds this content whenever you publish an artifact associated with attached data content.

For details, see "Publishing Service Infrastructure from Definition Documents" in the *HP SOA Systinet User Guide*.

The content consists of resources (the data content) organized into collections (folders).

You can access the feed using the following URL:

```
http://localhost:8080/soa/platform/rest/location
```

If you use a browser, open a view which enables you to examine the data content and interact with it.



The view of a collection location only displays resources that you have permissions for.

SOA Systinet publisher creates a collection within the publishing location when you upload data content.

For more details, see "Publishing Service Infrastructure from Definition Documents" in the HP SOA Systinet User Guide.

Open a collection by clicking its name, or download a zip file of its content by clicking **Download as Archive**.

At the lowest level, the browser shows the actual data content.

For the actual content, click the content name.

Click **Advanced View** to open the detail view of the related artifact in SOA Systinet.

For details, see "Artifact Detail Pages" in the *HP SOA Systinet User Guide*.

You can change the output of the location space on your browser using alternative media types:

- `http://hostname:port/context/platform/rest/location`

The default output as described above.

- `http://hostname:port/context/platform/rest/location?alt=text/html`

The HTML representation which is the default output.

- `http://hostname:port/context/platform/rest/location/foo?alt=application/zip`

Output all files from a particular collection (foo) to a zip archive.

- `http://hostname:port/context/platform/rest/location/test?alt=application/atom%2bxml`

View the Atom feed for a collection location.

- `http://hostname:port/context/platform/rest/location/foo?alt=text/javascript`

Output a particular collection location as a JSON representation.

By default, the last revision of a resource or collection is shown, but you can request revisions from a particular date using the following pattern:

`http://hostname:port/context/platform/rest/location;datetime=[datetimeValue]`

For example, `http://hostname:port/context/platform/rest/location/foo/a.wsdl`, corresponds to the last revision of the `a.wsdl` resource in the `foo` location.

`http://hostname:port/context/platform/rest/location;datetime=2008-01-01T12:00:00.000Z/foo/a.wsdl`, corresponds to the revision of the `a.wsdl` resource at 12:00 on 1/1/2008.



If a particular collection is reused and the older resources overwritten, then the older revisions of the original content may not be available.

By default, specifying a collection location that does not exist returns an exception.

Use the `noexc` parameter to prevent the exception and return the location as an empty collection.

For example, `http://hostname:port/xcontext/platform/rest/location/foo?noexc`

Artifact History Feed

You can view the revision history of an artifact as a feed.

For example, to view the revision history of `my.wsdl`, use the URL:

`http://host:port/context/platform/rest/artifact/wsdlArtifact/my.wsdl/history`

Entries

The detailed information about an artifact in the repository is available as an entry.

Entries are described in the following sections:

- [Artifact Atom Entries on page 28](#)
- [Artifact History Entries on page 31](#)
- [Atom Entry Property Descriptors on page 31](#)
- [Artifact Data on page 37](#)

- [Resource Identification on page 38](#)

Artifact Atom Entries

The information about each entry in the collection feed is only a summary. Each entry can be accessed directly using its `self link` as referenced in the artifact feed, which is formed from either its `restName` or `id`.

For example, you can access a particular user profile entry with URL:

```
http://localhost:port/context/platform/rest/artifact/personArtifact/admin
```

Example 3: Admin User Profile Entry

```
<entry xml:base="http://localhost:8080/soa/platform/rest/artifact/personArtifact"
  xmlns="http://www.w3.org/2005/Atom">
  <id>47526745-d855-44a4-afa0-bb1fddcf6f37</id>
  <updated>2008-05-22T16:41:42.623+02:00</updated>
  <title type="text" xml:lang="en">admin</title>
  <link href="personArtifact/101?alt=application%2Fatom%2Bxml" type="application/atom+xml"
    rel="self" title="artifact detail"/>
  <link href="personArtifact/101?alt=application%2Fxml" type="application/xml" rel="alternate"
    title="XML representation"/>
  <link href="personArtifact/101?alt=application%2Fatom%2Bxml" type="application/atom+xml"
    rel="last-revision" title="last revision"/>
  <link href="personArtifact" type="application/atom+xml;type=feed" rel="collection"
    title="sdm feed"/>
  <link href="personArtifact/101/history" type="application/atom+xml;type=feed" rel="history"
    title="history feed"/>
  <link href="personArtifact/101?alt=text%2Fhtml" type="text/html" rel="alternate"
    title="UI view page"/>
  <category label="S1" scheme="uddi:systinet.com:soa:model:taxonomies:accountStates:accountState"
    term="S1"/>
  <category label="Artifact"
    scheme="uddi:systinet.com:soa:model:taxonomies:artifactTypes:_artifactType"
    term="urn:com:systinet:soa:model:artifacts"/>
  <category label="Content"
    scheme="uddi:systinet.com:soa:model:taxonomies:artifactTypes:_artifactType"
    term="urn:com:systinet:soa:model:artifacts:content"/>
  <category label="Contact"
    scheme="uddi:systinet.com:soa:model:taxonomies:artifactTypes:_artifactType"
    term="urn:com:systinet:soa:model:artifacts:content:contact"/>
  <category label="User Profile"
    scheme="uddi:systinet.com:soa:model:taxonomies:artifactTypes:_artifactType"
    term="urn:com:systinet:soa:model:artifacts:content:contact:person"/>
  <author>
    <name>admin</name>
  </author>
  <content type="application/xml">
    ...
  </content>
  <summary type="text" xml:lang="en">Automatically created user account</summary>
</entry>
```

Each artifact entry consists of the following descriptors:

| Descriptor | Description | |
|------------|--|---|
| id | A unique id for the artifact (uuid). | |
| updated | The last update time. | |
| title | The name of the entry. | |
| link | A set of links with the following link types indicated by the <code>rel</code> attribute: | |
| | self | The atom entry details. |
| | collection | The associated artifact collection feed. For details, see Artifact Collection Feeds on page 20 . |
| | last-revision | The last revision of this artifact. |
| | edit-media | The associated data content for an artifact. |
| | history | The collection feed for revisions of this artifact. |
| | alternate | A set of alternate views of the artifact, including: <ul style="list-style-type: none"> • <code>application/xml</code> The bare XML representation of the <code>content</code> descriptor. • <code>text/html</code> Points to the SOA Systinet Services UI view of the artifact or the Tools UI view as a fallback. |
| related | Links to related artifacts.  Related artifacts may also be linked where the link has the <code>rel</code> attribute with a specific relationship name. For details, see Relationship Properties Atom Representation on page 35 . | |

| Descriptor | Description |
|-------------------|---|
| category | A set of taxonomic values from: <ul style="list-style-type: none"> • Taxonomy property values • categoryBag and identifierBag • sdmTypes taxonomy values |
| author | The creator of this revision of the artifact. |
| content | The bare XML representation of the content descriptor. For details, see Atom Entry Property Descriptors on page 31 . |
| summary | An artifact description. |

Artifact History Entries

By default, entries display the latest revision. You can view older revisions by adding `;rev=X` to the entry URL.

For example, the first revision of a WSDL can be obtained with the URL:

```
https://host:port/context/platform/rest/artifact/wsdlArtifacts/mywsdl;rev=1
```

Atom Entry Property Descriptors

Atom entries contains an XML representation of an artifact in the `content` descriptor.

Example 4: Admin User Entry Content

```
<content type="application/xml">
  <art:artifact name="personArtifact" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:p="http://hp.com/2008/02/systinet/platform/model/property"
    xmlns:sdm="http://hp.com/2007/10/systinet/platform/model/propertyType"
    xmlns:art="http://hp.com/2008/02/systinet/platform/model/artifact">
    <p:loginName sdm:type="text">admin</p:loginName>
    <p:address xsi:nil="true" sdm:type="address"/>
    <p:email sdm:type="text" p:multi="true">admin@example.com</p:email>
    <p:phone xsi:nil="true" sdm:type="text" p:multi="true"/>
    <p:instantMessenger xsi:nil="true" sdm:type="text" p:multi="true"/>
    <p:primaryGroup xsi:nil="true" sdm:type="text"/>
    <p:accountState name="S1" taxonomyUri="uddi:systinet.com:soa:model:taxonomies:accountStates"
      value="S1" sdm:type="category"/>
    <p:designTimePolicy xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:documentation xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:_ownerLowerCase sdm:type="text">admin</p:_ownerLowerCase>
    <p:_uuid sdm:type="uuid">47526745-d855-44a4-afa0-bblfddcf6f37</p:_uuid>
    <p:_revision sdm:type="integer">1</p:_revision>
    <p:_checksum sdm:type="bigInteger">0</p:_checksum>
    <p:_contentType sdm:type="text">application/xml</p:_contentType>
    <p:_revisionTimestamp sdm:type="date">2008-05-22T14:41:41.013Z</p:_revisionTimestamp>
    <p:categoryBag xsi:nil="true" sdm:type="categoryBag"/>
    <p:_revisionCreator sdm:type="text">admin</p:_revisionCreator>
    <p:_artifactType name="Artifact"
      taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
      value="urn:com:systinet:soa:model:artifacts" sdm:type="category" p:multi="true"/>
    <p:_artifactType name="Content"
      taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
      value="urn:com:systinet:soa:model:artifacts:content" sdm:type="category" p:multi="true"/>
    <p:_artifactType name="Contact"
      taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
      value="urn:com:systinet:soa:model:artifacts:content:contact" sdm:type="category"
      p:multi="true"/>
    <p:_artifactType name="User Profile"
      taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
      value="urn:com:systinet:soa:model:artifacts:content:contact:person" sdm:type="category"
      p:multi="true"/>
    <p:identifierBag xsi:nil="true" sdm:type="identifierBag"/>
    <p:description sdm:type="text">Automatically created user account</p:description>
    <p:_owner sdm:type="text">admin</p:_owner>
    <p:_deleted sdm:type="boolean">false</p:_deleted>
    <p:name sdm:type="text">admin</p:name>
  </art:artifact>
</content>
```

```

    <p:consumptionContract xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:consumptionRequest xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:provides xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:contactRole xsi:nil="true" sdm:type="category"/>
    <p:geographicalLocation xsi:nil="true" sdm:type="category" p:multi="true"/>
    <p:languageCode xsi:nil="true" sdm:type="category"/>
    <p:hpsoaApplicationContact xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
    <p:externalDefinition xsi:nil="true" sdm:type="documentRelationship" p:multi="true"/>
  </art:artifact>
</content>

```

The content is effectively a list of the properties of an artifact.

The property types are described in the following sections:

- [Primitive Properties Atom Representation on page 33](#)
- [Taxonomy Properties Atom Representation on page 34](#)
- [Relationship Properties Atom Representation on page 35](#)
- [Special Properties Atom Representation on page 36](#)

Primitive Properties Atom Representation

Primitive properties are represented as follows:

```
<p:NAME sdm:type="TYPE">VALUE<p:NAME>
```

The following primitive property types use this form:

| Property Type | xsi:type Correspondance |
|---------------|-------------------------|
| date | xs:dateTime |
| boolean | xs:boolean |
| double | xs:double |
| integer | xs:int |
| bigInteger | xs:integer |
| text | xs:string |

| Property Type | xsi:type Correspondance |
|---------------|-------------------------|
| uuid | xs:string |

For example:

```
<p:phone sdm:type="text">774 789 784</p:phone>
```

Taxonomy Properties Atom Representation

Taxonomy properties are propagated in two places in the Atom entries.

The `category` descriptor, which also appears in collection feeds, describes the taxonomy and category as follows:

```
<category label="..." scheme="..." term="..." />
```

- `label` corresponds to the category name.
- `scheme` corresponds to the taxonomy URI combined with the property name.
- `term` corresponds to the category URI.

This is reproduced in the entry `content` as a property:

```
<p:NAME name="..." taxonomyUri="..." value="..." sdm:type="category" />
```

For example, a web service with Failure Impact set to High is represented as a property in the entry for the web service:

```
<p:criticality name="High" taxonomyUri="uddi:systinet.com:soa:model:taxonomies:impactLevel"
value="uddi:systinet.com:soa:model:taxonomies:impactLevel:high" sdm:type="category" />
```

Note that the property representing this taxonomic category is `criticality`.

The property is propagated to Atom metadata as an `atom:category` element:

```
<atom:category label="High" scheme="uddi:systinet.com:soa:model:taxonomies:impactLevel:criticality"
term="uddi:systinet.com:soa:model:taxonomies:impactLevel:high" / >
```

Relationship Properties Atom Representation

Relationship properties are propagated in two places in the Atom entry.

In feeds the `link` exists as metadata.

The `link` descriptor describes the following link types:

- A generic `related` link.
- A specific relationship bound link where the `rel` attribute uses a `hp-systinet-` prefix with the relationship name.

In entries, relationships are described as a set of `property` atom content descriptors:

Example 5: Relationship Properties

Incoming relationship example:

```
<p:inBusinessService xlink:href="businessServiceArtifact/1210"
  sdm:type="documentRelationship" p:multi="true">
  <t:source>c519d961-03b3-4303-b61b-8809b945b7ae</t:source>
  <t:exact>>false</t:exact>
</p:inBusinessService>
```

Exact incoming:

```
<p:inBusinessService xlin:href="businessServiceArtifact/1210"
  sdm:type="documentRelationship" p:multi="true">
  <t:source>c519d961-03b3-4303-b61b-8809b945b7ae</t:source>
  <t:exact>>true</t:exact>
</p:inBusinessService>
```

Outgoing relationship example:

```
<p:service xlin:href="webServiceArtifact/5"
  sdm:type="documentRelationship" p:multi="true">
  <t:target
    deleted="false">5a4aeca7-a8f9-4761-b504-82723ab2f417</t:target>
</p:service>
```

Exact outgoing:

```
<p:service xlin:href="xmlServiceArtifact/101.xml;rev=1"
  sdm:type="documentRelationship" p:multi="true">
  <t:target revision="1"
    deleted="false">72ab6f1f-e943-4fd2-a7bc-5d227e6e134a</t:target>
</p:service>
```

Special Properties Atom Representation

Special properties are defined by an XML schema which determines their structure.

SOA Systinet contains an XML schema which defines the following property types:

- address
- categoryBag
- identifierBag

- dailyInterval
- nameURLPair
- nameValuePair
- parameterList (XQuery parameter)
- scheduled
- selector

Artifact Data

If an artifact has associated data content, then you can directly access the data content.

For example, a WSDL artifact is usually associated with the actual WSDL file.

Access the WSDL entry with the URL:

<https://localhost:8443/soa/platform/rest/artifact/wsdlArtifact/mywsdl?alt=application%2Fatom%2Bxml>

Example 6: WSDL Entry

```
<entry xml:base="https://localhost:8443/soa/platform/restSecure/artifact/wsdlArtifact"
xmlns="http://www.w3.org/2005/Atom">
  <id>c686605d-6149-48f8-a7c7-d593999735eb</id>
  <updated>2008-06-12T17:45:17.808+02:00</updated>
  <title type="text" xml:lang="en">mywsdl</title>
  ...
  <link href="../../../location/wsdl/google/mywsdl.wsdl" type="text/xml" rel="edit-media" title="attached
data"/>
  ...
</entry>
```

The entry contains a link pointing to the locations workspace. The data is also available using a `/data` suffix.

For example, <https://localhost:8443/soa/platform/rest/artifact/wsdlArtifact/mywsdl/data>

You can also access older revisions of the data with the URL:

<https://localhost:8443/soa/platform/rest/artifact/wsdlArtifact/mywsdl;rev=1/data>



Using any relative references in the XML data will probably cause an error because they are resolved relatively to the GET context. Use the `location` context to navigate references instead.

Resource Identification

A web service artifact with `uuid` `65a2b119-9a6b-491e-8353-3692f4b9e3e5` and name `MyService` is available in the artifacts collection:

<http://localhost:port/context/soa/platform/rest/artifact/>

At the following locations:

- `artifactBase/65a2b119-9a6b-491e-8353-3692f4b9e3e5`
- `implementation/65a2b119-9a6b-491e-8353-3692f4b9e3e5`
- `webServiceArtifact/65a2b119-9a6b-491e-8353-3692f4b9e3e5`

These URLs are not user-friendly. For newly created artifacts, SOA Systinet auto-generates a REST name which in most cases is more user-friendly than the `uuid`.

This REST name can be used instead of the `uuid` in the URL.

<http://localhost:port/context/soa/platform/rest/artifact/webServiceArtifact/MyService>



If you migrate or federate resources (for example, with UDDI Registry import/export), the user-friendly URLs are lost.

User-friendly REST names remain the same, even if you change the artifact name.

Category Documents

Atom categories are a way to categorize large amounts of data. The permitted values in Atom categories can be either fixed or unrestricted. Category documents group permitted category values.

An example of a category group with a fixed set of values is the impact level criticality category group.

```
http://host:port/context/platform/rest/category-  
document/uddi:systinet.com:soa:model:taxonomies:impactLevel:criticality
```

Example 7: Impact Criticality Category Document

```
<?xml version="1.0" encoding="UTF-8"?>  
<app:categories xmlns:app="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"  
  xmlns:hp="http://hp.com/2008/02/systinet/platform/model/taxonomy"  
  xmlns:v355tax="http://systinet.com/uddi/taxonomy/v3/5.5"  
  xmlns:v350tax="http://systinet.com/uddi/taxonomy/v3/5.0" fixed="yes"  
  scheme="uddi:systinet.com:soa:model:taxonomies:impactLevel:criticality">  
  <atom:category term="uddi:systinet.com:soa:model:taxonomies:impactLevel:high" label="High"/>  
  <atom:category term="uddi:systinet.com:soa:model:taxonomies:impactLevel:medium" label="Medium"/>  
  <atom:category term="uddi:systinet.com:soa:model:taxonomies:impactLevel:low" label="Low"/>  
</app:categories>
```

SOA Systinet uses taxonomies, which are an abstraction almost identical to Atom categories. These taxonomies are sometimes transferable to Atom category documents, which can be referenced from the service document.

The categories in the taxonomy then appear as Atom categories, corresponding to the taxonomy values in artifact entries and feeds.

Proprietary REST Interface

The types of resource that the REST interface provides access to are described in the Repository Element Formats section in the HP Systinet Reference Guide. That is, a hierarchy of collections each containing:

- Collections
- Documents, of type:
 - XML
 - Binary

This chapter contains the following sections:

- [REST Interface URIs on page 40](#)

The set of HTTP URL paths identifying resources

- [Resource Representations on page 46](#)

The REST view of resources

- [REST Operations on page 52](#)

REST operation mapping to HTTP methods and URL query parameters

- [REST Exceptions on page 69](#)

HTTP response formats

- [Executable Objects on page 72](#)

How REST handles reports and queries

REST Interface URIs

As the interface uses HTTP, the URIs used to access it are HTTP URLs. [Table 1](#) explains how different parts of a URL are used for different purposes, including:

- Specifying a WS endpoint provided by the REST interface.
- Identifying an existing resource.
- Specifying details of an operation, in conjunction with the HTTP method.
- Specifying how a resource is represented, in conjunction with the HTTP `Content-type` header.

A REST interface URL has this form:

```
protocol://hostname:port/soa/systinet/platform/interface/namespace/{collection/*document?
```

Table 1. Components of a REST Interface HTTP URL

| Component | Description |
|------------------------|--|
| protocol | HTTP, or HTTPS if using SSL. |
| hostname | The host where the JEE server running SOA Systinet is installed. It can also be the host where the loadbalancer or proxy runs. |
| port | <p>The port depends on the configuration of the JEE server. The default installation on JBoss uses:</p> <ul style="list-style-type: none">• 8080 if the protocol is http• 8843 if the protocol is https <p>The port must be specified unless it is the default configured for the protocol in a browser.</p> |
| soa/systinet/platform/ | SOA Systinet context path. <code>soa/</code> is the context of the application in the JEE container. |
| interface | <p>This completes the service path. For REST it is one of the following:</p> <ul style="list-style-type: none">• <code>rest</code> for anonymous access or internal request using internal SSO authentication• <code>restBasic</code> for access using HTTP Basic authentication |
| namespace | <p>This field allows more than one database storage. It can also be used as a namespace for non-persistent objects. For SOA Systinet, two values are legal:</p> <ul style="list-style-type: none">• <code>repository</code> is the database storage for collections and documents• <code>service</code> is a namespace where functional resource reside |
| collection | The remainder of the path identifies a resource in the repository. The first / identifies the root collection. Each subsequent /, separates a collection from a resource it contains. Zero or more additional collections can be specified in this way. |

| Component | Description |
|--------------|--|
| document | <p>A single document resource name may be present to complete the identification of the resource. If none are present, then the resource is the collection specified by the preceding fields.</p> |
| query string | <p>An HTTP URL query string may encode additional options and parameters necessary to fully specify a REST operation:</p> <ul style="list-style-type: none"> • The choice of REST operation is encoded in an HTTP request using the HTTP method and/or a query string field. For details, see REST Operations on page 52. • Support for the creation of resources requires that the new resource is specified as a query parameter relative to an existing collection (specified in the URL path). For details, see REST Operations on page 52. • Parameters of the representation type may appear in the query string. For details, see Resource Representations on page 46 <p>In the following, <i>query parameter</i> is used to refer to fields of the form <code>name=value</code>. Boolean options are specified by the appearance or absence of a field containing only an option identifier, and are referred to as <i>query fields</i>.</p> <p>A complete list of query parameters is listed in Table 2.</p> |

Table 2. REST URL Query Parameters

| Category | Query parameter | Operation | Value | Description |
|-------------------------|-----------------|-----------|--------------|---|
| operation | create | N/A | no value | Denotes the CREATE operation on HTTP POST |
| operation | update | N/A | no value | Denotes the UPDATE operation on HTTP POST |
| operation | delete | N/A | no value | Denotes the DELETE operation on HTTP POST |
| operation | undelete | N/A | no value | Denotes the UNDELETE operation on HTTP POST |
| operation | purge | N/A | no value | Denotes the PURGE operation on HTTP POST |
| operation | get | N/A | no value | Denotes the GET operation on HTTP POST. |
| operation | exist | N/A | no value | Denotes the EXIST operation on HTTP GET. |
| resource identification | resource | C | string | Specifies the name and type of resource to create. If the name ends with "/", then a collection is created. Otherwise, a document is created. If not given, then a document is created with a generated name. |
| resource identification | resource | G,U,D,UnP | string | Identifies the name of a subresource in the collection given by request URI |
| resource identification | revision | G | integer, >=0 | Identifies the resource revision to GET |
| resource identification | revision | U,D,P | integer, >=0 | Identifies the resource revision to UPDATE/DELETE/PURGE. If the given revision is not the last resource revision, then the UPDATE/DELETE/PURGE operation fails. |

| Category | Query parameter | Operation | Value | Description |
|-------------------------|-----------------|-----------|---|---|
| resource identification | datetime | G | datetime in ISO8601 based format (e.g., 2006-07-11T11:28:51.348Z) | Datetime identifying the resource revision to GET (timeslice). |
| resource identification | datetime | U,D,P | datetime in ISO8601 based format (e.g., 2006-07-11T11:28:51.348Z) | Datetime identifying the resource revision to UPDATE/DELETE/PURGE. If the given revision is not the last resource revision, then the UPDATE/DELETE/PURGE operation fails (timeslice). |
| resource identification | original | G | no value | Returns unresolved data, such as when a WSDL is imported and contains unresolved imports. |
| resource identification | deleted | G | no value | Returns only deleted subresources in a collection listing. If not given, then only live subresources are listed. |
| request data param | contentType | C | string | Sets the content-type metadatum of a created resource. If not given, then the HTTP header Content-Type header is used (for RAW request message) or content-type metadatum in passed metadata (for REST resource serialization). |
| representation | meta | G | no value | If given, then the REST resource serialization is returned with a set metadata section. |
| representation | desc | G | no value | If given, then the REST resource serialization is returned with a set descriptor section. |
| representation | desc | C,U | no value | If given, then the REST response to a CREATE/UPDATE operation will contain the descriptor section of the created/updated resource. |

| Category | Query parameter | Operation | Value | Description |
|----------------------|-----------------|-----------|-----------------|--|
| representation | data | G | no value | If given, then the REST resource serialization is returned with a set data section. |
| representation | acl | G | no value | If given, then the REST resource serialization is returned with a set acl section. |
| representation | acl | C,U | no value | If given, then the REST response to CREATE/UPDATE operation will contain the ACL section of created/updated resource. |
| representation | history | G | no value | If given, then the REST resource serialization is returned with the data section filled with all resource revisions. |
| representation | rss | G | optional string | If given, then the RSS of the resource is returned. An optional value specifies the RSS format (for example <code>atom_0.3</code> ; for the full list of values, see Chapter 5, RSS). |
| representation | view | G | no value | If given, then the HTTP redirect (302) to URL of the resource UI representation (for browsers). |
| execution | execute | G | no value | Executes the resource. It is bound to HTTP GET. |
| execution | ex_ | G | string | Prefix for execute parameters. A functional resource can accept parameters without this prefix, however the prefix can be used for backward compatibility. |
| execution | fulltext | G | string | Fulltext search query string. |
| processing parameter | chp_ | G,U,D,UhP | string | Prefix for a collection handler parameter (for example, 'chp_foo=bbb' for parameter 'foo'). |

| Category | Query parameter | Operation | Value | Description |
|-------------------------|-----------------|------------------------|---------------------|--|
| response data parameter | style | G,U,D,U _n P | URI | URI of an XSLT stylesheet for a response transformation. The URI can be absolute or relative. A relative URI is resolved to a repository base URL. |
| response data parameter | st_ | G,U,D,U _n P | string | Prefix for a stylesheet parameter (for example, 'st_foo=bbb' for parameter 'foo'). |
| response data parameter | export | G,U,D,U _n P | optional string | Overrides HTTP header Content-Type. If no value is given, then application/octet-stream is used. |
| search | cs_ | G | string | Prefix for parameter that limits result of collection listing to artifact matching the condition. |
| search | co_ | G | (integer,)?asc desc | Prefix for parameter that specifies order of artifact in collection listing. |
| paging | page | G | positive integer | If specified, only the resource on the page with the number are included in collection listing. |
| paging | pageSize | G | positive integer | The page size. Default value is 30. |

Resource Representations

REST Representations are views of resources. In SOA Systinet, they are HTTP messages using one of the two models shown in the following table:

Table 3. Models of Communication

| Model | Description |
|---------------------------------|--|
| Raw | <p>This is the default (if no URL query fields configure the XML model).</p> <p>The resource is represented by its raw data.</p> <p>For documents, the raw data is contained in the message body (for both request and response) or in the first part of a multi-part request message resulting from the use of a file upload on an HTML form. For details, see RFC1867 [http://www.zvon.org/tmRFC/RFC1867/Output/chapter1.html].</p> <p>In a request, the type of the raw data is specified with an HTTP <code>Content-type</code> value, either in the message header or in the message part containing the data. In response, the <code>Content-Type</code> is set to the value stored in the resource's metadatum <code>content-type</code>. It is <i>not</i> <code>application/xml</code>, because it is used for REST resource serialization model.</p> <p>For collections, an XML serialization is always used in place of a native format, as shown in Example 8 on page 48.</p> |
| XML REST resource serialization | <p>This model is requested by specific URL query fields in the HTTP request, as described in XML REST Resource Serialization Model on page 49.</p> <p>The resource is represented in the canonical XML format, described by the XML schema in the Resource Serialization Schema section in the HP SOA Systinet Reference Guide. This supports the content types specified by the URL query fields. The namespace prefix <code>rest:</code> is used for its target namespace in HTTP responses.</p> <p>The HTTP <code>Content-type</code> header is <code>application/xml</code>.</p> |

Representations of resources are encoded in the HTTP message body. Type information is also given in:

- The HTTP `content-type` header
- URL query parameters

Example 8: XML Serialization of a Single-item Collection

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/rest/repository/test/"
  xml:base="http://localhost:8080/soa/systinet/platform/rest/repository/"
  type="collection" name="test/"
  requestURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/"
  readURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?update"
  updateRevisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?update&revision=1"

  purgeURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?purge"
  viewURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?view"
  createURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?create"
  aclURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
  <rest:data representation="list">
    <rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/rest/repository/test/a"
      type="document"
      name="a"
      readURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?meta&desc&data"
      revisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?revision=1"
      updateURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?update"
      updateRevisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?update&revision=1"

      deleteURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?delete"
      deleteRevisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?delete&revision=1"

      undeleteURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?undelete"
      purgeURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?purge"
      purgeRevisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?purge&revision=1"

      viewURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?view"
      viewRevisionURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?view&revision=1"
```

```
aclURI="http://localhost:8080/soa/systinet/platform/rest/repository/test/a?acl"
revision="1"/>
</rest:data>
</rest:resource>
```

XML REST Resource Serialization Model

Messages that use the XML model, as shown in [Table 3](#), have XML content in the format described in the Repository Element Formats section in the HP SOA Systinet Reference Guide. This has a number of optional elements, meeting the various requirements of the REST interface.

The following messages are in the XML REST resource serialization model:

- **Responses**

- CREATE, UPDATE, DELETE, UNDELETE, and PURGE response messages

Returned `Resource` contains the metadata of the requested resource.

- GET response message

Requests, which have set `data`, `meta`, `acl`, `desc` in the request URL. Returned `Resource` contains parts according to the request.

- GET on a collection response message

If there is no `data`, `meta`, `acl` or `desc`, in the request URL, then the default is `data`. Returned `Resource` contains parts according to the request.

If the request for collection's data contains `meta` and/or `desc` representation option, the metadata or descriptor of the documents are also included. Note, however, that these data are not fully processed and, for example, information about relations is not included.

- **Request**

- CREATE and UPDATE request message

`Resource` in a request message has set resource parts.

For full details, see the Repository Element Formats section in the HP SOA Systinet Reference Guide. The root element of the representation is described, and it provides a number of attributes designed for the REST interface.

Resource Revision Identification

Each resource modification creates a new resource revision. Each revision is identified by these parameters:

- **Revision number**

The revision number can be specified by a URI parameter `revision`, as shown in [Table 2](#), where the value is the revision number. The first revision (a new resource) is 1. Value 0 represents the latest revision.

- Timeslice (datetime)

Timeslice identifies the revision using the value of parameter `datetime`, as shown in [Table 2](#).



- Collection and document revision semantics differ - the collection revision is incremented by a meta update, not on create/update/delete/purge of subresources. Document revision is incremented by a meta/descriptor/data update.
- Revision is not incremented by an incoming relationship creation. That is, if another resource establishes a relationship targeting the resource.

The following table presents a combination of `revision` and `datetime` parameters:

Table 4. URL Query Parameters: revision and datetime

| Parameters | GET url | relationship in CREATE, UPDATE | UPDATE, DELETE, PURGE |
|-----------------------------|---|--|---|
| no revision, datetime | Returns last revision | Targets last revision | Resource's last revision |
| revision=0 | Returns last revision | Targets last revision | Resource's last revision |
| revision>=1 | Returns the content of the given revision number. Relationships are as they were at datetime just before revision ended or at the current datetime. | Targets given revision | Resource must be in given revision |
| datetime=DT | Returns the content of the revision at the given DT. Relationships are as they were at DT (timeslice). Relationships in metadata and descriptor have set datetime parameter of DT | Targets revision at given DT | Resource revision in DT must be the last revision |
| revision=0, datetime=DT | See datetime=DT | Targets last revision | See datetime=DT |
| revision>=1, datetime=DT | Returns the content of the given revision number. Relationship are as they were at given DT (if DT falls into datetime of given revision, otherwise datetime just before revision end of current datetime. Relationships in metadata and descriptor have set datetime parameter of DT | Targets given revision (datetime is ignored) | Resource must be in given revision |



If datetime has no specified value (DT), then it is set as follows:

- If no revision parameter is given or `revision=0`, then it DT is set to datetime when request is processed. That is, current datetime.
- If `revision>=1`, then DT is set to datetime of the revision

REST Operations

To use the SOA Systinet REST interface, applications must map each operation to an HTTP request. For details, see [Table 5](#).

SOA Systinet REST operations map to HTTP GET, POST, and HEAD only, because these requests can be received by servers or via proxies that do not support HTTP PUT and DELETE.

Each REST operation is executed on a resource identified by a request URL. For CREATE, the URL identifies an existing collection in which the new resource will be created. The query parameter `resource` names the new resource.

For other operations (GET, EXIST, UPDATE, DELETE, UNDELETE and PURGE), the resource is identified in the following ways:

- The URL is consistent with CREATE and specifies:
 - a collection, in the path
 - a resource, in query parameter `resource`
- or
- the complete path to the resource is specified in the path, and there is no resource parameter

The name of a collection is always followed by a `/`, whether in the path, or in the resource query parameter. When a resource is created, the new resource is a collection only if the resource parameter includes a trailing `/`.

Table 5. Summary of REST Operations

| REST Operation | HTTP method | Query Field | Notes |
|----------------|-------------|-------------|---|
| CREATE | POST | create | The path specifies the containing collection and the <code>resource</code> URL parameter contains the name of the resource to create (if omitted, then a unique name is generated by the server). |
| GET | GET | None | The data represented in the response depends on the request. |
| GET | POST | get | It is possible to convert all GET requests to POST requests using <code>get</code> , and moving parameters from query part to a body of type <code>multipart/form-data</code> . However, this is not recommended unless the size of parameters is too high. |
| EXIST | GET | exist | Used to check if the resource exists. |
| EXIST | HEAD | none | Used to check if the resource exists. |
| UPDATE | POST | update | Updates the resource. |
| DELETE | POST | delete | Only for documents. GET, UNDELETE, and PURGE operations can be run on deleted resources. |
| UNDELETE | POST | undelele | Undeletes the deleted resource. It can then be updated again. |
| PURGE | POST | purge | Purge physically removes a resource. |

For operations other than GET and EXIST, the response contains a message from the XML REST resource serialization model, where `Resource` contains the metadata of the requested resource and optionally, the descriptor and/or acl sections (when the request URL contains desc and/or acl query parameters).

CREATE

The request message can contain any model of representation.

For collections, it is not possible to specify the data and so only metadata can optionally be represented.

The created document type is `XML` if the content-type of the data is `text/xml`, otherwise it is `binary`. It is not possible to change the document type using update.

Example 9: Create collection `c/` at the root collection.

POST `http://localhost:8080/soa/systinet/platform/restBasic/repository/?resource=c/&create`



Since this operation requires an HTTP POST request, you cannot simply enter the URL into a browser. Typically the request is coded in an application. It is possible to use Javascript or HTTP command line clients.

Example 10: Response to Example 9 on page 54

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource
  xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  type="collection"
  name="c/"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/?create=&resource=c%2F"

  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?update"

updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?update&revision=1"

  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?purge"
  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?view"
  createURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?create"
  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
<rest:metadata>
  <rest:path>c</rest:path>
  <rest:collection></rest:collection>
  <rest:binary>0</rest:binary>
  <rest:type>collection</rest:type>
  <rest:deleted>0</rest:deleted>
  <rest:owner>demouser</rest:owner>
  <rest:revision>
    <rest:number>1</rest:number>
    <rest:timestamp>2007-04-25T16:01:35.031Z</rest:timestamp>
    <rest:creator>demouser</rest:creator>
    <rest:label xsi:nil="true"/>
    <rest:last>1</rest:last>
  </rest:revision>
</rest:relationships/>
```

```
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
</rest:resource>
```

Example 11: Create an XML document

```
POST http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?resource=d&create
POST /soa/systinet/platform/restBasic/repository/c/?create&resource=d HTTP/1.1
User-Agent: Systinet Server for Java/5.5 (Java/1.4.2_10; Windows XP/5.1; build SSJ-5.5-20070426-0008)
Host: localhost:8080
Transfer-Encoding: chunked
Connection: keep-alive
Content-type: text/xml; charset=utf-8
Authorization: Basic ZGVtb3VzZXI6Y2hhbmdlaXQ=

4
<a/>
0
```

Example 12: Response to Example 11 on page 56

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.0.5.GA (build: CVSTag=Branch_4_0 date=200610162339)/Tomcat-5.5
Location: http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d
Content-Type: application/xml;charset=utf-8
Transfer-Encoding: chunked
Date: Thu, 26 Apr 2007 11:35:30 GMT

ab0
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/" type="document" name="d"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?create=&resource=d"

  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update"

updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update&revision=1"

  deleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete"

deleteRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete&revision=1"

  undeleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete"
  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge"
  purgeRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge&revision=1"

  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view"
  viewRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view&revision=1"

  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl" revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">

<rest:metadata>
<rest:path>c/d</rest:path>
<rest:collection>c</rest:collection>
```

```
<rest:binary>0</rest:binary>
<rest:contentType>text/xml; charset=utf-8</rest:contentType>
<rest:type>document</rest:type>
<rest:deleted>0</rest:deleted>
<rest:owner>demouser</rest:owner>
<rest:revision>
  <rest:number>1</rest:number>
  <rest:timestamp>2007-04-26T11:35:29.812Z</rest:timestamp>
  <rest:creator>demouser</rest:creator>
  <rest:label xsi:nil="true"/>
  <rest:last>1</rest:last>
</rest:revision>
<rest:relationships/>
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
</rest:resource>
0
```

GET

The REST GET differs significantly from other operations, as shown in [Table 5](#). In particular, there is no need to specify the operation with a query parameter, because it is one of two operations that maps to HTTP GET.

The response contains a representation of the resource depending on the request. Parameters can specify which representation is required. The default is Raw. For details, see [Resource Representations on page 46](#).

For details of REST GET operations that execute the resource they operate on and return the execution result as a REST representation, see [Executable Objects on page 72](#).

Example 13: Get collection /c/

```
GET http://localhost:8080/soa/systinet/platform/restBasic/repository/c/
```

The content of a response to a similar request is shown in [Example 8 on page 48](#).

Example 14: Get the XML serialization of a document

```
http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&data&acl
```

Example 15: Response to Example 14 on page 58

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  type="document"
  name="d"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl=&data=&meta="

  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update"

updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update&revision=1"

  deleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete"

deleteRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete&revision=1"

  undeleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete"
  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge"
  purgeRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge&revision=1"

  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view"
  viewRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view&revision=1"

  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
<rest:metadata>
  <rest:path>c/d</rest:path>
  <rest:collection>c</rest:collection>
  <rest:binary>0</rest:binary>
  <rest:contentType>text/xml; charset=utf-8</rest:contentType>
  <rest:type>document</rest:type>
  <rest:deleted>0</rest:deleted>
  <rest:owner>demouser</rest:owner>
```

```

<rest:revision>
  <rest:number>1</rest:number>
  <rest:timestamp>2007-04-26T11:35:29.812Z</rest:timestamp>
  <rest:creator>demouser</rest:creator>
  <rest:label xsi:nil="true"/>
  <rest:last>1</rest:last>
</rest:revision>
<rest:relationships/>
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
<rest:acl>
  <rest:owner>demouser</rest:owner>
  <rest:ace>
    <rest:principal type="group">system#everyone</rest:principal>
    <rest:permission>read</rest:permission>
  </rest:ace>
  <rest:effectivePermissions>
    <rest:permission>read</rest:permission>
    <rest:permission>write</rest:permission>
  </rest:effectivePermissions>
</rest:acl>
<rest:data representation="xmlData">
  <rest:xmlData contentType="text/xml; charset=utf-8"
    xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d">
    <a/>
  </rest:xmlData>
</rest:data>
</rest:resource>

```

Search in collection

The collection search functionality enables you to choose which documents are included in a collection listing. The listing contains only documents whose descriptor satisfies the conditions expressed in the request.

The search does not work for documents without a descriptor. A condition containing a property which is not in the descriptor's schema is never satisfied.

The condition consists of a name of a property, prefixed with `cs_`. The value is a string. If the value is prefixed with an asterisk (*), it is interpreted as case insensitive substring match. Otherwise, it is case sensitive. For properties with multiple value, one matching value is enough to satisfy the condition.

The query can contain more conditions. More conditions for one property form a set of alternatives (logical OR). Finally all remaining conditions and alternatives are joined with an AND logical operation.

It is possible to set the order of the documents using parameters, starting with the name of the property prefixed with `co_`. For example `co_name=1,asc` or equivalently `co_name=asc`. The number is precedence of the ordering. The property cannot be of some type with (potentially) multiple values.

Example 16: All WSDLs containing substring `hello` in its name

```
GET http://localhost:8080/soa/systinet/platform/rest/repository/wsdlcs/?cs_name=*hello
```

Example 17: All person artifacts ordered by name

```
GET http://localhost:8080/soa/systinet/platform/rest/repository/contactArtifacts/?data&cs_artifactType=urn:com:systinet:soa:model:artifacts:content:contact:person&co_name=asc
```

Search options can be combined with `meta`, `desc`, and `rss`.

EXIST

The EXIST operation is used to check the existence of a resource.

It is bound to HTTP GET or HEAD. The response does not contain an HTTP body, it contains only HTTP headers. The most important part of the response is the HTTP status code: 200 – resource exists, or 404 – resource does not exist.

Example 18: Check the existence of resource `/c/a`

```
GET http://localhost:8080/soa/systinet/platform/restBasic/repository/c/a?exist
```

Example 19: Response to [Example 18 on page 61](#)

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.0.5.GA (build: CVSTag=Branch_4_0 date=200610162339)/Tomcat-5.5
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 0
Date: Thu, 26 Apr 2007 12:17:22 GMT
```

Example 20: Check the existence of resource `/c/a`

```
HEAD http://localhost:8080/soa/systinet/platform/restBasic/repository/c/a
```

Example 21: Response to [Example 20 on page 62](#)

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.0.5.GA (build: CVSTag=Branch_4_0 date=200610162339)/Tomcat-5.5
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 0
Date: Thu, 26 Apr 2007 12:17:22 GMT
```

UPDATE

A similar operation to CREATE, this creates a new revision of the resource.

As with CREATE, it is not possible to specify the content of a collection.

Example 22: Update a document

```
POST http://localhost:8080/soa/systinet/platform/restBasic/repository/c/?resource=d&update
```



You cannot enter the URL into a browser, as this will result in a GET request.

DELETE

A successful DELETE results in a response containing resource metadata, which shows that it has been deleted.

Example 23: Delete a document and return the XML serialization

POST <http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete>



You cannot enter the URL into a browser, as this will result in a GET request.

Example 24: Response to Example 23 on page 63

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  type="document"
  name="d"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete="
  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update"
  updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update&revision=1"

  deleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete"
  deleteRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete&revision=1"

  undeleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete"
  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge"
  purgeRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge&revision=1"

  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view"
  viewRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view&revision=1"

  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
  <rest:metadata>
    <rest:path>c/d</rest:path>
    <rest:collection>c</rest:collection>
    <rest:binary>0</rest:binary>
    <rest:contentType>text/xml; charset=utf-8</rest:contentType>
    <rest:type>document</rest:type>
    <rest:deleted>1</rest:deleted>
    <rest:owner>demouser</rest:owner>
    <rest:revision>
      <rest:number>1</rest:number>
      <rest:timestamp>2007-04-26T12:41:19.906Z</rest:timestamp>
```

```
<rest:creator>demouser</rest:creator>
<rest:label xsi:nil="true"/>
<rest:last>1</rest:last>
</rest:revision>
<rest:relationships/>
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
</rest:resource>
```

UNDELETE

A successful UNDELETE results in a response containing resource metadata, which shows that it has been undeleted.

Example 25: Undelete a document and return the XML serialization

```
POST http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete
```



You cannot enter the URL into a browser, as this will result in a GET request.

Example 26: Response to Example 25 on page 65

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  type="document"
  name="d"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete="
  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update"
  updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update&revision=1"

  deleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete"
  deleteRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete&revision=1"

  undeleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete"
  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge"
  purgeRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge&revision=1"

  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view"
  viewRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view&revision=1"

  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
  <rest:metadata>
    <rest:path>c/d</rest:path>
    <rest:collection>c</rest:collection>
    <rest:binary>0</rest:binary>
    <rest:contentType>text/xml; charset=utf-8</rest:contentType>
    <rest:type>document</rest:type>
    <rest:deleted>0</rest:deleted>
    <rest:owner>demouser</rest:owner>
    <rest:revision>
      <rest:number>1</rest:number>
      <rest:timestamp>2007-04-26T12:41:19.906Z</rest:timestamp>
```

```
<rest:creator>demouser</rest:creator>
<rest:label xsi:nil="true"/>
<rest:last>1</rest:last>
</rest:revision>
<rest:relationships/>
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
</rest:resource>
```

PURGE

A successful PURGE results in a response containing resource metadata, which shows that it has been purged:

Example 27: Delete a document and return the XML serialization

POST <http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge>



You cannot enter the URL into a browser, as this will result in a GET request.

Example 28: Response to Example 27 on page 67

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d"
  xml:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  type="document"
  name="d"
  requestURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge="
  readURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?meta&desc&data"
  revisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?revision=1"
  updateURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update"
  updateRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?update&revision=1"

  deleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete"
  deleteRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?delete&revision=1"

  undeleteURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?undelete"
  purgeURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge"
  purgeRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?purge&revision=1"

  viewURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view"
  viewRevisionURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?view&revision=1"

  aclURI="http://localhost:8080/soa/systinet/platform/restBasic/repository/c/d?acl"
  revision="1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
  <rest:metadata>
    <rest:path>c/d</rest:path>
    <rest:collection>c</rest:collection>
    <rest:binary>0</rest:binary>
    <rest:contentType>text/xml; charset=utf-8</rest:contentType>
    <rest:type>document</rest:type>
    <rest:deleted>1</rest:deleted>
    <rest:owner>demouser</rest:owner>
    <rest:revision>
      <rest:number>1</rest:number>
      <rest:timestamp>2007-04-26T12:41:19.906Z</rest:timestamp>
```

```
<rest:creator>demouser</rest:creator>
<rest:label xsi:nil="true"/>
<rest:last>1</rest:last>
</rest:revision>
<rest:relationships/>
<rest:cached>0</rest:cached>
<rest:checksum>0</rest:checksum>
</rest:metadata>
</rest:resource>
```

REST Exceptions

Exceptions that result from a REST operation are represented in the HTTP response in XML.

Example 29: Error Response

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:exception xmlns:base="http://localhost:8080/soa/systinet/platform/restBasic/repository/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:a="http://systinet.com/2005/05/soa/model/artifact"
  xmlns:r="http://systinet.com/2005/05/repository"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://systinet.com/2005/05/soa/model/property/type"
  xmlns:rest="http://systinet.com/2005/05/soa/resource"
  xmlns:p="http://systinet.com/2005/05/soa/model/property"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:g="http://systinet.com/2005/05/soa/model/propertyGroup">
  <rest:code>r:document-not-found</rest:code>
  <rest:message>{http://systinet.com/2005/05/repository}document-not-found:
    REST request processing failed. Method: GET, URL:
    http://localhost:8080/soa/systinet/platform/restBasic/repository/absent.xml
  </rest:message>
  <rest:stackTrace>com.systinet.platform.RepositoryException: {http://systinet.com/2005/05/repository}document-
not-found:
  REST request processing failed. Method: GET, URL:
  http://localhost:8080/soa/systinet/platform/restBasic/repository/absent.xml

  at com.systinet.platform.rest.service.RestService.process(RestService.java:236)
  at com.systinet.platform.servlet.processing.RawServiceClassWrappingServlet.
  genericDo(RawServiceClassWrappingServlet.java:139)
  at com.systinet.platform.servlet.processing.RawServiceClassWrappingServlet.
  doGet(RawServiceClassWrappingServlet.java:126)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:697)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:810)
  at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:252)
  at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
  at com.systinet.platform.servlet.processing.security.HttpBasicFilter.doFilter(HttpBasicFilter.java:116)
  at
  com.systinet.platform.servlet.processing.security.AbstractSecurityFilter.doFilter(AbstractSecurityFilter.java:72)

  at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:202)
  at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
  at com.systinet.platform.servlet.processing.security.SiteminderFilter.doFilter(SiteminderFilter.java:95)

  at
  com.systinet.platform.servlet.processing.security.AbstractSecurityFilter.doFilter(AbstractSecurityFilter.java:72)

  at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:202)
  at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
```

```

at com.systinet.platform.servlet.processing.security.InitSecurityFilter.doFilter(InitSecurityFilter.java:30)

at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:202)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
at org.jboss.web.tomcat.filters.ReplyHeaderFilter.doFilter(ReplyHeaderFilter.java:96)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:202)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:173)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:213)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:178)
at org.jboss.web.tomcat.security.SecurityAssociationValve.invoke(SecurityAssociationValve.java:175)
at org.jboss.web.tomcat.security.JaccContextValve.invoke(JaccContextValve.java:74)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:126)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:105)
at org.jboss.web.tomcat.tc5.jca.CachedConnectionValve.invoke(CachedConnectionValve.java:156)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:107)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:148)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:869)
at
org.apache.coyote.http11.Http11BaseProtocol$Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:664)

at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket(PoolTcpEndpoint.java:527)
at org.apache.tomcat.util.net.MasterSlaveWorkerThread.run(MasterSlaveWorkerThread.java:112)
at java.lang.Thread.run(Thread.java:595)
Caused by: com.systinet.platform.RepositoryException: {http://systinet.com/2005/05/repository}document-
not-found:
The document absent.xml was not found.

at com.systinet.platform.rdbms.runtime.impl.Resource.invokeGetResourceInternal(Resource.java:1383)
at com.systinet.platform.rdbms.runtime.impl.Resource.getResourceInternal(Resource.java:1327)
at com.systinet.platform.rdbms.runtime.impl.Resource.getResourceByPath(Resource.java:323)
at com.systinet.platform.rdbms.runtime.xmldbadapter.DbSessionImpl.doGetResource(DbSessionImpl.java:109)
at com.systinet.platform.xmldb.DbSession.getResource(DbSession.java:271)
at com.systinet.platform.impl.SessionImpl.getDocument(SessionImpl.java:426)
at com.systinet.platform.rest.service.RestHelper.getDocumentResource(RestHelper.java:620)
at com.systinet.platform.rest.service.RestGETProcessing.processGetRaw(RestGETProcessing.java:751)
at
com.systinet.platform.rest.service.RestGETProcessing.getProcessSetHttpOkNoCache(RestGETProcessing.java:206)

at com.systinet.platform.rest.service.RestGETProcessing.processGetPrepared(RestGETProcessing.java:147)
at com.systinet.platform.rest.service.RestGETProcessing.processGet(RestGETProcessing.java:117)
at com.systinet.platform.rest.service.RestService.executeOperation(RestService.java:279)
at com.systinet.platform.rest.service.RestService.process(RestService.java:226)
... 34 more
</rest:stackTrace>
</rest:exception>

```

The `Content-type` of the response is `text/xml`. The interpretation of HTTP response codes for different operations is summarized in the following table.

Table 6. Possible HTTP Response Codes

| Code | CREATE | GET | UPDATE | DELETE | Meaning |
|------------------|--------|-----|--------|--------|--|
| 400 Bad Request | yes | yes | yes | yes | Bad request if: <ul style="list-style-type: none"> the request REST operation is invalid serialization is erroneous mime type is not supported resource is not supported |
| 401 Unauthorized | yes | yes | yes | yes | Authentication failure. Credentials are required or were invalid. Serialization of the exception is not provided because the Java object is not available. |
| 403 Forbidden | yes | yes | yes | yes | The current user does not have the right to perform the requested action. |
| 404 Not Found | yes | yes | yes | yes | The resource does not exist – the containing collection in the case of CREATE. |
| 409 Conflict | yes | no | yes | yes | Conflict, if the resource already exists for create or concurrent modification for update and delete. |

Executable Objects

In SOA Systinet there are two kinds of objects that can be executed: functional resources and some artifacts, for example, task artifacts. What these objects have in common is that they are basically a description of a function which builds its result based on (some part of) data stored on the repository.

Functional Resources

A functional resource is a piece of code that can handle a request and somehow provide a response. In SOA Systinet, they are accessible through the REST interface with URIs that contain `service` namespace. For details, see [Table 1](#).

For example:

```
http://localhost:8080/soa/systinet/platform/rest/service/system/product-information
```

It is possible to map more resources to the same collection (e.g. `system/`). However, it is not possible to map one functional resource under another one. For example it would be illegal to add a resource mapped to `system/product-information/jvm-information`; the space already belongs to the product information functional resource.

A functional resource (or its programmer) can choose which HTTP operations to support, and is responsible for handling the request parameters.

The functional resources are meant to replace XQueries which, in the previous versions, serve the same purpose.

For example the WSIL functionality is now implemented by the functional resource available at:

```
http://localhost:8080/soa/systinet/platform/rest/service/system/wsil
```

However, in the previous version the XQuery providing the WSIL document was located elsewhere. This problem is addressed by the introduction of aliases: documents with a special content type that are located in the `repository` namespace. An execute request for the alias is forwarded to the associated functional resource.

For example, the following request returns the same document as the previous one:

```
GET http://localhost:8080/soa/systinet/platform/rest/repository/queries/wsil.xqy?execute
```

Executable Artifacts

Several kinds of repository documents can be executed: task artifacts, saved search artifacts, and aliases. A document is executed by a request containing an `execute` parameter.



The parameter `execute` is also used in conjunction with the `fulltext` parameter, to run a full text search through a given collection (and its subcollections).

The result of an execution of a task artifact is the resulting report document.

The representation of the result can be modified by request parameters `data` or `rss`.

The last kind executable documents are aliases. They are, technically speaking, not artifacts but they appear in a collection for artifacts. The data they contain is a reference to associated functional resource:

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/rest/repository/queries/wsil.xqy"
  xml:base="http://localhost:8080/soa/systinet/platform/rest/repository/"
  type="document"
  name="wsil.xqy"
  ...
>
<rest:metadata>
  <rest:path>queries/wsil.xqy</rest:path>
  <rest:collection>queries</rest:collection>
  <rest:binary>0</rest:binary>
  <rest:contentType>x-application/alias</rest:contentType>
  <rest:type>document</rest:type>
  <rest:deleted>0</rest:deleted>
  <rest:owner>systinet:admin</rest:owner>
  <rest:revision>
    <rest:number>1</rest:number>
    <rest:timestamp>2007-04-26T09:42:31.578Z</rest:timestamp>
    <rest:creator>systinet:admin</rest:creator>
    <rest:label xsi:nil="true"/>
    <rest:last>1</rest:last>
  </rest:revision>
  <rest:relationships/>
  <rest:cached>0</rest:cached>
  <rest:checksum>0</rest:checksum>
  <rest:extensions>
    <r:alias xmlns:r="http://systinet.com/2005/05/repository"/>system/wsil</r:alias>
  </rest:extensions>
</rest:metadata>
<rest:descriptor/>
<rest:data representation="xmlData">
  <rest:xmlData contentType="x-application/alias"
    xml:base="http://localhost:8080/soa/systinet/platform/rest/repository/queries/wsil.xqy"/>
```

```
</rest:data>  
</rest:resource>
```

Proprietary REST Client

The Java REST HTTP client hides the technical details of the REST protocol.

It is composed of the following base packages and classes:

- `org.systinet.platform.rest`

A package containing the foundation of the client that is used through its implementation, mainly:

- `org.systinet.platform.rest.Client`

REST Client implementation.

- `org.systinet.platform.rest.Source`

Hides REST request data format complexity.

- `org.systinet.platform.rest.ClientException`

Client exception thrown by the client.

- `org.systinet.platform.rest.RestHelper`

Constants and helper methods.

- `org.systinet.platform.rest.schema.model.xsd`

A package containing object representation classes of a REST resource.

- `org.systinet.platform.rest.schema.model.xsd.Resource`

The root class of the REST resource serialization.

For more details, see the Javadoc located in `SOA_HOME/doc/api`.

Basic Principles

This section will show you how to interact with SOA Systinet using the REST HTTP client.

For a REST GET, use the following steps:

1 GET credentials.

It is possible to omit this step and access the server without any credentials - using anonymous access. However, only publicly visible documents can be accessed in this way.

The other option is to use HTTP Basic authentication:

```
Credentials credentials = SecurityHelper.createCredentials("demouser", "changeit",  
SecurityHelper.HttpBasic);
```

2 Construct the resource URL. For HTTP Basic authentication the base URL is:

```
public static String RESTBaseHttpBasicUrl =  
"http://localhost:8080/soa/systinet/platform/restBasic/repository/";
```

For single sign-on, it resembles the following URL:

```
public static String RESTBaseAnonymousUrl =  
"http://localhost:8080/soa/systinet/platform/rest/repository/";
```

A resource URL is composed of a server base URL, collection name, and the resource name:

```
String resourceUrl = RESTBaseHttpBasicUrl + "businessServiceArtifacts/test";
```

3 GET resource by invoking the GET method on the REST client. In this example, we GET the REST resource representation with sections data, meta, descriptor, and ACL:

```
Resource resource = Client.get(resourceUrl, AccessMode.DATA_META_DESC_ACL, credentials);
```

4 Output the result:

```
System.out.println(resource.getMetadata().getPath());
```

The complete code fragment (HTTP Basic) is shown below:

```
// Copyright 2001-2007 Systinet Corp. All rights reserved.
// Use is subject to license terms.

package example;

import org.idoox.security.Credentials;
import org.idoox.wasp.SecurityHelper;
import org.systinet.platform.rest.AccessMode;
import org.systinet.platform.rest.Client;
import org.systinet.platform.rest.Source;
import org.systinet.platform.rest.schema.model.xsd.Resource;

public class RESTExample {
    public static String RESTBaseHttpBasicUrl =
        "http://localhost:8080/soa/systinet/platform/restBasic/repository/";

    public static void main(String[] args) throws Exception {
        Credentials credentials =
            SecurityHelper.createCredentials("demouser", "changeit", SecurityHelper.HttpBasic);

        String resourceUrl = RESTBaseHttpBasicUrl + "businessServiceArtifacts/test";

        Resource resource = Client.get(resourceUrl, AccessMode.DATA_META_DESC_ACL, credentials);
        System.out.println(resource.getMetadata().getPath());
    }
}
```

To REST CREATE, follow these steps:

- 1 GET credentials in the same way as described in the inquiry case.
- 2 CREATE resource parts (in this example we provide data only):
Source data = new Source("example text content");
- 3 Use a secure endpoint to publish the artifact:

```
Resource resource = Client.createDocument(rootCollectionUrl, "test", data, credentials);
```

The complete code fragment (HTTP Basic) is shown below:

```
// Copyright 2001-2007 Systinet Corp. All rights reserved.
// Use is subject to license terms.

package example;

import org.idoox.security.Credentials;
import org.idoox.wasp.SecurityHelper;
import org.systinet.platform.rest.AccessMode;
import org.systinet.platform.rest.Client;
import org.systinet.platform.rest.Source;
import org.systinet.platform.rest.schema.model.xsd.Resource;

public class RESTExample {
    public static String RESTBaseHttpBasicUrl =
        "http://localhost:8080/soa/systinet/platform/restBasic/repository/";

    public static void main(String[] args) throws Exception {
        Credentials credentials =
            SecurityHelper.createCredentials("demouser", "changeit", SecurityHelper.HttpBasic);

        String rootCollectionUrl = RESTBaseHttpBasicUrl;

        Source data = new Source("example text content");
        Resource resource = Client.createDocument(rootCollectionUrl, "test", data, credentials);

        System.out.println(resource.getMetadata().getPath());
    }
}
```

For more details, see the HP SOA Systinet Demo Guide and [Chapter 2, REST Interface](#).

REST Client Package

This section describes how to use the client distribution. This client allows you to access SOA Systinet through a REST HTTP interface.

The installation program creates the client distribution in the subdirectory, `client`, of the directory in which SOA Systinet is installed. In this section, the system property `CLIENT_HOME` refers to this directory.

▶ The `CLIENT_HOME` directory contains all required files and can be copied to a location of your choice.

The `CLIENT_HOME` contains three subdirectories:

- `bin` - shell scripts for running the REST tools. Not necessary unless you want to use these tools.
- `conf` - files with client's configuration.
- `lib` - jar files that compose the client.

▶ Directory `conf/sdm/`, including its content and file `lib/platform_sdm.jar`, are not necessary for the REST client. They are used only in the SDM client.

▶ If you want to use an HTTPS connection to an SOA Systinet server, you must import the server's certificate into the truststore using the standard Java `keytool` command. The recommended location and name is `CLIENT_HOME/conf/client.truststore`.

▶ You do not have to place client files to directories that have specific names. For example, all client files can be copied to the flat directory.

Client Classpath

For each Java program using the REST client, the associated `.jar` files must be added to the classpath. The classpath must contain all the `.jar` files located in the `lib` directory of the client distribution (except the `platform_sdm.jar`):

```
jdbc-driver-path.jar
extensions-lib.jar
pl-repository-old.jar
pl-repository-api.jar
sc-merged-sdm.jar
pl-model.jar
pl-persistence.jar
client-utils.jar
configuration.jar
```

Client Environment

To run your HP SOA Registry Foundation client code, you must set the following Java properties:

```
-Dwasp.location=CLIENT_HOME  
-Dwasp.config.location=conf/clientconf.xml  
-Djava.security.auth.login.config=CLIENT_HOME/conf/jaas.config  
-Didoox.debug.level=1  
-Didoox.debug.logger=log4j  
-Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog
```

If the client will use HTTPS transport, set this additional property:

```
-Djavax.net.ssl.trustStore=CLIENT_HOME/conf/client.truststore
```



Replace `CLIENT_HOME` with an appropriate directory name or variable.

3 SDM Client

The SDM client provides a high-level abstraction of artifact descriptors and relationships. It is built on top of the [Proprietary REST Client on page 75](#), and provides methods to speed up modeling and interaction with SOA Systinet.

It is composed of the following base packages and classes:

- `org.systinet.platform.sdm`

A package containing the foundation of the client that is used through its implementation, mainly:

- `org.systinet.platform.sdm.SdmClientConstants`

Constants used to reference particular repository collections where artifacts are stored and taxonomy URN.

- `org.systinet.platform.sdm.SdmClientHelper`

Helper class providing useful methods for artifacts manipulation.

- `org.systinet.platform.sdm.SdmClient`

Core of the generic client allowed to work with any artifact type. On top of this core is a specific adaptor for each artifact. This client is not typically used directly.

- `org.systinet.platform.sdm.xsd.artifact`

A package containing implementation classes for each artifact type. Each artifact type class contains CRUD methods allowing the creation of artifact instances and their manipulation.

- `org.systinet.platform.sdm.xsd.artifact.Artifact`

Superclass of all artifact type classes.

- `org.systinet.platform.sdm.xsd.property`

Implementation of all artifact properties.

For more details please see the Javadoc located in `SOA_HOME/doc/api`.

Basic Principles

This section shows how to interact with SOA Systinet using the SDM Client.

For an inquiry pattern, use the following steps:

- 1 Get credentials.

Unless you want to use anonymous access, it is necessary to create HTTP Basic credentials:

```
Credentials credentials
    =SecurityHelper.createCredentials("demouser", "changeit", SecurityHelper.HttpBasic);
```

- 2 Construct the artifact URL. For SSL transport HTTP Basic authentication the base URL follows this example:

```
static final String restEndpoint="https://localhost:8843/soa/systinet/platform/restBasic/repository/";
```

In the case of anonymous access, use the following URL:

```
static final String restEndpoint="https://localhost:8843/soa/systinet/platform/rest/repository/";
```

An artifact URL is comprised of a server base URL, collection name, and the artifact name:

```
String webServiceUrl
    = restEndpoint+"/"+SdmClientConstants.COLLECTION_NAME_WEBSERVICE+"/"+artifactName;
```

- 3 Get the artifact by invoking the static method on the artifact type class:

```
WebServiceArtifact webServiceArtifact
    = WebServiceArtifact.get(webServiceUrl, credentials);
```

4 Output the result:

```
SdmClientHelper.showArtifact(webServiceArtifact);
```

The complete code fragment (HTTP Basic) is shown below:

```
// Copyright 2001-2007 Systinet Corp. All rights reserved.
// Use is subject to license terms.

package example;

import org.idoox.security.Credentials;
import org.idoox.wasp.SecurityHelper;
import org.systinet.platform.sdm.SdmClientConstants;
import org.systinet.platform.sdm.SdmClientHelper;
import org.systinet.platform.sdm.xsd.artifact.WebServiceArtifact;

public class InquiryExample {
    static final String restEndpoint="https://localhost:8843/soa/systinet/platform/restBasic/repository/";
    static final String artifactName="MyWebServiceArtifact";

    public static void main(String[] args) throws Exception {
        Credentials credentials
            =SecurityHelper.createCredentials("demouser", "changeit", SecurityHelper.HttpBasic);

        String webServiceUrl
            = restEndpoint + "/" + SdmClientConstants.COLLECTION_NAME_WEBSERVICE + "/" + artifactName;

        WebServiceArtifact webServiceArtifact
            = WebServiceArtifact.get(webServiceUrl, credentials);

        SdmClientHelper.showArtifact(webServiceArtifact);
    }
}
```

To publish, follow these steps:

- 1 Get credentials in the same way as described in the inquiry case.

2 Build artifact:

```
WebServiceArtifact artifact = new WebServiceArtifact();

artifact.setNameGroup(new NameGroup(new Name[] {
    new Name("en", "FTP Web Service")
}));

artifact.setDescriptionGroup(new DescriptionGroup(
    new Description[] {
        new Description("en",
            "Web Service artifact representing a Web Service interface to the FTP protocol.")
    }));

AccessPoint accessPoint = new AccessPoint("http://soap.systinet.net:9080/FTPService");
accessPoint.setUseType("Unsecured endpoint");
artifact.setAccessPointGroup(new AccessPointGroup(new AccessPoint[]{
    accessPoint
}));

// production stage
artifact.setProductionStage(
    new ProductionStage(
        "Production",
        SdmClientConstants.TAXONOMY_LIFECYCLE_STAGES,
        "uddi:systinet.com:soa:model:taxonomies:lifecycleStages:production"));
```

3 Use a secure endpoint to publish the artifact:

```
String webServiceUrl
    = WebServiceArtifact.create(
        restEndpoint,
        artifactName,
        buildWebServiceArtifact(),
        null, null, credentials);
```

The complete code fragment (HTTP Basic) is shown below:

```
// Copyright 2001-2007 Systinet Corp. All rights reserved.
// Use is subject to license terms.
package example;

import org.idoox.security.Credentials;
```

```

import org.idoox.wasp.SecurityHelper;
import org.systinet.platform.sdm.SdmClientConstants;
import org.systinet.platform.sdm.xsd.artifact.WebServiceArtifact;
import org.systinet.platform.sdm.xsd.group.AccessPointGroup;
import org.systinet.platform.sdm.xsd.group.DescriptionGroup;
import org.systinet.platform.sdm.xsd.group.NameGroup;
import org.systinet.platform.sdm.xsd.property.AccessPoint;
import org.systinet.platform.sdm.xsd.property.Description;
import org.systinet.platform.sdm.xsd.property.Name;
import org.systinet.platform.sdm.xsd.property.ProductionStage;

public class PublicationExample {
    static final String restEndpoint="https://localhost:8843/soa/systinet/platform/restBasic/repository/";
    static final String artifactName="MyWebServiceArtifact";

    public static void main(String[] args) throws Exception {
        Credentials credentials
            =SecurityHelper.createCredentials("demouser", "changeit", SecurityHelper.HttpBasic);

        String webServiceUrl = WebServiceArtifact.create(
            restEndpoint,
            artifactName,
            buildWebServiceArtifact(),
            null, null, credentials);

        System.out.println("Creates Web Service artifact: "+webServiceUrl);
    }

    private static WebServiceArtifact buildWebServiceArtifact() {
        WebServiceArtifact artifact =new WebServiceArtifact();

        artifact.setNameGroup(new NameGroup(new Name[] {
            new Name("en", "FTP Web Service")
        }));

        artifact.setDescriptionGroup(new DescriptionGroup(
            new Description[] {
                new Description("en", "Web Service artifact representing FTP Web Service.")
            }));

        AccessPoint accessPoint = new AccessPoint("http://soap.systinet.net:9080/FTPService");
        accessPoint.setUseType("Unsecured endpoint");
        artifact.setAccessPointGroup(new AccessPointGroup(new AccessPoint[]{
            accessPoint
        }));
    }
}

```

```
// production stage
artifact.setProductionStage(
    new ProductionStage(
        "Production",
        SdmClientConstants.TAXONOMY_LIFECYCLE_STAGES,
        "uddi:systinet.com:soa:model:taxonomies:lifecycleStages:production"));

return artifact;
}
}
```

For more details, see the HP Systinet Demo Guide and [Chapter 2, REST Interface](#).

SDM Client Package

The SDM client is the same as the REST client described in [REST Client Package on page 78](#).



Do not remove `conf/sdm` and include `lib/platform_sdm.jar` in the classpath.

4 Technical Security

This chapter provides a technical description of SOA Systinet security.

Security is described in the following sections:

- [SOA Systinet Overview on page 87](#)
- [Users and Groups on page 88](#)
- [Transport Security on page 90](#)
- [Authentication on page 91](#)
- [Resource ACL on page 91](#)
- [WEB Security on page 93](#)
- [Platform Services on page 93](#)
- [Reporting Services on page 94](#)
- [Policy Manager Services on page 94](#)

SOA Systinet Overview

SOA Systinet consists of the following components:

- **Web UI**
Exposes the WEB service providing the SOA Systinet UI.
- **Platform**
Provides a repository (data store) for artifacts.

Exposes WEB and REST services to manage artifacts.

- **Policy Manager**

Engine for policy validation.

Exposes REST services to policy management and validation.

- **Reporting**

Store for report definitions and data.

Engine for report generation.

Exposes REST service for report management.

These components are deployed as a single EAR file which is generated by the installation.

For details, see the *HP SOA Systinet Installation and Deployment Guide*.

Users and Groups

SOA Systinet delegates authentication to the J2EE container. The userstore is not managed by SOA Systinet, but by the application server or LDAP/AD tools.

SOA Systinet uses the following definitions:

- **User**

A user represents the identity accessing SOA Systinet.

Use your application or LDAP/AD tools to manage users.

- **User Profile**

Profiles provide additional information for SOA Systinet. For example, a contact email used for mail notifications and a primary group used for collective ownership.

- **Role**

Roles are defined by functional security. They define the actions permitted to a user. Currently, only the *administrator* role is defined.

- **Group**

Groups are defined by organizational security following the company structure.

SOA Systinet uses the following types of groups:

- **external**

Groups defined by LDAP. These must be managed within LDAP.

- **internal (local)**

Groups managed within SOA Systinet by the administrator.

SOA Systinet uses the following user types for processing:

- **authenticated**

A user authenticated by J2EE. For example, a user/password for HTTP.

See [Authentication on page 91](#) for authentication mechanisms.

- **anonymous**

A user who does not pass any credentials and accesses service on access points with an anonymous authentication mechanism.

The name used in ACL is `systinet#anonymous`.

- **resource owner**

A user who owns the accessed resource. Used in ACL evaluation.

- **administrator**

A user with the administrator role. The administrator has the rights to perform all actions (no ACLs are applied on resources, management tasks, and so on).

During installation, you must define an administrator.

SOA Systinet user and group management enables you to assign the administrator role to users or entire user groups.

- **system administrator**

An internal identity used for the execution of internal tasks. It is not possible to authenticate (log in) with this identity. This user has the same capabilities as an administrator.

The name used in ACL is `systinet:admin`.

SOA Systinet uses the following built-in groups:

- `system#registered`

All users who exist in the userstore. In other words, users who are authenticated.

- `system#everyone`

Both authenticated users (group `system#registered`) and anonymous users (`systinet#anonymous`).

Transport Security

SOA Systinet provides several REST and WEB services. They are exposed at access points mapped on the HTTP and HTTPS transports provided by the hosting application server. It also provides installation scenarios where you can enable or disable HTTP or HTTPS.

SOA Systinet does not provide SSL management (certificates) because HTTPS transport is provided by the application server.

For simple JBoss configuration, SOA Systinet provides automatic SSL enablement (certificate generation and SSL configuration) during installation.

On the client side (for example, SOA Systinet accesses HTTPS URLs to upload WSDLs), the handling of SSL certificates is configurable (for example, the selection of truststores, enable/disable hostname verification).

Authentication

Authentication is provided by the J2EE application server. The application server capability determines which method is used (for example, HTTP Basic, SiteMinder). For backward compatibility, it is possible to configure SOA Systinet authentication (SiteMinder and client SSL certificates) but the preferred authentication is via J2EE application servers,

For details on SOA Systinet authentication, see the *HP SOA Systinet Installation and Deployment Guide*.

J2EE session management is used for both WEB and REST services.

Resource ACL

SOA Systinet does not use J2EE authorization to access service resources (for example, REST resources are artifacts and collection or WEB resources are tasks).

Platform, Policy Manager, and Reporting Service components provide hierarchical resource models accessible by REST. In these models there are collections and resources, where a collection can contain both individual resources and other collections.

Platform and Reporting Service use the same ACL model.

When access to a resource is requested, ACL is used to authorize access for a user using the following model:

- An ACL is a list of ACEs, where an ACE is composed of the following model:

- **resource owner**

Can be either a user or a group.

resource owner and **administrator** always have read and write permission granted so ACLs are not evaluated in these cases.

- ACL is a list of ACEs, where an ACE is composed of:

- user or group identification
- granted permission:
 - **read:**
 - `artifact/resource` — permission to read any data and metadata of the artifact.
 - `collection` — permission to read the content and metadata of the collection.
 - **write:**
 - **artifact/resource** — permission to update any data and metadata of the artifact.
 - **collection** — permission to create new artifacts, resources, and sub-collections, and to update the metadata of the collection.

- No negative ACE.

It is not possible to deny permission to a user or group.

- No inheritance or propagation of ACL.

Only the ACL of the accessed artifact is used for authorization.

A change to a collection ACL does not change any ACLs of collection members.

To read or update an artifact, it is sufficient to have read or write permission on the resource.

- When a resource is created, its default ACL is set by artifact. It is possible to configure default ACLs per collection (for example, artifact type).

For details about the default ACL configuration, see "Default ACL Configuration" in the *HP SOA System Reference Guide*.

For details about changing the default ACL configuration, see "Configuring the Default ACLs" in the *HP SOA Systinet Administrator Guide*.

WEB Security

The UI is composed of *tasks* mapped on URLs. All UI tasks require an authenticated user who must sign in to SOA Systinet.

The UI is composed of static tasks, so this setup is part of the WEB configuration.

WEB uses J2EE session management, provided by the application server.

Platform Services

Platform provides two REST services. They are exposed on the following access points, mapped on HTTP and HTTPS transports provided by the hosting application server:

- **Proprietary REST**

`http://host:port/context/systinet/platform/rest/` and `https://host:port/context/systinet/platform/rest/` operate with the *anonymous* authentication mechanism.

`http://host:port/context/systinet/platform/restBasic/` and `https://host:port/context/systinet/platform/restBasic/` operate with the default HTTP Basic authentication mechanism, specified by the application server.

- **Atom-Based REST**

`http://host:port/context/platform/rest/` and `https://host:port/context/platform/rest/` operate with the *anonymous* authentication mechanism.

`http://host:port/context/platform/restSecure/` and `https://host:port/context/platform/restSecure/` operate with the default HTTP Basic authentication mechanism, specified by the application server.

The REST service uses J2EE session management, provided by the application server.

Reporting Services

Reporting provides a REST service. It is exposed on the following access points, mapped on HTTP and HTTPS transports provided by the hosting application server:

- **Atom-Based REST**

`http://host:port/context/reporting/rest/` and `https://host:port/context/reporting/rest/` operate with the *anonymous* authentication mechanism.

`http://host:port/context/reporting/restSecure/` and `https://host:port/context/reporting/restSecure/` operate with the default HTTP Basic authentication mechanism, specified by the application server.

The REST service uses J2EE session management, provided by the application server.

Policy Manager Services

Policy Manager provides a REST service. It is exposed on the following access points, mapped on HTTP and HTTPS transports provided by the hosting application server:

- **Atom-Based REST**

`http://host:port/context/policymgr/rest/` and `https://host:port/context/policymgr/rest/` operate with the *anonymous* authentication mechanism.

`http://host:port/context/policymgr/restSecure/` and `https://host:port/context/policymgr/restSecure/` operate with the default HTTP Basic authentication mechanism, specified by the application server.

The REST service uses J2EE session management, provided by the application server.

5 RSS

RSS (Really Simple Syndication) is an XML-based system for subscribing to information sources. For details, see <http://www.rss-specifications.com/rss-specifications.htm>.

SOA Systinet provides RSS feeds, and can be used to subscribe to others.

This section describes the kind of RSS Feeds supported, where to find them, and how to use the **Feed Reader** dashboard portlet.

Kinds of RSS Feed

SOA Systinet provides the following kinds of RSS feed:

- **document feed**
Contains the document revisions as separate items.
- **collection feed**
Consists of recently changed documents in collections.
- **saved search feed**
The RSS representation of saved search results.

Syndication Syntax

Each request for RSS must contain a URL parameter `rss`, which can be parameterized. The optional value specifies the required format of the syndicate. SOA Systinet supports the following popular syntaxes of syndicates:

- Atom v0.3 (use URL parameter `?rss=atom_0.3`)

- Atom 1.0 (use URL parameter `rss=atom_1.0`)
- RSS 0.9 (use URL parameter `rss=rss_0.9`)
- RSS 0.92 (use URL parameter `rss=rss_0.92`)
- RSS 0.93 (use URL parameter `rss=rss_0.93`)
- RSS 0.94 (use URL parameter `rss=rss_0.94`)
- RSS 1.0 (use URL parameter `rss=rss_1.0`)
- RSS 2.0 (use URL parameter `rss` or `rss=rss_2.0`), which is the default RSS format.

Subscriptions over RSS

Although SOA Systinet has no abstraction of subscriptions, you can be notified of changes to repository data.

Notifications about new items in syndicates are a natural feature of RSS feed readers. Feed readers cache the syndicated items and inform users about new ones from the latest feed.

RSS feed readers use the item attribute, `link`, to recognize if the item has already been read or not. SOA Systinet's RSS feed item identifiers are based on the REST revision URL of the syndicated resource. So, when the resource is created/modified, the URL of the current revision of the resource is changed. The RSS feed reader is then able to recognize that a new item has appeared in the syndication (replacing the old one), and informs the user about the changes.

The main advantage of this kind of subscription is that users need not learn any new proprietary subscription API. Users can use their favorite RSS feed readers or the one implemented as a SOA Systinet dashboard portlet.

6 Custom Source Parsers

The source parser you write creates an object representation of a log of messages—when your input source is only a single message, it creates a log of one message.

The following list specifies a mapping between concepts and classes in HP SOA Systinet Policy Manager API:

- A log of messages corresponds to an instance of `org.systinet.policy.validation.ValidationSourceCollection`. It can contain both inline request/response messages and references to external messages. As credentials are passed along, the external messages can be secured with HTTP basic authentication.
- A request/response conversation (or a single message, if it is one-way) corresponds to an instance of `org.systinet.policy.validation.ValidationSource`. When creating an instance of this class, make sure you set up:
 - `SourceType` – this should be set to `org.systinet.policy.validation.ValidationConstants#Elements.SOURCE_CONVERSATION`, in case of request/response conversation, or `soap:Envelope` for single-message validation.
 - One (for one-way) or two (for request-response conversation) messages.
- A message corresponds to an instance of `org.systinet.policy.validation.ValidationSourceDocument`.

You should set up:

- `content`

The SOAP payload of the message.

- `contentURL`

The url of the SOAP payload. If the SOAP message is inline in the parsed source, you can use `org.systinet.xml.XPointerHelper.appendToURL(java.lang.String, java.lang.String)`, together with

`org.systinet.xml.DOMHelper.getXPointer(org.w3c.dom.Element)` to create a URL pointing directly to the payload.

- **contentBOM** (optional)

The BOM signature of the content.

- **description** (optional)

The WSDL description of the message.

- **descriptionURL** (optional)

URL of the WSDL description of the message.

- **metadata** (optional)

Metadata associated with the message. Anything which is `java.io.Serializable` can be added to the metadata. The built-in handlers understand only

`org.systinet.policy.validation.SOAPMetadataConstants.METADATA_MESSAGE_HEADERS`, which is used as a key to access transport headers.

- **sourceType**

This field should be either `soap:Envelope` to indicate that only a SOAP content is available, or `org.systinet.policy.validation.ValidationConstants#Elements.SOURCE_MESSAGE`, to indicate that additional metadata is available.

- **sourceDocumentURL**

This field should be set to the URL of the whole message; that is, the container for the SOAP payload and metadata. If this container is inlined in a bigger structure, you may use the `XPointerHelper` class mentioned above to get a more detailed URL. If there is no URL, rather than leaving this field empty, use the URL of the SOAP payload or of the whole request/response conversation.

The parser's main method is `public ValidationSourceCollection parse(String uri, String rootElementNamespaceURI, String rootElementLocalName, SourceResolver resolver, CredentialsList credentials)` throws `SourceParseException`, `CredentialsException`. Usually, the parser follows these steps:

- 1 The parser inspects the `rootElementNamespaceURI` and `rootElementLocalName` to determine if the document should be handled by this parser. If not, it returns immediately with `null` and the parsing framework continues with the next parser.
- 2 The parser retrieves the parsed document from the source resolver: `Source source = resolver.getSource(uri, credentials)`. This call fetches the document if this is the first time the document was accessed (this is why credentials must be passed) or uses a cached version if the document has been fetched already. The cache expires when the validation of this source ends.
- 3 The source parser should either create an instance of `ValidationSourceDocument`, pass a reference to another document, or do both. For example, a WSDL source parser creates an instance of `ValidationSource`, adds the parsed WSDL as a new `ValidationSourceDocument`, and then includes each contained/referenced xml schema via `ValidationSource.addReferencedDocument`. All the referenced documents are parsed before the validation starts.
- 4 If the resource being parsed is a collection, the parser should create a `ValidationSourceCollection` and add the references via `addReferencedSource`.

The URL which goes to the `addReferencedXXX` methods might point inside the parsed resource if `XPointer` is used. You can use `DOMHelper.getXPointer()` and `XPointerHelper.appendToURL()` to create such a URL.

To be recognized by the source parsing framework, the parser must be bound to the `/systinet/policy/validation/sources/` JNDI context.

7 Custom Validation Handlers

In addition to the built-in handlers described in the Assertion Schema section in the HP SOA Systinet Reference Guide, you can write and deploy your own validation handlers without further changes to the HP SOA Systinet Policy Manager installation.

The following points should be kept in mind:

- **Home and remote interfaces**

The handler must have `org.systinet.policy.validation.handlers.DialectValidator` as its remote interface and `org.systinet.policy.validation.handlers.DialectValidatorHome` as its remote home interface.

- **Classloaders**

The handler should be deployed within the same classloader. This not only makes sure of better performance, but you also do not have to modify the existing `systinet-policy.ear`. For further details, see `jboss-app.xml`.

- **Deployment path**

The handler must be deployed to the `systinet/policy/validation/handlers/` JNDI context.

- **Exceptions**

The handler should never throw an exception, apart from `org.systinet.http.CredentialsException`. If an error occurs, the handler should always create a report saying that there has been an error.

- **Incoming assertions**

The incoming list of assertions contains instances of `org.systinet.policy.validation.handlers.DialectValidator#AssertionRecord`.

- **Return value**

The return value must be a list of `org.systinet.policy.model.report.Result`. In this list, there is one result for each of the assertions in the incoming list, placed in the same order.

- `getDialect()`

This method returns the URI of the dialect this handler accepts. It must be the same as the namespace URI of the first element in the `pe:Enforcement` section of the assertion definition. It is used to filter the input list of assertions. Only the assertions with this namespace are passed into this handler.

8 Validation Client

Policy Manager includes a command-line validation client that you can copy to another computer on the network. The validation client is designed for the following uses:

- Validating local and/or remote documents against local policies. These validations run on the client.
- Validating remote documents against policies located on a server. These validations run on the server.

The validation client is located at `SOA_HOME/client`. To install the client, copy this folder to the location of your choice.

The validation client command-line tools are located in `SOA_HOME/client/bin`. The tools and their functions are described in the following sections:

- [Downloading Policies and Assertions \(sync\) on page 103](#)
- [Local Validations \(validate\) on page 104](#)
- [Validating Against Policy On Server \(server-validate\) on page 107](#)
- [Rendering Output from XML Reports \(render\) on page 108](#)

Downloading Policies and Assertions (sync)

To perform validations locally, you need local copies of the policies and assertions in the SOA Systinet repository. To download these policies and assertions, run the `sync` tool. Your computer has to be connected to the SOA Systinet server/cluster when you run `sync`.

To run `sync`, simply enter **`sync -u SOA Systinet username -p SOA Systinet password`**. If SOA Systinet does not require any credentials, enter **`sync -noauth`**. The `sync` tool gets the hostname and port of the SOA Systinet host from the `SOA_HOME/client/conf/setup/policy-manager.properties` file, created automatically when SOA Systinet is installed.

The property used is determined by the `shared.https.use` property and is either:

- `shared.http.urlbase=http\://host\;port/context`
- `shared.https.urlbase=https\://host\;8443/context`

Local Validations (validate)

Validate documents against local copies of technical policies by running the `validate` tool. The syntax is `validate [OPTIONS] [--policy local_technical_policy_name,_file_or_uri...] [--source source_file_or_uri...]`. For a full list of options and examples of commands, enter **validate --help**.



Local documents must exist before you can validate them. Download a set of documents with the `sync` tool before running `validate`.

Policy Formats

You can write technical policies in the following formats:

- As the plain text name of the policy, in quotation marks. For example, "Systinet Best Practices".
- As the file name (full or relative) of the policy file. For example, `C:/opt/systinet/policymgr/client/data/policies/systinet-best-practices.xml`.
- As the full URI of the policy. For example, `file:///opt/systinet/policymgr/client/data/policies/systinet-best-practices.xml`.

Source Formats

You can write source document locations in the following formats:

- As the file name (full or relative) of the document. For example, `C:/tmp/services/service1.wsdl`.
- As the full URI of the document. For example, `http://host:port/services/service1.wsdl`.

To validate one source against one policy it is not necessary to include any options in the command line. For example, to validate a local copy of `service1.wsdl` against a local copy of the **Systinet Best Practices** technical policy, you can run **validate "Systinet Best Practices" C:/tmp/services/service1.wsdl**.

Validating Multiple Sources With Multiple Policies

You can validate multiple source documents and/or use multiple technical policies. In this case, it is mandatory to use the `-p|--policy` and `-d|--source` options. For example, **validate -p "Systinet Best Practices" -p file:///opt/systinet/policymgr/client/data/policies/wSDL-validity.xml -d C:/tmp/services/service1.wsdl -d C:/tmp/services/service2.wsdl** validates `service1.wsdl` and `service2.wsdl` against the **Systinet Best Practices** and **WSDL Validity** technical policies.

You can make the validation stop the first time a policy is violated. Use the `-c/--stop` option. For example, the validation launched by **validate --stop -p "WSDL Validity" -p "Systinet Best Practices" -d C:/tmp/services/service1.wsdl -d C:/tmp/services/service2.wsdl** would stop when either `service1.wsdl` or `service2.wsdl` violated either **Systinet Best Practices** or **WSDL Validity**.

Selecting Sources By Wildcard

Instead of specifying every source document to be validated, you can specify a directory of documents and pass a wildcard so all matching documents in that directory will be validated. Specify the directory with the `-d|--source` option and use the `-e|--pattern` to pass the wildcard. For example, **validate -p "Systinet Best Practices" -d C:/tmp/services -e service*.wsdl** would validate `service1.wsdl`, `service2.wsdl`, etc, against the **Systinet Best Practices** technical policy.

Setting Up Output

By default, validation reports are created in text format and printed in the console window. You can save the report as a file by using the `-o|--outputDir` option and the file location. For example, **validate -o C:/tmp/reports "Systinet Best Practices" C:/tmp/services/service1.wsdl** would create the file `C:/tmp/reports/service1.txt`.

Report names are based on source names by default. To give a report a different name, use the `-n|--name` option.

You can produce output in HTML or XML format instead of text. Use the `--format html` or `--format xml` option, respectively. When producing HTML or XML output, specify an output location with the `-o|--outputDir` option. Otherwise the raw HTML or XML is only printed out to the console.

If you produce a report in XML format, you can use it to produce any number of HTML reports with the `render` tool. See [Rendering Output from XML Reports \(render\) on page 108](#).

When the `validate` tool produces HTML output, it uses a template combining XSL and graphics. The validation client comes with a default template that reproduces the Policy Manager report style. You can add additional templates by saving them in the `../client/templates` folder. Specify the template to be used by using the `-m|--template` option. For example, if you saved a custom template in `../client/templates/MyCustomTemplate`, use it to produce HTML output by running **`validate.sh --format html --template MyCustomTemplate [-p policy] [-d source]`**. If you do not specify a template, the default template is used.

ANT Task Automation of `validate`

You can automate the execution of the `validate` tool as an ANT task. Write an ANT script to launch `validate` and save the script in `../client/bin`. Launch it with the **`ant`** command. For example, if you create an ANT script called `/client/bin/validatetask.xml`, launch it with **`ant -f validatetask.xml`**.

The elements of the ANT task are given in [Table 7. Example 1 on page 107](#) is an example of an ANT task script for launching `validate`.

Table 7. `validate` ANT Task Elements

| Element name | Attributes | |
|------------------------------|--|--|
| taskdef | name | Must be <code>validate</code> . |
| | classname | Must be <code>com.systinet.policy.tools.ant.ValidateTask</code> |
| validate (Child of target) | format | Output format. Takes one of <code>xml</code> , <code>html</code> , or <code>txt</code> |
| | policyPropsFile | Specifies Policy Manager properties file. Usually <code>../conf/policy-manager.properties</code> |
| | output | Output file path, such as <code>C:/opt/reports/</code> or <code>C:/tmp/myreport.html</code> . If file name is not specified, it will match the validated source's name or <code>summary.txt xml html</code> if it is a summary report. |
| | cancel | Boolean. <code>true</code> stops the validation at the first failure. |
| policies (Child of validate) | No attributes. Contains a list of all policies to be used for the validation, in nested ANT elements (<code>fileset/include</code>). | |
| sources (Child of validate) | No attributes. Contains a list of sources to be validated, in nested ANT elements (<code>uri</code> , <code>fileset/include</code>). | |

Example 1: validate ANT Task

```
<?xml version="1.0"?>
<project name="validatetool" default="main">
  <taskdef name="validate" classname="com.systinet.policy.tools.ant.ValidateTask"/>
  <target name="main">
    <validate format="html" policyPropsFile="../conf/policy-manager.properties" output="C:/tmp/out">
      <policies>
        <fileset dir="../data/policies/">
          <include name="wsdl-validity.xml"/>
          <include name="systinet-best-practices.xml"/>
        </fileset>
      </policies>
      <sources>
        <uri value="http://api.google.com/GoogleSearch.wsdl"/>
        <fileset dir="../data/policies/">
          <include name="wsdl-validity.xml"/>
        </fileset>
      </sources>
    </validate>
  </target>
</project>
```

Validating Against Policy On Server (server-validate)

Validate a document against a technical policy in an SOA Systinet repository, or remotely run a business policy validation, by running the `server-validate` tool. The tool publishes a report in the same SOA Systinet repository that contains the policy. The URL of the report is printed on the command-line console.

The syntax for validating a document against a technical policy is

```
server-validate [OPTION] {-u SOA Systinet username} {-p SOA Systinet password} [-S SOA Systinet server URL]
{ POLICY_URI } {SOURCE_FILE_OR_URI}
```

. The syntax for running a business policy validation is

```
server-validate [OPTION] {-u SOA Systinet username} {-p SOA Systinet password} [-S SOA Systinet server URL]
{-b BUSINESS_POLICY_URI}
```

For a full list of options and examples of commands, enter **server-validate --help**.

Policy URIs

Policy URIs are in the following formats:

- **Technical policy URI:** `http|https://host:port/soa/systinet/platform/rest/repository/wsPolicies/policy-name`
- **Business policy URI:** `http|https://host:port/soa/systinet/platform/rest/repository/businessPolicies/policy-name`

Source Formats

Only specify a source document if you are validating one against a technical policy. You can write source document locations in the following format:

- As the full URI of the document. For example, `http://api.google.com/GoogleSearch.wsdl`.

Selecting the SOA Systinet Server

By default, the `server-validate` tool communicates with the installation of SOA Systinet from which the validation client was copied. It can use a policy in a different SOA Systinet repository. Specify the SOA Systinet repository with the `-s|--server` option and the URL of the SOA Systinet host. Be careful to use the authorization credentials for that server.

Rendering Output from XML Reports (render)

If you have a report in XML, you can use it to generate HTML reports by running the `render` tool. The syntax is

```
render {--input full_path_to_XML_report} {--outDir output_directory} [OPTIONS]
```

. For a full list of options and examples of commands, enter **render.bat|.sh --help**.

Overwriting Reports

The `render` tool cannot overwrite existing reports of the same name in the same directory. By default, `render` gives the output file the same name as the input file. If a file of the default name already exists and you want to generate a report in the same location, give it a different name by using the `-n|--name` option.

Selecting Output Template

The `render` tool uses a template combining XSL and graphics. The validation client comes with a default template that reproduces the Policy Manager report style. You can add additional templates by saving them in the `../client/templates` folder. Specify the template to be used by using the `-m|--template` option. For

example, if you saved a custom template in `../client/templates/MyCustomTemplate`, use it to produce HTML output by running **`render.sh [-i XML_input_file] [-o output_directory] -m MyCustomTemplate`**. If you do not specify a template, the default template is used.