HP SOA Systinet

Software Version: 2.50, Standard Edition

Reference Guide

Document Release Date: May 2007
Software Release Date: May 2007

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

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

Copyright Notices

Trademark Notices

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

Contents

Welcome to 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.

How This Guide is Organized

SOA Systinet Overview and Introduction describes the basic concept of Services Oriented Architecture, how SOA Systinet manages an SOA, the architecture of the SOA Systinet family of products and an introduction to them.

It contains the following parts:

Part I, "SOA Systinet Repository". A guide to the structure and format of the repository.

Part II, "SOA Definition Model". A guide to the default SOA Definition Model used in SOA Systinet.

Document Conventions

The typographic conventions used in this document are:

run.bat make	Script name or other executable command plus mandatory arguments.
[--help]	A command-line option.
either | or	A choice of arguments.
replace_value	A command-line argument that should be replaced with an actual value.
{arg1 | arg2}	A choice between two command-line arguments where one or the other is mandatory.
`rmdir /S /Q System32`	Operating system commands and other user input that you can type on the command line and press **Enter** to invoke. Items in *italics* should be replaced by actual values.
`C:\System.ini`	Filenames, directory names, paths and package names.
`a.append(b);`	Program source code.
`server.Version`	An inline Java or C++ class name.
`getVersion()`	An inline Java method name.
Shift-N	A combination of keystrokes.
Service View	A label, word or phrase in a GUI window, often clickable.
New->Service	Menu choice.

Documentation Updates

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

- Software version number
- 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://ovweb.external.hp.com/lpe/doc_serv/

Support

Mercury Product Support

You can obtain support information for products formerly produced by Mercury as follows:

- If you work with an HP Software Services Integrator (SVI) partner (www.hp.-com/managementsoftware/svi_partner_list), contact your SVI agent.
- If you have an active HP Software support contract, visit the HP Software Support Web site and use the Self-Solve Knowledge Search to find answers to technical questions.
- For the latest information about support processes and tools available for products formerly produced by Mercury, we encourage you to visit the Mercury Customer Support Web site at: http://support.-mercury.com.
- For the latest information about support processes and tools available for products formerly produced by Systinet, we encourage you to visit the Systinet Online Support Web site at: **http://www.systinet.-com/support/index**.
- If you have additional questions, contact your HP Sales Representative.

HP Software Support

You can visit the HP Software Support Web site at:

www.hp.com/managementsoftware/services

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts

- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To find more information about access levels, go to: www.hp.-com/managementsoftware/access_level

To register for an HP Passport ID, go to: www.managementsoftware.hp.com/passport-registration.html

Part I. SOA Systinet Repository

This part is a reference to the format and structure of the repository.

This part contains the following chapters:

1 Repository Element Formats

The repository is composed of resources. A resource is document or collection. Collections are containers of documents. Collection can also contain collection.

Resource is described by data and metadata. For each resource there is an association with various types of data and metadata.

Each resource in the repository can be represented using a canonical XML format. A REST-based application program interface uses this format. Read more about this API in the Developer Guide.

The subsections that follow describe resources in XML elements of this format starting with the root element resource. The target namespace of the schema for a canonical XML format is `http://systinet.com/2005/05/soa/resource`. The complete schema is given in Resource Serialization Schema on page 31.

Metadata from nature of SOA Systinet can be divided into two groups: general metadata and SOA artifacts. General metadata describe documents in general repository terms such as revision, owner or creator. *SOA artifacts describe documents in terms of Service Oriented Architecture — represented by SOA Definition Model (SDM)* — such as lifecycle stage or relationship. Also *descriptor* or *SDM descriptor* are used in SOA Systinet; they're both synonyms for SOA artifacts.

The types of metadata associated with a resource are summarized in the following table:

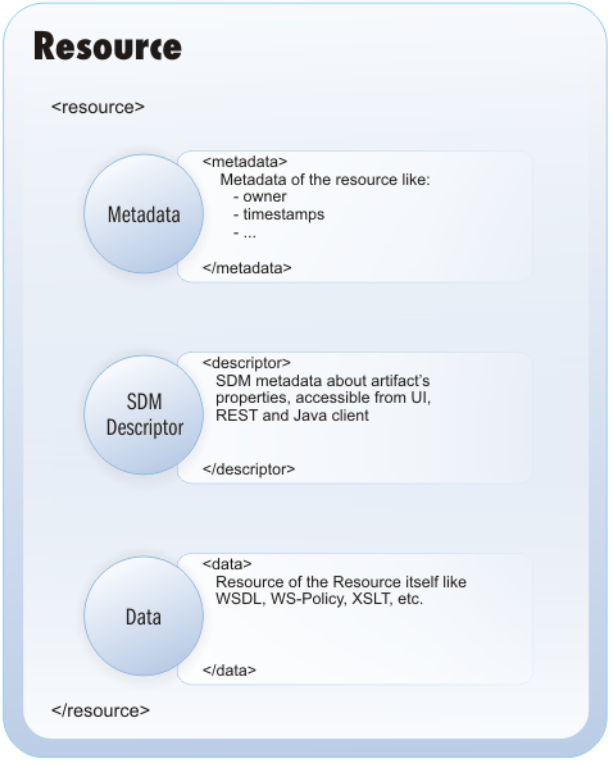
Table 1. Data Associated with Resources

Element	Description
data on page 25	The document content, or if the resource is a collection, a representation listing the resources it contains.
descriptor on page 25	Description of SOA artifacts.
metadata on page 19	Other metadata including: • miscellaneous properties of the resource such as its title • relationships with other documents • revision history • extensions. That is, elements of a different namespace used to implement features defined at a higher level.
acl on page 23	Description of Access Control Lists.

Resource Layout

SDM metadata is available as artifact descriptors via the REST interface. This metadata forms part of the resource representation.

Figure 1. REST Resource Layout.



An artifact descriptor representation of the resource can be obtained using `desc` within the REST interface – the URL has a format like this:

```
http://hostname:port/soa/systinet/platform/rest/repository/businessServiceArtifacts/
FTPBusinessService?desc
```

An XML schema description for the artifact descriptors is available in the directory `PLATFORM_HOME/conf/sdm/xsd`.

The descriptor of the document must be valid for the collection, in which it is located. This means that it is not possible to create a document with descriptor in collection which is not described in SDM or create for example a person artifact in `wsdls` collection.

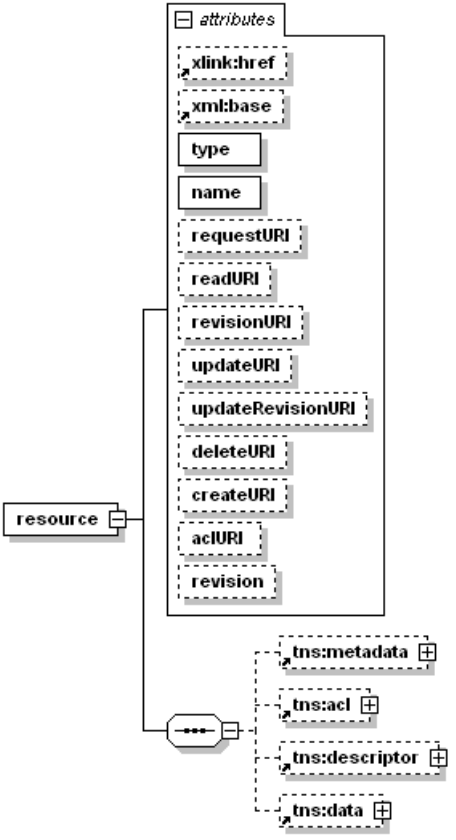
Once the server is started, you may find a low-level technical description of the SDM either in the file `PLATFORM_HOME/conf/sdm/sdmConfig.html` or in the repository:

- http://localhost:8080/soa/systinet/platform/rest/repository/sdm/sdmConfig.html

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

resource

This is the root element of the XML representation of resources.



The high level structure was introduced in the Concept Guide and is reflected in Table 3 on page 19.

Many of the attributes are required http responses, as described in REST Representations in the Developer Guide.

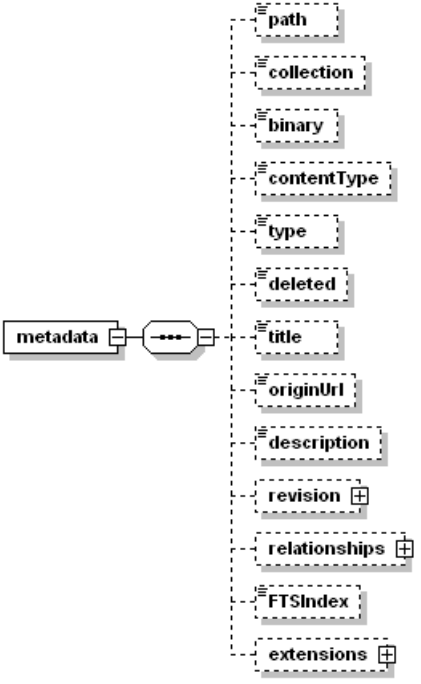
Table 2. Attributes

Name	Description
xlink:href	URL of this resource (without any URL query parameters)
xml:base	XML Base URL, used as base for relative URIs
type	collection, or document for documents
name	Name (without parent collection path, not the same as SDM artifact name)
revision	Revision of the resource
requestURI	The URL of the http request that resulted in this response
createURI	These URLs can be used to perform the corresponding operations on the resource. To run the correct HTTP operation (GET or POST) must be used
readURI	
updateURI	
deleteURI	
undeleteURI	
purgeURI	
deleteRevisionURI	URI for the delete operation, which fails when the last revision is not this revision (optimistic locking)
purgeRevisionURI	URI for the purge operation, which fails when the last revision is not this revision (optimistic locking)
revisionURI	URL to GET this revision of the resource
updateRevisionURI	URI for the update operation, which fails when the last revision is not this revision (optimistic locking)
aclURI	URL to GET the XML http resource serialization with ACL section
viewURI	URL which redirects the browser to the resource's web UI representation (if available)
viewRevisionURI	URL which redirects the browser to the resource's web UI representation (if available) of this revision

Table 3. Content

Element	Description
`acl`	See acl on page 23
`descriptor`	See descriptor on page 25
`data`	See data on page 25
`metadata`	See metadata on page 19

metadata

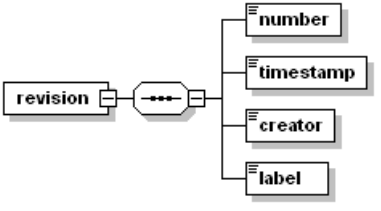


The element `metadata` can contain a number of simple elements representing properties of the resource.

Table 4. Content

Element	Description
`path`	URL of the resource relative to the database path
`collection`	URL of the containing collection
`binary`	Is the resource binary (else its content is XML)
`contentType`	`text/xml` for XML documents, something else for binary documents
`type`	`collection` or `document`
`deleted`	`0` or `1`
`title`	Metadata title of the resource (not the same as the SDM artifact name)
`originUrl`	URL of the original data source. Used for synchronization
`description`	Metadata description of the resource (not the same as the SDM artifact description)
`owner`	Username who owns the resource (the first creator)
`revision`	See revision on page 20
`relationships`	See relationships on page 21
`FTSIndex`	FullText search flag (for collection only). If set, then on the collection a fulltext searched can be run
`cached`	Flag indicating if the resource is cached
`checksum`	Server's resource checksum
`extensions`	See extensions on page 23

revision



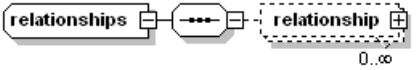
The element `revision` contains the following mandatory elements.

Table 5. Content

Element	Description
`number`	Integer version number
`timestamp`	When the revision was created
`creator`	The user who created the revision
`label`	Text description

For more information see the Revision History section in the HP Systinet User Guide.

relationships

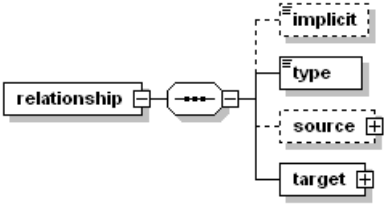


The element `relationships` contains zero or more `relationship` elements.

Table 6. Content

Element	Description
`relationship`	See relationship on page 21

relationship

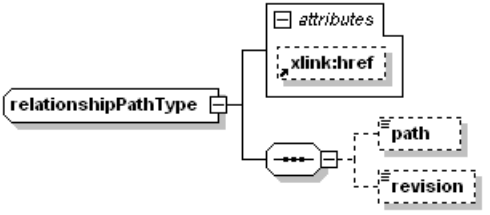


Each element `relationship` contains the following elements.

Table 7. Content

Element	Description
deleted	A flag indicating that the relationship points to a deleted or purged resource
type	Specification of the relationship type in the repository
source	Location of the source document in the repository. See relationshipPathType on page 22
target	Location of the target document in the repository. See relationshipPathType on page 22
extensions	See extensions on page 23

relationshipPathType



This is the element containing the source and target information for relationship elements.

Table 8. Attributes

Name	Description
xlink:href	The relative URI (resolved to xml:base) of the source or target revision in the repository

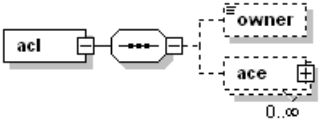
Table 9. Content

Element	Description
`path`	The relative path of the source or target in the repository.
`revision`	An integer revision number (See Resource Revision Identification in the Developer Guide)
`datetime`	ISO8601 datetime value of timeslice (See Resource Revision Identification in the Developer Guide)

extensions

The optional element `extensions` can contain a sequence of elements from any namespace. It is used to meet requirements that the repository does not explicitly cater for.

acl

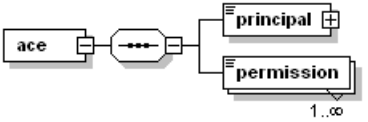


The element `acl` can have an `owner` element and one or more `ace` elements.

Table 10. Content

Element	Description
`owner`	The user who owns the resource
`ace`	See ace on page 24

ace

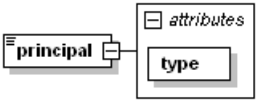


The element `ace` specifies one or more permissions that a user or group has for the resource.

Table 11. Content

Element	Description
`principal`	See principal on page 24
`permission`	One of: • `read` • `write`

principal



The element `principal` contains the name of a user or group.

Table 12. Attributes

Name	Description
`type`	`user` or `group`

descriptor

If the resource is an SOA artifact, a `descriptor` element may contain metadata describing it. This metadata represents the properties of the artifact as specified in Artifacts in the Concept Guide. This schema allows any element content from another namespace because it is designed to be general purpose where possible. However, in SOA Systinet the content will comply to one of the schemas included in the distribution directory `etc/sdm/xsd`.

data

The element `data` contains the content or other representation of the resource. The content is one of the optional elements in Table 14 on page 25.

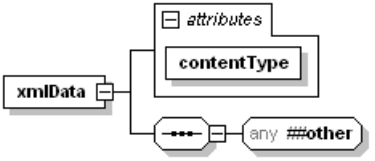
Table 13. Attributes

Name	Description
`representation`	One of: • `list` denotes a collection list • `data` denotes binary data • `xmldata` denotes XML data • `history` denotes history • `FTSResult` denotes a fulltext search result

Table 14. Content

Element	Description
`resource`	For collection list, history and fulltext search representations, it holds a number of resource elements. See resource on page 16
`binaryData`	For binary documents. See binaryData on page 26
`xmlData`	For XML documents. See xmlData on page 26

xmlData

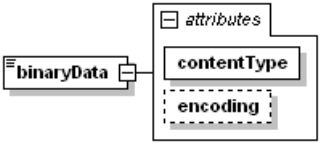


The element `xmlData` is used in `data` to represent the data of an XML document. The content can be any XML element.

Table 15. Attributes

Name	Description
`contentType`	The specific MIME type of this XML document

binaryData

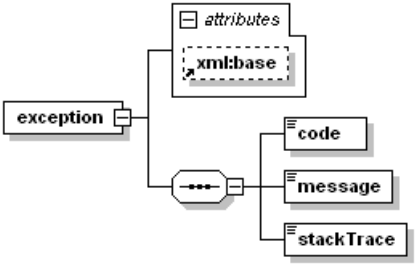


The element `binaryData` is the alternative to `xmlData` and allows for other data formats. If encoding is not set, then the element contains text data.

Table 16. Attributes

Name	Description
`contentType`	The MIME type
`encoding`	Currently the only allowed value is `base64`

exception



The element `exception` is defined in the resource schema for use with the http interface described in REST Interface in the Developer Guide.

Table 17. Attributes

Name	Description
`xml:base`	XML Base URL, used as a base for relative URIs

Table 18. Content

Element	Description
`code`	exception code – string identifying the exception. If this element was empty, that would mean that the exception was not anticipated internally
`message`	Exception message
`stackTrace`	JVM stack trace

2 Collections

A collection is a type of repository resource. There are two kinds of collections:

- Generic repository collections.
- Collections defined by the SDM.

The list of collections can be found in the `PLATFORM_HOME/conf/sdm/sdm.xml` file.

3 Resource Serialization Schema

This schema specifies the XML representation of resources and their metatdata. XQueries act on resources in this format. It is also used by the REST XML communications model described in REST Representations in the Developer Guide.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:rest="http://systinet.com/2005/05/soa/resource"
            xmlns:xml="http://www.w3.org/XML/1998/namespace"
            targetNamespace="http://systinet.com/2005/05/soa/resource"
            elementFormDefault="qualified">

    <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="xml.xsd"/>

    <!-- Serialization of Resource in REST -->
    <xsd:element name="resource">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="rest:metadata" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="rest:acl" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="rest:descriptor" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="rest:data" minOccurs="0" maxOccurs="1"/>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>

            <!-- URI of the resource -->
            <xsd:attribute ref="xlink:href" use="optional"/>
            <!-- XML base -->
            <xsd:attribute ref="xml:base" use="optional"/>
            <!-- type of the resource -->
            <xsd:attribute name="type" type="rest:resourceType" use="required"/>
            <!-- name of the resource -->
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <!-- used request URI -->
            <xsd:attribute name="requestURI" type="xsd:anyURI" use="optional"/>
            <!-- readURI -->
            <xsd:attribute name="readURI" type="xsd:anyURI" use="optional"/>
```

```
        <!-- revisionURI -->
        <xsd:attribute name="revisionURI" type="xsd:anyURI" use="optional"/>
        <!-- updateURI -->
        <xsd:attribute name="updateURI" type="xsd:anyURI" use="optional"/>
        <!-- update revision URI -->
        <xsd:attribute name="updateRevisionURI" type="xsd:anyURI" use="optional"/>
        <!-- deleteURI -->
        <xsd:attribute name="deleteURI" type="xsd:anyURI" use="optional"/>
        <!-- delete Revision URI -->
        <xsd:attribute name="deleteRevisionURI" type="xsd:anyURI" use="optional"/>
        <!-- undeleteURI -->
        <xsd:attribute name="undeleteURI" type="xsd:anyURI" use="optional"/>
        <!-- purge URI-->
        <xsd:attribute name="purgeURI" type="xsd:anyURI" use="optional"/>
        <!-- purge Revision URI-->
        <xsd:attribute name="purgeRevisionURI" type="xsd:anyURI" use="optional"/>
        <!-- view URI -->
        <xsd:attribute name="viewURI" type="xsd:anyURI" use="optional"/>
        <!-- view revision URI -->
        <xsd:attribute name="viewRevisionURI" type="xsd:anyURI" use="optional"/>
        <!-- createURI (for collection only)-->
        <xsd:attribute name="createURI" type="xsd:anyURI" use="optional"/>
        <!-- acl uri -->
        <xsd:attribute name="aclURI" type="xsd:anyURI" use="optional"/>
        <!-- revision -->
        <xsd:attribute name="revision" type="xsd:int" use="optional"/>

        <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="metadata">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="path" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="collection" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="binary" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="contentType" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="type" type="rest:resourceType" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="deleted" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="title" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="originUrl" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="owner" type="xsd:string" minOccurs="0" maxOccurs="1"/>

            <xsd:element name="revision" minOccurs="0" maxOccurs="1">
                <xsd:complexType>
```

```
                        <xsd:sequence>
                            <xsd:element name="number" type="xsd:int" minOccurs="1" maxOccurs="1"/>
                           <xsd:element name="timestamp" type="xsd:string" minOccurs="1" maxOccurs="1"/>

                            <xsd:element name="creator" type="xsd:string" minOccurs="1" maxOccurs="1"/>

                            <xsd:element name="label" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                            <xsd:element name="last" type="xsd:int" nillable="true" minOccurs="0"
maxOccurs="1"/>
                            <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>

               <xsd:element name="relationships" type="rest:relationships" minOccurs="0" maxOccurs="1"/>

                <xsd:element name="FTSIndex" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="cached" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="checksum" type="xsd:long" minOccurs="0" maxOccurs="1"/>

                <xsd:element ref="rest:extensions" minOccurs="0" maxOccurs="1"/>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="relationships">
        <xsd:sequence>
            <xsd:element name="relationship" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="deleted" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                        <xsd:element name="source" type="rest:relationshipPathType" minOccurs="0"
maxOccurs="1"/>
                        <xsd:element name="target" type="rest:relationshipPathType" minOccurs="1"
maxOccurs="1"/>
                        <xsd:element ref="rest:extensions" minOccurs="0" maxOccurs="1"/>
                        <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

```
<xsd:complexType name="relationshipPathType">
    <xsd:sequence>
        <xsd:element name="path" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="revision" type="xsd:int" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="datetime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
        <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="xlink:href" use="optional"/>
</xsd:complexType>

<xsd:element name="extensions">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="descriptor">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="data">
    <xsd:complexType>
        <xsd:sequence>
            <!-- for representation of listing of collections-->
            <xsd:element ref="rest:resource" minOccurs="0" maxOccurs="unbounded"/>
            <!-- for representation of binary data (see encoding and contentType)-->
            <xsd:element name="binaryData" minOccurs="0" maxOccurs="1">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:attribute name="contentType" type="xsd:string" use="required"/>
                            <xsd:attribute name="encoding" use="optional">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="base64"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:attribute>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
```

```
                </xsd:element>
                <!-- for representation of XML data (see contentType on "content")-->
                <xsd:element name="xmlData" minOccurs="0" maxOccurs="1">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:any processContents="skip" namespace="##other" minOccurs="1"
maxOccurs="1"/>
                        </xsd:sequence>
                        <xsd:attribute name="contentType" type="xsd:string" use="optional"/>
                        <xsd:attribute ref="xml:base" use="optional"/>
                    </xsd:complexType>
                </xsd:element>
                <!-- for representation of XQuery result -->
                <xsd:element name="xqueryResult" minOccurs="0" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:choice>
                          <xsd:element name="textResult" type="xsd:string" minOccurs="1" maxOccurs="1"/>

                            <xsd:element name="nodeResult" minOccurs="1" maxOccurs="1">
                                <xsd:complexType>
                                    <xsd:sequence>
                                      <xsd:any processContents="skip" namespace="##other" minOccurs="1"
 maxOccurs="1"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:choice>
                    </xsd:complexType>
                </xsd:element>

                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>

            <xsd:attribute name="representation">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="list"/>
                        <!-- collection's sub-resources -->
                        <xsd:enumeration value="data"/>
                        <!-- binary document's data -->
                        <xsd:enumeration value="xmldata"/>
                        <!-- xml document's data-->
                        <xsd:enumeration value="history"/>
                        <!-- history of document -->
                        <xsd:enumeration value="xqueryResult"/>
                        <!-- xquery result -->
                        <xsd:enumeration value="FTSResult"/>
```

```
                    <!-- full text search result -->
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>

        <!-- page, pageSize and totalItemCount: used only if representation is list and paging is
turned on -->
        <xsd:attribute name="page" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedInt">
                    <xsd:minInclusive value="1"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="pageSize" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedInt">
                    <xsd:minInclusive value="1"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="totalItemCount" type="xsd:unsignedInt" use="optional"/>

    </xsd:complexType>
</xsd:element>

<xsd:simpleType name="resourceType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="collection"/>
        <xsd:enumeration value="document"/>
        <xsd:enumeration value="service"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="permission">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="read"/>
        <xsd:enumeration value="write"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="acl">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="owner" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="ace" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
```

```
                        <xsd:sequence>
                            <xsd:element name="principal" minOccurs="1" maxOccurs="1">
                                <xsd:complexType>
                                    <xsd:simpleContent>
                                        <xsd:extension base="xsd:string">
                                            <xsd:attribute name="type" use="required">
                                                <xsd:simpleType>
                                                    <xsd:restriction base="xsd:string">
                                                        <xsd:enumeration value="user"/>
                                                        <xsd:enumeration value="group"/>
                                                    </xsd:restriction>
                                                </xsd:simpleType>
                                            </xsd:attribute>
                                        </xsd:extension>
                                    </xsd:simpleContent>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="permission" type="rest:permission" minOccurs="1"
maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="effectivePermissions" minOccurs="0" maxOccurs="1">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="permission" type="rest:permission" minOccurs="1"
maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <!-- Serialization of exception in REST -->
    <xsd:element name="exception">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="code" type="xsd:QName"/>
                <xsd:element name="message" type="xsd:string"/>
                <xsd:element name="stackTrace" type="xsd:string"/>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute ref="xml:base" use="optional"/>
            <!-- XML base -->
        </xsd:complexType>
```

```
    </xsd:element>
</xsd:schema>
```

Part II. SOA Definition Model

This part describes the structure and content of the SDM model used in SOA Systinet. It contains the following chapters:

SDM Reference on page 41. A guide to the content and hierarchy of the SDM model.

Taxonomy Schema on page 91. A guide to the format of significant taxonomies in SOA Systinet.

Policy Schema on page 97. The schema structure on which HP SOA Systinet Policy Manager bases technical policies.

Assertion Schema on page 99The schema of assertions and collections of assertions. Assertions are the atomic level of policies created by HP SOA Systinet Policy Manager.

4 SDM Reference

This chapter is a reference to the SDM model. It contains the following sections:

Artifacts Taxonomy

The taxonomy of *artifact* types follows. Each category of an *abstract artifact type* is formatted with *emphasis*. Abstract artifact types cannot be instantiated and are used for classification. The lowest level categories are all of *instantiable artifact types*, for which instances can exist in the repository.

Each category links to a section describing the corresponding artifact type:

- *Artifact*
  - *Content*
    - *Categorization*
      - Taxonomy
    - *Contact*
      - Organizational Unit
      - Person

- Documentation
- *Interface*
  - WSDL
- Report
- *Schema*
  - XML Schema
  - DTD
- *Search And Transform*
  - XQuery
  - XSLT
  - Stored Search
- UDDI Entity

- *SOA*
  - *Agreement*
    - Contract
    - Request
    - SLO
  - Business Service

- Endpoint
- *Implementation*
  - Web Application
  - SOAP Service
  - HTTP Service
- *Policy*
  - WS-Policy
  - Assertion
  - Business Policy

- *System*
  - *Integration*
    - UDDI Registry
      - UDDI Channel
    - BAC Server
  - Task
  - *Tool*
    - Impact Tool
    - Job Tool

- Reporting Tool
- Sync Tool
- Aoi Preparation Tool
- Policy Compliance Tool

Artifact Types

This section provides an overview of *artifact* types. *Properties* are described for *instantiable* artifact types but not for *abstract* types. See Artifacts Taxonomy on page 41.

Agreement

Represents agreements artifacts

This artifact type is abstract and serves as a super-category.

Aoi Preparation Tool

Aoi Preparation Tool

This artifact type is not abstract, so instances can exist.

Artifact

Represents an SOA artifact

This artifact type is abstract and serves as a super-category.

Assertion

A definition of a ws-policy assertion.

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 1. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
imports	Imports	0	∞	No	Lists artifact(s) that are imported by this artifact
importedBy	Imported by	0	∞	No	List resources that import/include this artifact
associatedApplication	Associated Application	0	1	No	Application that maintains this artifact
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact
assertionQName	Assertion QName	1	1	No	Template QName of Assertion artifact
applicableSourceMimetype	Applicable source MIME type	0	∞	No	Applicable source MIME type
applicableSourceQName	Applicable source QName	0	∞	No	Applicable source QName
assertionClassification	Assertion Classification	0	1	No	Assertion Classification

BAC Server

Represents a BAC Server

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 2. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
baseUrl	Base URL	0	1	No	Base (URL) of the installed Server
username	Username	0	1	No	Username
encryptedPassword	Password	0	1	No	Password
runtimePolicy	Runtime Policy	0	∞	No	Reference to run-time policy of this artifact
resourceOfUsagePlan	Usage Plan Resource	0	∞	No	Reference to the usage plan

Business Policy

Document in the format described by the WS-Policy attachment

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 3. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
imports	Imports	0	∞	No	Lists artifact(s) that are imported by this artifact
importedBy	Imported by	0	∞	No	List resources that import/include this artifact
associatedApplication	Associated Application	0	1	No	Application that maintains this artifact
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact
artifactTypeSelector	Artifact Type Selector	0	1	No	Artifact Type Selector for Business Policy
includedPropertySelector	Included Property Selector	0	∞	No	Included Property Selector for Business Policy
excludedPropertySelector	Excluded Property Selector	0	∞	No	Excluded Property Selector for Business Policy
includedArtifactSelector	Included Artifact Selector	0	∞	No	Included Artifact Selector for Business Policy
excludedArtifactSelector	Excluded Artifact Selector	0	∞	No	Excluded Artifact Selector for Business Policy

Business Service

Service described in business terms that can be implemented using one of more Web Services or other Implementations. Contracts enable re-use of Business Services

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86
- Table 33 on page 85
- Table 32 on page 85
- Table 35 on page 87

This artifact type has the following specific properties.

Table 4. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
service	Implementation	0	∞	No	Lists implementations that form this Business Service
providedBy	Provided by	0	∞	No	Refers to the Contact who provides this artifact. Optional usetype describes how the artifact is provided - 'responsible', 'developer', 'administrator', etc.
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact
keyword	Keyword	0	∞	No	Keyword associated with the artifact

Categorization

Represents an artifact used for categorization

This artifact type is abstract and serves as a super-category.

Contact

Describes a contact

This artifact type is abstract and serves as a super-category.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 5. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
provides	Provides	0	∞	No	Artifacts provided by this Contact. Optional usetype describes how the artifact is provided - 'responsible', 'developer', 'administrator', 'project manager', etc.
contactRole	Contact Role	0	1	No	Role of the contact, such as 'business expert', 'developer' or 'architect'
geographicalLocation	Geographical Location	0	∞	No	Geographical Location
languageCode	Language Code	0	1	No	Preferred language

Content

SOA related documents

This artifact type is abstract and serves as a super-category.

Contract

Represents a contract

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 6. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
consumerContractor	Consumer	0	1	No	Reference to consumer contact
usagePlan	Usage Plan	0	1	No	Reference to usage plan
slo	SLO	0	∞	No	Contract SLO
providerContractor	Provider	0	1	No	Reference to provider
acceptedRequest	Accepted Request	0	1	No	Reference from contract to accepted requests
rejectedRequest	Rejected Request	0	1	No	Reference from contract to rejected requests
revokedRequest	Revoked Request	0	1	No	Reference from contract to revoked requests
contractAcceptedBy	Accepted Contract of	0	1	No	Reference from accepting provider
rejectedContractOf	Rejected Contract of	0	1	No	Reference from rejecting provider
revokedContractOf	Revoked Contract of	0	1	No	Reference from revoking provider
obsoletedContract	Obsoleted Contract	0	∞	No	Reference to obsoleted contract(s)
obsoletedByContract	Obsoleted by Contract	0	1	No	Contract that changed this contract
obsoletedByRequest	Obsoleted by Request	0	1	No	Request that changed this contract
contractState	Contract State	0	1	No	State of contract

Documentation

Information about a service or related items

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 7. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
documentationOf	Documentation of	0	∞	No	Artifact that this artifact documents

DTD

DTD document

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 8. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
importedDtdDocument	Imported DTD	0	∞	No	Reference to DTD document imported by this artifact. Public/System external entity declaration can be differentiated using useType attribute
dtdImportedBy	Imported by	0	∞	No	List resources that import/include this DTD artifact
schemaInstanceProducedBy	Instance Produced by	0	∞	No	Reference to a transformation (XSLT or XQuery) that produces XML documents adhering to this schema
schemaInstanceConsumedBy	Instance Consumed by	0	∞	No	Reference to a transformation (XSLT or XQuery) that produces XML documents conforming to this schema

Endpoint

Endpoint of implementation.

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 9. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
accessPoint	Endpoint Address	0	1	No	Endpoint(s) used to access the service
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact
instanceDetail	Binding Template Instance Detail	0	∞	No	Binding Template instance detail. Used by UDDI integration.
endpointOf	Implementation of endpoint	0	∞	No	Reference to the endpoint's implementation.
portName	WSDL Port Name	0	1	No	WSDL Port Name

HTTP Service

Any HTTP Service is in fact a Web Service that does not conform to the existing WS-Standards. It usually provides HTTP GET access for ease of use, hence avoiding the need for sophisticated SOAP frameworks

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 10. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
definition	Definition	0	∞	No	References the primary repository document that defines this artifact, such as a WSDL, XSD, UML or IDL resource

Impact Tool

Impact Tool provides the ability to list all artifacts related to a specific artifact

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 11. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
impactType	Impact Type	0	1	No	Impact type specifies how other artifacts relate to this artifact

Implementation

Implementation of a service

This artifact type is abstract and serves as a super-category.

This artifact type has properties in the following groups:

- Table 35 on page 87
- Table 34 on page 86

This artifact type has the following specific properties.

Table 12. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact
inBusinessService	In Business Service	0	∞	No	The Business Service artifact represents a logical service and contains descriptive information in business terms. Technical information about the Business Service is found in the referenced Implementation artifacts
providedBy	Provided by	0	∞	No	Refers to the Contact who provides this artifact. Optional usetype describes how the artifact is provided - 'responsible', 'developer', 'administrator', etc.
accessPoint	Endpoint Address	0	∞	No	Endpoint(s) used to access the service
endpoint	Endpoint	0	∞	No	Reference to the endpoint.
runtimePolicy	Runtime Policy	0	∞	No	Reference to run-time policy of this artifact
resourceOfUsagePlan	Usage Plan Resource	0	∞	No	Reference to the usage plan

Integration

Registries used for governance

This artifact type is abstract and serves as a super-category.

Interface

Metadatum representing a Web Service

This artifact type is abstract and serves as a super-category.

Job Tool

Starts a simple job

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 13. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
jobClassId	Job Implementation Class ID	0	1	No	Class id of class implementing a job

Organizational Unit

Represents an organizational unit

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 32 on page 85

This artifact type has the following specific properties.

Table 14. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
loginName	Login Name	0	1	No	Artifact login name
address	Address	0	1	No	Surface Address
email	Email	0	∞	No	Email Address
phone	Phone	0	∞	No	Phone number
instantMessenger	Instant Messenger	0	∞	No	Instant messenger ID
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact

Person

Represents a person

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 32 on page 85

This artifact type has the following specific properties.

Table 15. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
loginName	Login Name	0	1	No	Artifact login name
address	Address	0	1	No	Surface Address
email	Email	0	∞	No	Email Address
phone	Phone	0	∞	No	Phone number
instantMessenger	Instant Messenger	0	∞	No	Instant messenger ID

Policy

Describes a policy

This artifact type is abstract and serves as a super-category.

Policy Compliance Tool

Policy Compliance Tool

This artifact type is not abstract, so instances can exist.

Report

Report is the result of a tool or task execution

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 16. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
created	Created	0	1	No	Date/time of report creation
reportType	Report Type	0	1	No	Type of the report, such as 'Report' or 'Index Report'
reportCategory	Report Category	0	1	No	Type of the report, such as 'compliance', 'validity', 'integrity', etc.
reportStatus	Status	0	1	No	Status of the report, such as 'Complete' or 'Not complete'
reportResultCode	Result Code	0	1	No	Result code of the report, such as 'OK' or 'Error'
reports	Reports	0	∞	No	References reports that are collected by index report
indexReport	Index Report	0	1	No	References index report that collects other reports created during processing of input
reportRendererId	Report Renderer Id	0	1	No	Identification of report artifact content renderer
closed	Closed	0	1	No	Marks index report artifact as closed. All associated reports have already been created
associatedApplication	Associated Application	0	1	No	Application that maintains this artifact
task	Task	0	1	No	References the task that created this report
sourceArtifact	Source Artifact	0	1	No	Artifact processed by this report
sourceURI	Source URI	0	1	No	URI of resource processed by this report

ID	Label	Cardinality		Read Only	Description
		Min	Max		
errorSubreports	Error Subreports	0	1	No	Specify number of subreports with result code Error
okSubreports	OK Subreports	0	1	No	Specify number of subreports with result code OK
undefinedSubreports	Undefined Subreports	0	1	No	Specify number of subreports with result code Undefined
reviewSubreports	Review Subreports	0	1	No	Specify number of subreports with result code Review
notApplicableSubreports	Not Applicable Subreports	0	1	No	Specify number of subreports with result code Not applicable
businessPolicyArtifact	Business Policy artifact	0	1	No	Business policy used to create this report
technicalPolicyArtifact	Technical Policy artifact	0	1	No	Technical policy used to create this report

Reporting Tool

Posts a document to a specified URL to execute build of a report

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 17. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
reportingUrl	Target URI	0	1	No	Location (URI) where a reporting request will be posted
reportingRequestContentType	Request Content Type	0	1	No	Value of the Content-Type HTTP header of the reporting request
reportingRequestContent	Request Content	0	1	No	Content to be posted

Request

Represents a contract request

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 18. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
consumer	Consumer	0	1	No	Reference to consumer contact
requestUsagePlan	Usage Plan	0	1	No	Reference to usage plan(s)
requestSlo	SLO	0	∞	No	Request SLO
provider	Provider	0	1	No	Reference to provider
acceptedRequestOfContract	Accepted Request of	0	1	No	Reference from contract to accepted request
rejectedRequestOfContract	Rejected Request of	0	1	No	Reference from contract to rejected request
revokedRequestOfContract	Revoked Request of	0	1	No	Reference from contract to revoked request
contractToObsolete	Contract to Obsolete	0	∞	No	Reference to contract(s) to be obsoleted

Schema

Represents a schema artifact of some type

This artifact type is abstract and serves as a super-category.

Search And Transform

Document of some type defining a search, query or transformation

This artifact type is abstract and serves as a super-category.

SLO

Represents a Service Level Objective (SLO)

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 19. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
artifactSloOf	SLO of	0	∞	No	SLO of Artifact
requestSloOf	SLO of	0	∞	No	SLO of Request
sloOf	SLO of	0	∞	No	SLO of Contract
sloOfPlan	Usage Plan	0	∞	No	Reference from usage plan
businessImpact	Business Impact	0	1	No	Impact on the business in the case of an outage - HIGH, MEDIUM or LOW
developmentEnvironmentAvailability	Development Environment Availability	0	1	No	Expected date/time availability of development environment
uatEnvironmentAvailability	UAT Environment Availability	0	1	No	Expected date/time availability of User Acceptance environment (UAT)
productionEnvironmentAvailability	Production Environment Availability	0	1	No	Expected date/time availability of production environment
disasterRecoveryEnvironmentAvailability	Disaster Recovery Environment Availability	0	1	No	Expected date/time availability of disaster recovery environment
hourOfConsumerOperation	Hours of Consumer Operation	0	∞	No	Day(s) of the week and hours during the day when the consumer is expected to operate. Property value format is WEEKDAY/FROM HOUR/TO HOUR for example monday/9:00/18:00

ID	Label	Cardinality		Read Only	Description
		Min	Max		
hourOfProviderOperation	Hours of Provider Operation	0	∞	No	Day(s) of the week and hours during the day when the provider is expected to operate. Property value format is WEEKDAY/FROM HOUR/TO HOUR for example monday/9:00/18:00
hourOfServiceOperation	Hours of Service Operation	0	∞	No	Day(s) of the week and hours during the day when the service is expected to operate. Property value format is WEEKDAY/FROM HOUR/TO HOUR for example monday/9:00/18:00
timingRequired	Timing Required	0	1	No	Require message delivery timing capture
guaranteedDelivery	Guaranteed Delivery	0	1	No	Require guaranteed delivery
filteringRulesDocumentation	Filtering Rules	0	∞	No	Links to various documents describing filtering rules
expectedMessagesPerDay	Expected Messages Per Day	0	1	No	Expected average volume of message traffic per day - number of messages
maximumMessagesPerDay	Maximum Messages Per Day	0	1	No	Maximum volume of message traffic per day - number of messages
maximumMessageSize	Maximum Message Size [kB]	0	1	No	Maximum expected message size
volumeDistributionDocumentation	Volume Distribution	0	1	No	Document describing volume distribution - sizes and frequency of messages (distribution function)

ID	Label	Cardinality		Read Only	Description
		Min	Max		
failureProcedureDocumentation	Failure Procedure	0	1	No	Step-by-step procedure to perform in the event of a service failure
escalationProcedureDocumentation	Escalation Procedure	0	1	No	Order of precedence for notification of support personnel in the case of a service outage
acceptableDowntime	Acceptable Downtime [hours]	0	1	No	Number of hours that the service can be down before there is a severe business impact
dependsOnArtifact	Depends on Artifact	0	∞	No	Model level dependency. Used by impact management tools
hasDependingArtifact	Depending Artifact	0	∞	No	Lists model artifacts dependant on this artifact. Used by impact management

SOA

Intangible artifact of a type significant at the architectural level

This artifact type is abstract and serves as a super-category.

SOAP Service

Web service is a software module whose functionality is accessible using the SOAP protocol. Web services are HTTP based and usually described using WSDL

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 20. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
serviceName	WSDL Service Name	0	1	No	WSDL Service Name
serviceNamespace	WSDL Service Namespace	0	1	No	WSDL Service Namespace
definition	Definition	0	∞	No	References the primary repository document that defines this artifact, such as a WSDL, XSD, UML or IDL resource
wsdlPortRelationship	WSDL Port	0	∞	No	Reference to WSDL port

Stored Search

Representation of criteria for a search

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 21. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
collectionName	Collection Name	0	1	No	Name of a collection
xqueryDocument	XQuery	0	1	No	Reference to the XQuery

Sync Tool

Provides resource synchronization

This artifact type is not abstract, so instances can exist.

System

System Artifacts

This artifact type is abstract and serves as a super-category.

Task

Task is a tool prepared for execution. It can be scheduled for automatic execution

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 22. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
tool	Tool	0	1	No	Reference to the tool associated with the task or report
selector	Selector	0	1	No	Selector of task
scheduled	Scheduled	0	1	No	Task schedule information
taskReports	Reports	0	∞	No	References index reports that generated by running the task

Taxonomy

UDDI compatible taxonomy definition

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 23. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
productionStage	Lifecycle Status	0	1	No	Stage of the service lifecycle (development, production etc.)
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
hierarchical	Hierarchical	0	1	No	Marks the taxonomy as hierarchical. If this property is false, then it is flat
checked	Checked	0	1	No	Marks a taxonomy as checked

Tool

Represents a tool

This artifact type is abstract and serves as a super-category.

UDDI Channel

UDDI Registry account used for synchronization

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 24. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
username	Username	0	1	No	Username
password	Password	0	1	No	Password
uddiRegistry	UDDI Registry	0	1	No	Reference to UDDI Registry artifact
runtimePolicy	Runtime Policy	0	∞	No	Reference to run-time policy of this artifact
resourceOfUsagePlan	Usage Plan Resource	0	∞	No	Reference to the usage plan

UDDI Entity

UDDI entity - Business Entity, Business Service, Binding Template or TModel

This artifact type is not abstract, so instances can exist.

This artifact type has the following specific properties.

Table 25. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
uddiEntityType	UDDI Entity Type	0	1	No	Type of UDDI entity
uddiEntityKey	UDDI Entity Key	0	1	No	UUID key of UDDI entity
uddiRegistryArtifact	UDDI Registry Artifact	0	1	No	UDDI Registry Artifact that has this UDDI entity in custody
crc	CRC	0	1	No	Cyclic Redundancy Code
uddiImport	UDDI Import Flag	0	1	No	Flag that indicates that the UDDI artifact was created by import from registry
defines	Definition	0	∞	No	Reference to the artifact that is defined by this UDDI entity

UDDI Registry

Represents a UDDI Registry

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 26. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
inquiryUrl	Inquiry URL	0	1	No	Location (URL) of UDDI Inquiry interface
publishingUrl	Publishing URL	0	1	No	Location (URL) of UDDI Publishing interface
uddiApiVersion	UDDI API Version	1	1	No	UDDI API version (v2,v3,..)
securityUrl	Security URL	0	1	No	Location (URL) of UDDI Security interface
taxonomyUrl	Taxonomy URL	0	1	No	Location (URL) of Systinet UDDI Taxonomy interface
fullVersion	Full Version	0	1	No	Full asset version. For example the version of a UDDI Registry artifact might be '3' and its full version might be 'Systinet Registry 5.5'
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
uddiChannel	UDDI Channel	0	∞	No	Reference to UDDI Channel artifact
technicalDetail	Technical Detail	0	∞	No	Technical details represented as name/value pairs
systinetRegistry	Systinet's Registry	0	1	No	This Registry is Systinet's Registry
allowImport	Allow Import	0	1	No	Allow Import
allowExport	Allow Export	0	1	No	Allow Export

ID	Label	Cardinality		Read Only	Description
		Min	Max		
webAdminConsoleUrl	Web Admin Console URL	0	1	No	Location (URL) of UDDI Web Admin Console
webUserConsoleUrl	Web User Console URL	0	1	No	Location (URL) of UDDI Web User Console
runtimePolicy	Runtime Policy	0	∞	No	Reference to run-time policy of this artifact
resourceOfUsagePlan	Usage Plan Resource	0	∞	No	Reference to the usage plan

Web Application

Web application (or web interface) is also a service. While it is accessible only by a human, it uses other services (soap services, databases) to function

This artifact type is not abstract, so instances can exist.

WSDL

Web Services Description Language document describing a Web Service

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 27. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
targetNamespace	Target Namespace	0	1	No	Target namespace of a WSDL/XSD document
importedWsdlDocument	Imported WSDL	0	∞	No	References a WSDL repository document imported with this artifact
wsdlImportedBy	Imported by	0	∞	No	List resources that import/include this WSDL artifact
importedXsdDocument	Imported XSD	0	∞	No	Reference to the XML Schema document imported by this artifact. Imports and includes should be classified using useType attribute ('import', 'include', 'externalRef', 'parentRef', etc.)
definitionOf	Definition of	0	∞	No	Artifacts that this artifact defines
wsdlServiceUsedBy	Service Used by	0	∞	No	List of references to a service of this WSDL

ID	Label	Cardinality		Read Only	Description
		Min	Max		
wsdlPortUsedBy	Port Used by	0	∞	No	List of references to a port of this WSDL

WS-Policy

Document in the format described by the WS-Policy specification

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 28. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
policyType	Policy type	0	1	No	Type of policy, such as 'run-time' or 'design-time'
imports	Imports	0	∞	No	Lists artifact(s) that are imported by this artifact
importedBy	Imported by	0	∞	No	List resources that import/include this artifact
designTimePolicyOf	Design-time Policy of	0	∞	No	List of artifact(s) for which this WS-Policy serves as design-time policy
runtimePolicyOf	Runtime Policy of	0	∞	No	Artifact(s) for which this WS-Policy is the run-time policy
associatedApplication	Associated Application	0	1	No	Application that maintains this artifact
uddiDefinition	UDDI Definition	0	∞	No	Reference to the UDDI Entity artifact that defines this artifact

XML Schema

XML Schema document

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 29. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
targetNamespace	Target Namespace	0	1	No	Target namespace of a WSDL/XSD document
importedXsdDocument	Imported XSD	0	∞	No	Reference to the XML Schema document imported by this artifact. Imports and includes should be classified using useType attribute ('import', 'include', 'externalRef', 'parentRef', etc.)
importedDtdDocument	Imported DTD	0	∞	No	Reference to DTD document imported by this artifact. Public/System external entity declaration can be differentiated using useType attribute
xsdImportedBy	Imported by	0	∞	No	List resources that import/include this XML Schema artifact

ID	Label	Cardinality		Read Only	Description
		Min	Max		
schemaInstanceProducedBy	Instance Produced by	0	∞	No	Reference to a transformation (XSLT or XQuery) that produces XML documents adhering to this schema
schemaInstanceConsumedBy	Instance Consumed by	0	∞	No	Reference to a transformation (XSLT or XQuery) that produces XML documents conforming to this schema

XQuery

XQuery Document

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 30. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
inputSchemaDocument	Input Schema	0	∞	No	Reference to schema (XSD/Relax/DTD) document that can be processed by this artifact
xqueryInputType	Input Type	0	1	No	Type of the XQuery input
xqueryInputParameterType	Input Parameter Type	0	∞	No	XQuery input parameter types. The name attribute is used for the parameter name and the value attribute is used for the type, such as 'string', 'integer' or 'node'. See 'XQuery Input Output Types' taxonomy
xqueryOutputType	Output Type	0	∞	No	Type of the XQuery output
xqueryOutputMultipleCardinality	Returns Multiple Results	0	1	No	If this property is true, then the XQuery query returns a sequence of results
outputSchemaDocument	Output Schema	0	∞	No	Reference to schema (XSD/Relax/DTD) document that can be processed by this artifact

XSLT

XSLT Transformation Document

This artifact type is not abstract, so instances can exist.

This artifact type has properties in the following groups:

- Table 34 on page 86

This artifact type has the following specific properties.

Table 31. Specific Properties

ID	Label	Cardinality		Read Only	Description
		Min	Max		
originUrl	Original URL	0	1	No	Location (URL) of the original resource which was imported into the repository
isCached	Cached	0	1	No	Marks a resource that is just cached in the repository. For example WSDL
version	Version	0	1	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to
xslOutputMethod	XSLT Output Method	0	1	No	XSLT output methods - 'XML', 'HTML' or 'text'
importedXslDocument	Imported XSL	0	∞	No	Reference to the XSL document imported by this artifact. Imports and includes should be classified using useType attribute ('import', 'include', 'externalRef', 'parentRef', etc.)
inputSchemaDocument	Input Schema	0	∞	No	Reference to schema (XSD/Relax/DTD) document that can be processed by this artifact
outputSchemaDocument	Output Schema	0	∞	No	Reference to schema (XSD/Relax/DTD) document that can be processed by this artifact

Property Groups

The following subsections describe groups of properties re-used in the definition of artifact types.

Table 32. Contract Consumer Property Group

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
consumptionRequest	Request	0	∞	no	No	Reference from request
consumptionContract	Contract	0	∞	no	No	Reference to contract

Table 33. Contract Property Group

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
artifactSlo	SLO	0	∞	no	No	Artifact SLO
providerUsagePlan	Usage Plan	0	∞	no	No	Reference to usage plan(s)
provisionRequest	Provided Request	0	∞	no	No	Reference from request
provisionContract	Provision Contract	0	∞	no	No	Reference to contract
acceptedContract	Accepted Contract	0	∞	no	No	Reference to accepted contract
rejectedContract	Rejected Contract	0	∞	no	No	Reference to rejected contract
revokedContract	Revoked Contract	0	∞	no	No	Reference to revoked contract
readyForConsumption	Enable Consumption Requests	0	1	no	No	Marks artifact as ready for consumption
stakeholderEmail	Stakeholder Email	0	∞	no	No	Stakeholder email address

Table 34. Model Property Group

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
documentation	Documentation	0	∞	no	No	Links to various documentation and descriptive documents, such as application system flows, sequence diagrams, use-case diagrams, deployment diagrams, network diagrams, DR plans, CPM documents, etc.
designTimePolicy	Design-time Policy	0	∞	no	No	Reference to design-time policy of this artifact

Table 35. Service Property Group

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
criticality	Failure Impact	0	1	no	No	Rating of the consequences of service downtime
productionStage	Lifecycle Status	0	1	no	No	Stage of the service lifecycle (development, production etc.)
version	Version	0	1	no	No	Version of the resource. Various versioning schemes can be used to describe, for example, the version of WSDL or UDDI that artifacts conform to

Table 36. System Property Group

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
name	Name	1	∞	no	No	Name which should identify the artifact
description	Description	0	∞	no	No	Description of the artifact, what it is used for etc.
revision	Revision	0	1	no	No	Artifact revision
lastRevision	Last Revision	0	1	no	No	Artifact last revision
artifactType	Artifact Type	0	∞	no	No	Categorization of the artifact using Artifact type taxonomy
documentRelationship	Other Relationship	0	∞	no	No	Relationship that has no mapping to an SDM property. The useType attribute specifies the type of relationship (typically a QName) that should be created

ID	Label	Cardinality		Displayed	Read Only	Description
		Min	Max			
incomingDocumentRelationship	Incoming Relationship	0	∞	no	No	Representation of an incoming relationship that has no mapping to a property. In this case the useType attribute is set to the relationship identifier (typically a QName)
categoryBag	Categories	0	1	no	No	Arbitrary classification
identifierBag	Identifier Bag	0	1	no	No	Arbitrary Identification

Property Types

The following table lists predefined property types.

Type	Description
address	Contact address
boolean	Boolean property
category	Used to assign several categories from a taxonomy to the artifact
categoryBag	Used to categorize the artifact using any taxonomies
dailyInterval	Daily Interval
dateTime	Date/time
documentRelationship	Used to reference other artifacts etc.
encryptedPassword	Encrypted Password
identifierBag	Used to identify the artifact using any taxonomies

Type	Description
instanceDetail	Property type used by UDDI integration.
integer	Integer number
nameUrlPair	Name-URL pair property
nameValuePair	(Name,Value) pair property
plainText	Text
portDocumentRelationship	Port-document relationship
qnamedDocumentRelationship	Used to reference parts of WSDLs, WS-Policies, etc.
scheduled	Property type used to display scheduling information for task
selector	Property type used to display selector for a task
text	Short text property displayed in an text input form field
textarea	Long text property displayed in a text area form field
xqueryParameter	Property type used to display parameters for executing an XQuery

5 Taxonomy Schema

Each *taxonomy* is represented as an XML document with root element `taxonomy`, as defined in the schema http://www.systinet.com/doc/wasp_uddi-55/wsdl/taxonomy.xsd.

The following sections each describe an element of the schema significant in the context of the *SDM*. Generally, the format of each section is:

1. Diagram;
2. Attribute specifications in the context of the SDM;
3. Additional detail if any;

In the current release these details are of limited significance to users, but will be increasingly of interest to architects and developers who wish to extend the SDM.

Other details of the schema are of some interest for the purpose of governance and the implementation of policies. The schema defines elements used for purposes, such as listing and referencing taxonomies on APIs. For more information see HP SOA Systinet Registry documentation [http://www.systinet.com/doc/wasp_uddi-55/html/dev_guide/dev.taxapi.html].

Details in diagrams, such as whether attributes and elements are optional, reflect the schema but descriptions reflect use of taxonomies in the SDM to define property types.

The target namespace of this schema is `http://systinet.com/uddi/taxonomy/v3/5.0` and the namespace prefix `taxonomy:` is used.

The schema uses definitions from the *UDDI* V3 schema with *URN* `urn:uddi-org:api_v3` for which the namespace prefix `uddi:` is used.

Taxonomies in SOA Systinet are compatible with taxonomies in HP SOA Systinet Registry. They are represented as XML documents using the same schema:

http://www.systinet.com/doc/wasp_uddi-55/wsdl/taxonomy.xsd

However, this schema defines elements used for other purposes, such as listing and referencing taxonomies on API's, and since SOA Systinet only uses a subset of possible taxonomies, it is only possible to import taxonomies satisfying additional constraints. These are described in Taxonomy Schema on page 91 in SOA Systinet Reference Guide.

You can see which taxonomies are installed by accessing the taxonomy collection as a URL such as:

```
http://hostname:8080/soa/systinet/platform/rest/repository/taxonomies/?view
```

Adjust the URL according to your installation (hostname, deployment context and port).

taxonomy

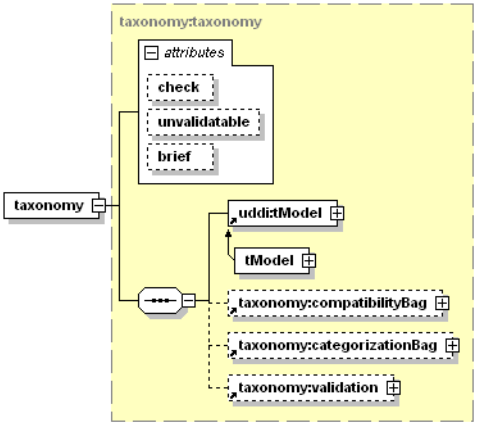


Table 1. Attributes

Name	Required	Description
`check`	In SOA Systinet	Always `yes`.
`unvalidateable`	no	Ignored.
`brief`	no	Ignored.

A `taxonomy:validation` element is always required in SOA Systinet.

tModel

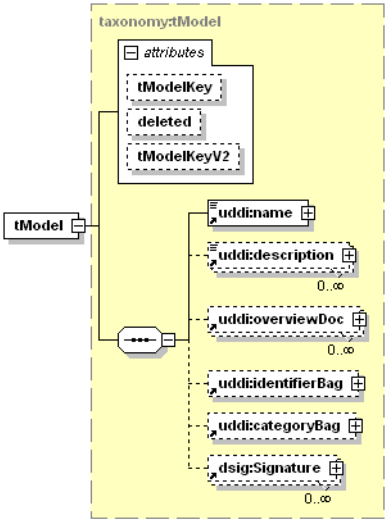


Table 2. Attributes

Name	Required	Description
`tModelKey`	no	Should be `uddi:uddi.org:categorization:types` in SOA Systinet.
`deleted`	no	Ignored
`tModelKeyV2`	no	Should be `uuid:c1acf26d-9672-4404-9d70-39b756e62ab4` in SOA Systinet.

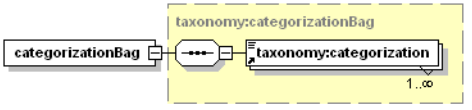
The type of element `taxonomy:tModel` extends the type of `uddi:tModel` by adding attribute `tModelKeyV2`. For full details of other attributes and elements, see the UDDI V3 specification.

compatibilityBag (and compatibility)

`taxonomy:compatibilityBag` contains one or more `taxonomy:compatibility` elements each containing one of the following text values:

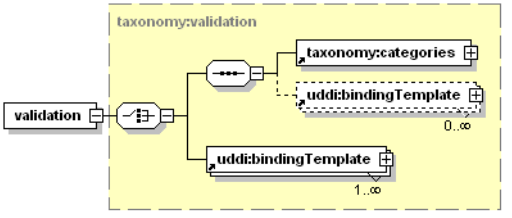
- tModel
- businessEntity
- businessService
- bindingTemplate

categorizationBag (and categorization)



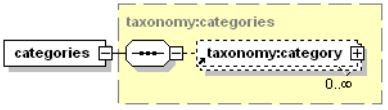
`taxonomy:categorizationBag` contains one or more `taxonomy:categorization` elements.

validation



In SOA Systinet a taxonomy:validation element is required and it must contain a `taxonomy:categories` element first. It may be followed by zero or more `uddi:bindingTemplate` elements, which are ignored.

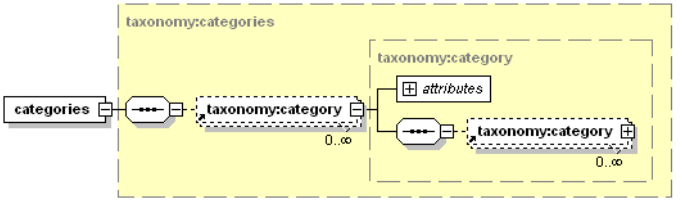
categories



A `taxonomy:categories` element contains zero or more `taxonomy:category` elements defining the categories in a taxonomy. It can be empty, but until possible categories are added, no artifacts can be created that use the taxonomy for a property type.

The `taxonomy:categories` element appears as the first child of taxonomy:validation because it is through validation constraints that the property type is defined. In SOA Systinet all taxonomies are defined in terms of a set of *category* values.

The categories can be arranged in a hierarchy as shown in this expansion of the above diagram.



category

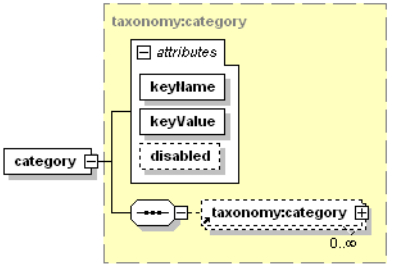


Table 3. Attributes

Name	Required	Description
keyName	Yes	Descriptive name used as a label.
keyValue	Yes	A unique identifier, typically mnemonic or numeric.
disabled	No	If yes then this value is now invalid. Either it is deprecated or it only exists to contain subcategories.

6 Policy Schema

The policy schema structure defines technical policies. HP uses the WS-Policy specification [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policy.asp] as a modeling framework for technical policies. Technical policies are prepared by Architects and Policy Developers who codify them as requested by the line-of-business managers, architectural councils, operational managers, etc.

In WS-Policy terms, a SOA Systinet technical policy = WS-Policy + name + documentation. SOA Systinet business policies are covered by the WS-PolicyAttachment specification [http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policyattachment.asp].

Policy strucure consists of a `wsp:Policy` element wrapping any number of assertion elements, as shown in Example 1 on page 97. SOA Systinet does not support `wsp:All` or `wsp:ExactlyOne`.

Example 1: Policy Definition Structure

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
                      xmlns:ex="http://www.example.com/assertions">

     <ex:Assertion1/>
     <ex:Assertion2/>
      <ex:Assertion3/>
     <ex:Assertion4/>

</wsp:Policy>
```

7 Assertion Schema

Assertions are SOA Systinet artifacts. They are stored in the HP SOA Systinet Platform repository and are accessed by the REST interface. The REST interface defines two different types of assertion endpoints, assertion collections and assertion definitions. Performing an HTTP GET operation on an assertion collection returns a list of assertion definitions. Performing an HTTP GET operation on an assertion definition returns SOA Systinet metadata and assertion details, specifically the reference template, implementations , and any parameters. The following sections cover each type and component of the assertion schema:

Assertion Collection Format on page 99. The XML collection listing multiple assertions.

Assertion Definition on page 101. The assertion definition, wrapping metadata and details.

Assertion Document Details on page 106. Details, or components, of the assertion definition.

Assertion Collection Format

The XML schema of a collection of assertions is the same as the XML schema of any other SOA Systinet listing of a collection. An assertion collection is shown in Example 1 on page 100

Example 1: Assertion Collection

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource xlink:href="http://localhost:8080/soa/systinet/platform/rest/repository/assertions/" ...
 >
  <rest:metadata>
    <rest:path>assertions/</rest:path>
    <rest:collection></rest:collection>
    <rest:binary>0</rest:binary>
    <rest:type>collection</rest:type>
    <rest:deleted>0</rest:deleted>

    <rest:owner>systinet:admin</rest:owner>
    <rest:revision>
      <rest:number>1</rest:number>
      <rest:timestamp>1970-01-01T00:00:00.000Z</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:metadata>
  <rest:descriptor/>
  <rest:data representation="list">
    <rest:resource
      xlink:href="http://localhost:8080/soa/systinet/platform/rest/repository/assertions/BP1013" ...>

      <rest:metadata>
        <rest:path>assertions/BP1013</rest:path>
        <rest:collection>assertions/</rest:collection>
        <rest:binary>0</rest:binary>
        <rest:contentType>text/xml</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>2007-02-14T10:18:52.498Z</rest:timestamp>
          <rest:creator>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/>

      </rest:metadata>
      <rest:descriptor>
        <p0:assertionArtifact deleted="0" ...>

          <p1:nameGroup>
            <p2:name>BP1013</p2:name>
          </p1:nameGroup>
          <p1:descriptionGroup>

            <p2:description>
              The content of the envelope matches the
              definition in the WSDL document. ...</p2:description>
          </p1:descriptionGroup>

          <g:artifactTypeGroup>
            ...
            <n3:artifactType name="Assertion"
taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
              value="urn:com:systinet:soa:model:artifacts:soa:policy:assertion"/>
          </g:artifactTypeGroup>
          <n4:lastRevision>1</n4:lastRevision>
          <n5:revision>1</n5:revision>
        </p0:assertionArtifact>
      </rest:descriptor>
    </rest:resource>
    ...
  </rest:data>
</rest:resource>
```

Assertion Definition

An assertion definition includes its description and definition of parameters and validation mechanisms.

You can view the XML serialization of an assertion in the UI. Open the assertion's detail page, accessible from either the **Catalog Browser** in the **Tools** tab or the **Views** menu in the assertion's page in the **Policies** tab. Click **XML View** in the detail page **Views** menu.

Alternatively, you can open a browser window with the XML View's URL, which is of the following form:

```
http://hostname:port/soa/systinet/platform/rest/repository/assertions/AssertionName
```

For example, in the standard installation of HP SOA Systinet Policy Manager, you can find the WS-I Basic Profile 1.1 assertion BP1001 at `http://localhost:8080/soa/systinet/platform/rest/repository/assertions/BP1001`.

The assertion definition has the following structure:

Example 2: Assertion Definition Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource
  xlink:href="http://b217:8080/soa/systinet/platform/rest/repository/assertions/BP1001" ...>
<rest:metadata>
 <rest:path>assertions/BP1001</rest:path>
 <rest:collection>assertions/</rest:collection>
 <rest:binary>0</rest:binary>
 <rest:contentType>text/xml</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>2007-02-14T10:17:41.045Z</rest:timestamp>
   <rest:creator>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/>

</rest:metadata>
  <rest:descriptor>
    <a:assertionArtifact deleted="0" xlink:href="assertions/BP1001" ...>

      <g:nameGroup>
        <p:name>BP1001</p:name>
      </g:nameGroup>
      <g:descriptionGroup>

        <p:description>...</p:description>
      </g:descriptionGroup>
       ...
      <g:artifactTypeGroup>
      ...
       <p:artifactType name="Assertion" taxonomyUri="uddi:systinet.com:soa:model:taxonomies:artifactTypes"

          value="urn:com:systinet:soa:model:artifacts:soa:policy:assertion"/>
```

```
      </g:artifactTypeGroup>
      <p:lastRevision>1</p:lastRevision>
      <p:revision>1</p:revision>
    </a:assertionArtifact>
  </rest:descriptor>
  <rest:data representation="xmldata">
    <rest:xmlData contentType="text/xml">
      <pm:Assertion xmlns:pm="http://systinet.com/2005/10/soa/policy">

        <pm:Template xmlns:wsi="http://www.ws-i.org/testing/2004/07/assertions/">
          <wsi:BP1001/>
        </pm:Template>
        <pm:Validation
SourceType="xmlns(ns=http://systinet.com/2005/10/soa/policy/report)qname(ns:Message)">
          <wsim:Message AssertionID="BP1001"
xmlns:wsim="http://systinet.com/2005/10/soa/policy/validation/wsi-message"/>
        </pm:Validation>
      </pm:Assertion>
    </rest:xmlData>
  </rest:data>

</rest:resource>
```

The elements of this structure are defined as follows:

- **`/rest:resource`.** An element containing the resource.
- **`/rest:resource/@xlink:href`.** Reference to the item. The same as the requestURI attribute on this element.
- **`/rest:resource/rest:metadata`.** The generic metadata of the resource. See HP SOA Systinet Platform documentation for details.
- **`/rest:resource/rest:descriptor/a:assertionArtifact`.** An assertion descriptor, containing assertion specific metadata.
- **`/rest:resource/rest:descriptor/a:assertionArtifact/p:*`.** A property. A generic format of property looks like: `<prefix:local taxonomyURI='uri' name='name' value='value'/>`.

It takes approximately 10 minutes for changes in the SDM configuration to propagate to HP SOA Systinet Policy Manager.

- **`/rest:resource/rest:descriptor/a:assertionArtifact/g:*`.** A property group. If more properties with the same namespace URI and local part are present, they should be included in a property group. Although this group should be named as a plural of the local part of the assertion, this is not checked by HP SOA Systinet Policy Manager.

- **`/rest:resource/rest:data/rest:xmlData`.** A wrapper for the "executive" part of the assertion.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion`.** Root element containing the definition of the "executive" part of an assertion.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Template`.** This required element must contain exactly one child element, which is a reference template of how this assertion looks as a WS-Policy document. If there are namespace definitions here, they are included in the reference template. If the assertion has any parameters, you can define default values for them in the reference template. If there are no namespaces or parameters, the reference template can be in the form <name/>.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter`.** An assertion in a WS-Policy document may contain parameters including timeouts (in WS-ReliableMessaging), type of authentication, required SOAP header elements, etc. This element gives a definition of such parameters, including the type of the parameter and where the parameter can be found in an instance of the assertion. This information is used both by the UI console and by policy validators.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter/@Name`.** The name of the parameter. This name will be shown in the UI.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter/@Type`.** Schema type of the parameter's value.

  In this release, the schema type is not used in either the UI or in the validation process. This behavior is likely to be changed.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter/@XPointer`.** In the absence of a `ValueXPointer` attribute, this attribute identifies the place of the parameter in the assertion's template (that is, how the attribute can be obtained from an instance of the assertion). Only a simplified form of the XPointer can be used.

  The evaluation context for the XPointer is the root of the actual assertion. So, for example, `b[1]` is the first "b" child of the assertion's element.

In this release, an XPath starting with "/" is interpreted to point to the root of the policy document. This behavior will be changed, so do not use absolute XPaths.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter/@ValueXPointer.`** `ValueXPointer` identifies the place of the parameter *relative to* the place identified by the `XPointer` attribute. When the parameter is not set, the element referenced by the `XPointer` attribute is removed from the instance. When the parameter is defined, its value is set to a place identified by the concatenation of the `XPointer` and `ValueXPointer` values. The rationale for this attribute is that there are assertions whose schema requires that either an attribute is set or the attribute's parent element is missing.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Parameter/@Optional.`** This attribute tells whether the parameter is optional, that is, if it can be omitted from the assertion instance.

- **`/rest:resource/rest:data/rest:xmlData/pm:Assertion/pm:Validation.`** The implementation, as described in Implementations on page 111.

Assertion Document Details

Example 3 on page 107 is the raw XML document of the UDDI BE 01 assertion.

Example 3: UDDI BE 01 Assertion XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
   <pm:Assertion xmlns:pm="http://systinet.com/2005/10/soa/policy"
                 xmlns:up="http://systinet.com/2005/10/soa/policy/uddi"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <pm:Parameter Name="lang" Type="xs:string" XPointer="xpointer(@RequiredLang)"/>
<!-- template of the instance of the assertion -->
        <pm:Template>
          <up:UDDI_BE_01 RequiredLang="en"/>
        </pm:Template>
        <pm:Validation SourceType="xmlns(ns=urn:uddi-org:api_v2)qname(ns:businessEntity)"
                xmlns:uddi="urn:uddi-org:api_v2"
         xmlns:val="http://systinet.com/2005/10/soa/policy/validation">
<!-- the validation is implemented via xpath expression -->
          <val:XPath>
           count(/uddi:businessEntity/uddi:name[@xml:lang=$lang])&gt;0
          </val:XPath>
       </pm:Validation>
 </pm:Assertion>
```

The key components of the assertion, visible in both the UI and the XML document, are:

- Reference Template
- Parameter
- Implementation, which includes the validation handler.

Reference Templates

The reference template defines what the assertion looks like instantiated as a WS-Policy document (See the generic `<pm:Template>` element shown in Example 3 on page 107.). If there is a namespace to be defined it is included in the reference template. If there are parameters, you can define the default values they point to. If there is no namespace or parameter, the template can be a simple empty tag, like `<assertionName/>`.

The UDDI BE 01 assertion reference template defines the `up` namespace. The assertion has one parameter, `lang`, which points to the `RequiredLang` attribute. The reference template sets the default value of this parameter, `en`. The actual XML of the reference template is:

```
<p:Template>
        <up:UDDI_BE_01 RequiredLang="en" xmlns:up="http://systinet.com/2005/10/soa/policy/uddi"/>
        </p:Template>
```

Reference templates must obey the following rules:

- The template name must be unique.
- The template must be a complete and valid XML element, not a fragment.
- The template can carry a namespace. This is the case with the WS-I BasicProfile assertion reference templates, such as `<wsi:BP1004 xmlns:wsi="http://www.ws-i.org/testing/2004/07/assertions/"/>`

Parameters

Parameters represent requirements whose specific values may vary. They include such things as timeouts, type of authentication, required SOAP header elements, etc. The value referenced by a parameter can differ between technical policies containing the parameter's parent assertion because each technical policy contains its own instance of the assertion.

Using parameters lets the policy developer reuse assertions. The developer can set a different required value for an assertion in each policy in which the assertion is used. Without parameters, the developer would need a separate assertion for each required value.

Example 4 on page 108 is an assertion taken from a policy file (namespaces omitted for brevity). Note the attribute `RequiredLang` with the value of "en". This attribute represents the `RequiredLang` parameter. Its default value is "en" for English. This default value is specified in the reference template (see Reference Templates on page 107) but the policy developer can change this value in individual policy files. If the assertion developer does not specify the parameter's default value in the reference template and does not set the parameter as optional, the policy developer must set the parameter value when creating a technical policy with the parameter's parent assertion.

Example 4: Assertion With Parameter

```
<wsp:Policy xmlns:wsp="..."/>
        <up:UDDI_BE_01 RequiredLang="en" xmlns:up="..."/>
</wsp:Policy>
```

A parameter definition has the following structure:

- **`pm:Parameter/@Name`.** Name of the parameter.
- **`pm:Parameter/pm:Description`.** Description of the parameter.
- **`pm:Parameter/@XPointer`.** Location of the modified attribute (expressed as an XPointer).
- **`pm:Parameter/@ValueXPointer`.** Location of the modified attribute (expressed as an XPointer). See below for details.
- **`pm:Parameter/@Optional`.** Optionality of the parameter (if it is optional, it might be left unfilled).

Another example:

```
<wsp:Policy xmlns:wsp="..."/>
        <up:Communication xmlns:up="...">
        <up:ConnectionTimeout value="10000"/>
        ...
        </up:Communication>
        </wsp:Policy>
```

This assertion checks whether communication settings contain a connection timeout set to at least 10 seconds. Additionally, the XML Schema of this assertion specifies that either the "value" must be present, or, to use the default value, the whole `up:ConnectionTimeout` element must be missing.

In this case, a single XPointer referencing the `up:ConnectionTimeout/@value` attribute is not enough, because HP SOA Systinet Policy Manager would not know that the whole element should be removed when the value is not entered. Therefore the parameter is now described in two XPaths:

- Location of the element that should be removed when the value of the parameter is not set
- Location of the value within the element defined above

The location of the element is set in the XPointer and the location of the value within the element is set in a ValueXPointer. For example, Example 5 on page 110 is a parameter with the ValueXPointer set at 5000. This results in the policy document in Example 6 on page 110. By contrast, if the developer leaves the ValueXPointer blank, the resulting policy document is Example 7 on page 110.

Example 5: Parameter with ValueXPointer Set at 5000

```
<p:Parameter Name="ConnectionTimeout" Optional="false" Type="xsd:integer"
        XPointer="xmlns(up=...)xpointer(up:ConnectionTimeout)"
        ValueXPointer="xpointer(@value)"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <p:Description>Connection timeout in milliseconds.</p:Description>
</p:Parameter>
```

Example 6: Policy Document with ValueXPointer in Parameter Set to 5000

```
<wsp:Policy xmlns:wsp="..."/>
    <up:Communication xmlns:up="...">
        <up:ConnectionTimeout value="5000"/>
    </up:Communication>
</wsp:Policy>
```

Example 7: Policy Document with Empty ValueXPointer in Parameter

```
<wsp:Policy xmlns:wsp="..."/>
        <up:Communication xmlns:up="...">
        </up:Communication>
</wsp:Policy>
```

Table 1 on page 111 shows the XML representations of various XPointer and ValueXPointer combinations, for optional and required attributes, and whether the value is defined or not. Example 8 on page 111 is a correctly defined XPointer.

Only a simplified form of XPointer is recognized in the parameter definition. The rationale is that in this context XPointer is used not only for retrieving data, but also for creating parameters via the UI. This is not possible with general XPointers. The recognized XPointer must have the following structure:

```
xmlns(prefix1=ns1)*xpointer({/{<prefix>:}?<localname>[<index>]}*)
```

Table 1. XPointer Combinations and Results

Optional	Value	XPointer	ValueXPointer	Result in Policy Schema
Yes/No	'ABC'	@P	—	<a P="ABC"/>
Yes	—	@P	—	<a/>
No	—	Prohibited		
Yes	'ABC'	b[1]	@P	<a>
Yes	—	b[1]	@P	<a/> (XPointer is removed.)
Yes	'ABC'	b[1]	—	<a>ABC
Yes	'ABC'	b[1]	c[1]	<a><c>ABC</c>
Yes	—	b[1]	c[1]	<a/> (XPointer is removed.)

Example 8: XPointer

```
xmlns(soap=http://schemas.xmlsoap.org/soap/envelope/)
xmlns(myns=http://systinet.com/examples/foo)xpointer(soap:Envelope[1]/soap:Body[1]/myns:Foo)
```

Implementations

An assertion has one implementation for each source type to which the assertion applies. Each implementation is propagated into its own `pm:Validation` element. An implementation contains the definition of the validation handler, in `p:Validation/##other[1]`, and the type of artifact which the assertion can be used to validate, in `p:Validation/@SourceType`.

Implementations use validation handlers if they do not specify manual validation. Validation handlers are pluggable pieces of code that show HP SOA Systinet Policy Manager how to validate a source document. Validation handlers are usually XPath or XQuery expressions, in which case the source code is included inside the implementation, but they can be custom made. Custom made validation handlers are written in Java and the implementation references the Java class.

Validation handlers and source types are described in the following sections:

Source Type on page 112. A description of all source types to which an implementation may apply.

XPath Assertions on page 114. XPath validation handlers.

XQuery Assertions on page 114. XQuery validation handlers.

Source Type

The `pm:Validation@SourceType` attribute defines the type of artifact validated by the assertion. `SourceType` must be a simplified XPointer identifying the root element of the resource which the assertion validates. If this parameter is omitted, the implementation would apply to sources of any type. However, for performance reasons it is better to map validation to a concrete source type, as narrowly as possible.

`SourceType` can be set as one of the following:

- A general artifact type with the namespace usually defined in the `pm:Validation` element. Please see Table 2 on page 112 for a list of these `SourceType`values and their associated artifacts and namespaces.
- A SOA Systinet artifact type. These share the namespace `xmlns:a="http://systinet.com/2005/05/soa/model/artifact"`. They are described in Part II, “SOA Definition Model”. A list of these `SourceType` values and their matching SOA Systinet artifact types is given in Table 3 on page 113.

Table 2. Source Types Applying to General Resources

SourceType value	Resource
xmlns(soap=http://schemas.xmlsoap.org/soap/envelope/)soap:Envelope	SOAP message
xmlns(wsdl=http://schemas.xmlsoap.org/wsdl/)wsdl:definitions	WSDL Definition
xmlns(xsd=http://www.w3.org/2001/XMLSchema)xsd:schema	XML Schema
xmlns(uddi=urn:uddi-org:api_v2)uddi:businessEntity	UDDI v2 Business Entity
xmlns(uddi=urn:uddi-org:api_v3)uddi:businessEntity	UDDI v3 Business Entity
xmlns(rest=http://systinet.com/2005/05/soa/resource)rest:resource	Any SOA Systinet resource

Table 3. SourceTypes Applying to SOA Systinet Artifacts

SourceType Value	SOA Systinet artifact
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)agreementArtifact	Agreement
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)businessPolicyArtifact	PM Business Policy
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)businessServiceArtifact	Business Service
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)contactArtifact	Contact
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)contractArtifact	Contract
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)contractRequestArtifact	Request
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)documentationArtifact	Documentation
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)personArtifact	Person
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)policyArtifact	Policy
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)registryArtifact	Registry
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)reportArtifact	Report
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)schemaArtifact	Schema
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)sloArtifact	SLA
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)taxonomyArtifact	Taxonomy
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)uddiChannelArtifact	UDDI Channel
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)uddiEntityArtifact	UDDI Entity
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)uddiRegistryArtifact	UDDI Registry
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)webArtifact	Web Application
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)webServiceArtifact	SOAP Service
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)wsPolicyArtifact	WS-Policy
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)wsdlArtifact	WSDL
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)xmlSchemaArtifact	XML Schema
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)xmlServiceArtifact	HTTP Service
xmlns(a=http://systinet.com/2005/05/soa/model/artifact)xsltArtifact	XSLT

XPath Assertions

Example 9 on page 114 is an XPath that applies to UDDI business entities and returns every `name` element whose `lang` attribute is set to the same value as the value of the `lang` parameter. If the XPath returns a non-empty list, the source document is considered to be valid against the assertion. If the returned node list is empty, validation has failed..

Example 9: XPath Expression

```
<val:XPath>
            count(/uddi:businessEntity/uddi:name[@xml:lang=$lang])&gt;0
            </val:XPath>
```

You must take the following points into account when writing XPath assertions:

- **Namespace.** The element `val:XPath` is the namespace context for the XPath expression. If you need to define a prefix-namespace mapping, do it on this element or its ancestors.
- **Type system.** The XPath engine used in this enforcer is the free version of the Saxon-B 8.5.1 [http://www.saxonica.com] XSLT/XPath/XQuery engine. Although this version does not contain XML Schema parsing, it still checks for type conformance. For example, if you need to check that the value of attribute "xyz" is greater than 5, include in your XPath expression:

  ```
  xs:integer(@xyz) > 5
  ```

  If you fail to retype to integer, the XPath expression will never be fulfilled and no warning will be returned.
- **Parameter type.** In this release, assertion parameters are always passed as strings, regardless of the schema type written in the parameter definition. For this reason you have to explicitly cast the parameter in numerical comparisons. For example, the following XPath expression would be used in an assertion which checks that the message's body has at most a given number of elements (defined as a parameter named MaxElements):

  ```
  count(soap:Body//*) <= xs:integer($MaxElements)
  ```

XQuery Assertions

XQuery expression can be represented as shown in Example 10 on page 115:

Example 10: XQuery Expression

```
<val:XQuery>

              declare namespace rest="http://systinet.com/2005/05/soa/resource";
              declare namespace a="http://systinet.com/2005/05/soa/model/artifact";
              declare namespace p="http://systinet.com/2005/05/soa/model/property";
              declare namespace val="http://systinet.com/2005/10/soa/policy/validation";

              declare variable $metadata.source.url external;

              if (exists(rest:resource/rest:descriptor/a:businessServiceArtifact/p:productionStage))
then
              val:assertionOK()
              else
              val:assertionFailed(concat('This service is not assigned a category from a lifecycle
taxonomy. ',
              'To fix this problem, go to <a href="', $metadata.source.url, '&view">the service</a>, ',

              'click on "Edit" and assign the category.'))

              </val:XQuery>
```

The XQuery in Example 10 on page 115 comes from the Service Supports Lifecycle assertion. The XQuery applies to business services and checks that each service has a lifecycle stage assigned to it. In the SOA Systinet use of XQueries, the `assertionOK` function is called only one time per tested artifact if the artifact passes validation, whereas if the artifact fails, the `assertionFailed` function is called for each individual violation. For the XQuery in Example 10 on page 115 there is no logical need to call `assertionFailed` more than once, since the artifact either has one lifecycle stage or none at all. In Example 11 on page 116, the XQuery checks each `include` and `import` element and makes sure they use relative references. The `assertionFailed` function is called for each element that does not use relative references.

Example 11: XQuery Reporting Multiple Failures

```
declare namespace xs = &quot;http://www.w3.org/2001/XMLSchema&quot;;
            declare namespace val=&quot;http://systinet.com/2005/10/soa/policy/validation&quot;;

            let $errors :=
            for $el in //xs:*[local-name() = 'include' or local-name() = 'import'] where
($el/@schemaLocation and contains($el/@schemaLocation, ':'))
            return
           val:assertionFailed(concat('This xs:', local-name($el), ' uses absolute reference to another
 schema.'), $el)
            return
            if (empty($errors)) then
            val:assertionOK()
            else
            ()
```

Namespaces are not propagated from parent elements but defined via standard XQuery declarations.

Together with the source document, XQuery assertions can be called with additional parameters. For example, these parameters can be used by the assertion to perform additional checks or output the location of the problem back to the user. The parameters are added to the XQuery expression of the assertion. A metadata parameter is shown in Example 10 on page 115.

Parameter name	Description
metadata.source.url	The URL of the source of validation. In the case of HTTP request/response, this points to the request/response message. For one-way messages, WSDL documents etc. it points to the resource being validated.
metadata.description.url	The URL of the associated description document (for example, WSDL associated to a log of messages).

If you want to write a new XQuery assertion or modify an existing one, follow these guidelines:

- The XQuery engine used in this enforcer is the free version of the Saxon-B 8.5.1 [http://www.saxonica.com] XSLT/XPath/XQuery engine. Although this version does not contain XML

Schema parsing, it still checks for type conformance. For example, if you need to check that the value of attribute "xyz" is greater than 5, write:

```
xs:integer(@xyz) > 5
```

Failing to do so, the XQuery expression might never be fulfilled. If this happens, no warning will be returned.

- In this release, assertion parameters are always passed as strings, regardless of the schema type written in the parameter definition. Because of this you must explicitly cast the parameter in numerical comparisons. For example, the following expression would be used in an assertion which checks that the message's body has at most a given number of elements (defined as a parameter named *MaxElements*):

```
count(soap:Body//*) <= xs:integer($MaxElements)
```

Index