HP Systinet

Software Version: 10.01
Windows and Linux Operating Systems

Reference Guide

Document Release Date: June 2015
Software Release Date: June 2015

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Intel® Xeon® and Intel® Core i7® are registered trademarks of Intel Corporation in the U.S. and other countries.

Microsoft®,Windows®,Windows® XP and Windows 7® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of TheOpenGroup.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Documentation Updates

The title page of this document contains the following identifying information:

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

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

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

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

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

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

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

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

About this PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online help.

Contents

In This Guide 7

Chapter 1: Service Definition Model 8
- Artifact Type Documentation 9
- Property Documentation 10
- Property Group Documentation 11
- Property Types 12

Chapter 2: Model Changes 14
- Collection Changes 14
- Artifacts 15
- Properties 16

Chapter 3: Breadcrumb Configuration 18

Chapter 4: Taxonomy Schema 19
- taxonomy 20
- tModel 20
- compatibilityBag (and compatibility) 21
- categorizationBag (and categorization) 21
- validation 21
- categories 22
- category 22
- Taxonomy Schema Reference 23

Chapter 5: Versioning Schema 28

Chapter 6: Policy Artifacts 31

Chapter 7: Policy Schema 32

Chapter 8: Assertion Schema 33
- Assertion Definition 33
- Assertion Document Details 36
  - Reference Templates 38
  - Parameters 39

Implementations 43
Source Type 44
XPath Assertions 46
XQuery Assertions 46

Chapter 9: Publishing Invalid WSDLs 49

Send Documentation Feedback 50

In This Guide

This HP Systinet Reference Guide provides a detailed look at the architecture of HP Systinet and contains the following topics:

- "Service Definition Model" on page 8

  A guide to the content and hierarchy of the SDM model.

- "Model Changes" on page 14

  Lists the changes to the SDM Model between Systinet 4.10 and Systinet 10.01.

- "Taxonomy Schema" on page 19

  A guide to the format of significant taxonomies in Systinet.

- "Versioning Schema" on page 28

  A guide to the format of versioning configurations in Systinet.

- "Policy Artifacts" on page 31

  Policy Manager artifacts and their relationships.

- "Policy Schema" on page 32

  The schema structure on which Policy Manager bases technical policies.

- "Assertion Schema" on page 33

  The schema of assertions and collections of assertions. Assertions are the atomic level of policies created by Policy Manager.

- "Publishing Invalid WSDLs" on page 49

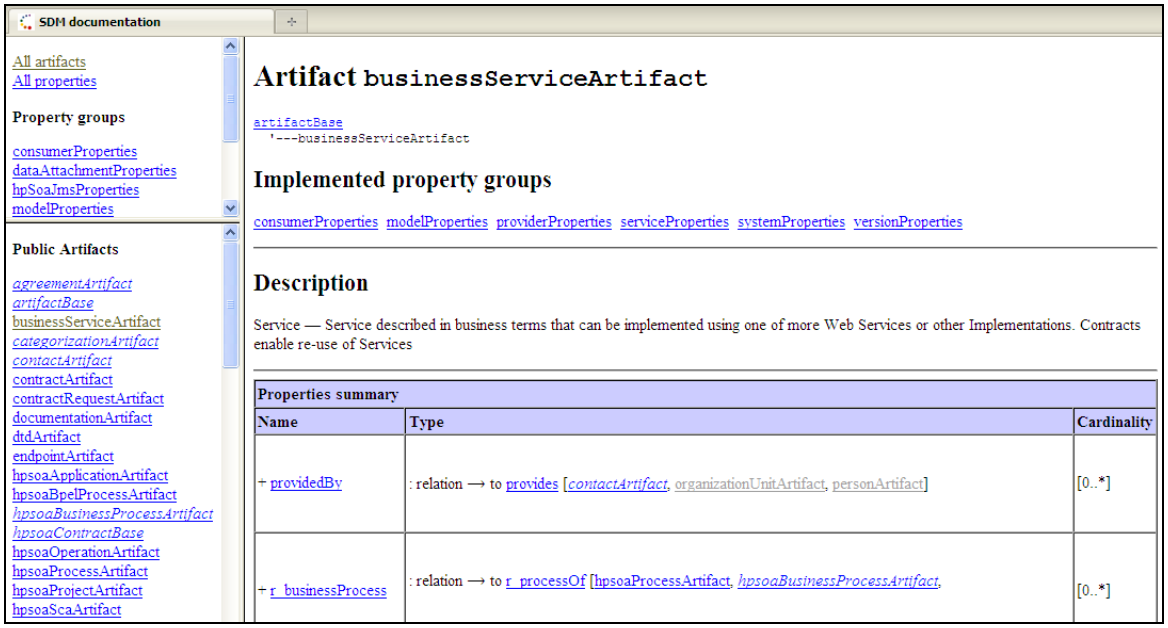
  How Systinet handles invalid WSDL documents.

Chapter 1: Service Definition Model

The Service Definition Model is a schema describing the hierarchy of artifact types in Systinet.

The model consists of a hierarchy of artifact types with each artifact type defining the set of properties applicable to it. The hierarchy enables properties to be defined for a higher level artifact and then inherited by the artifact types beneath it. Common properties are also organized into property groups which are assigned to artifact types.

The installation directory contains a full description of all the default artifact types, property groups, and properties accessible at `SYSTINET_HOME/doc/sdm/index.html` or `http://host:port/hp-systinet-doc/doc/sdm/index.html`.



The documentation provides a set of menus on the left and artifact type, property group, or property descriptions in the main pane on the right.

Note: All menu content uses the artifact type, property, and property group localnames.

Use the top menu to control the content of the menu below using the following links:

- **All Artifacts**

  View the list of all artifacts in the default model split into Public and System models. Artifact types in *italics* are abstract artifact types which do not have instances in the Catalog, but instead act as collective artifact types, such as Implementations, to group artifact types with instances, such as SOAP Services and Web Applications, and for property inheritance purposes.

- **All Properties**

View the list of all properties in the default model.

- **Property Groups**

  Click a property group to view a list of all the artifact types that use the property group.

Click an artifact type, property, or property group name to view its details in the main pane and view the following sections for details:

- "Artifact Type Documentation" below
- "Property Documentation" on the next page
- "Property Group Documentation" on page 11

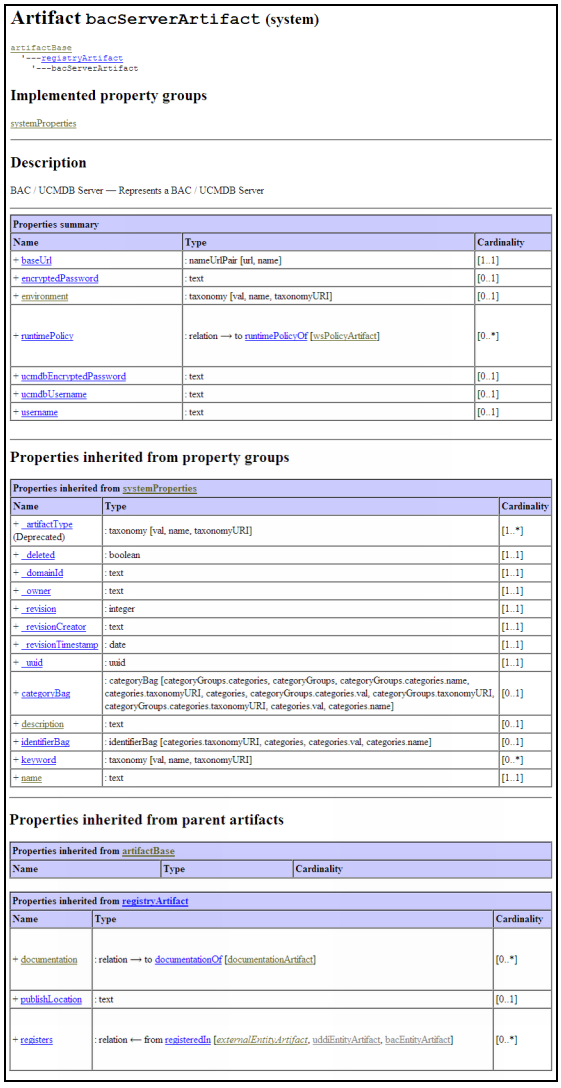
Artifact Type Documentation

The documentation for an artifact type displays the following information:

- The title showing the artifact localname. System artifacts are denoted with (system) and abstract artifacts denoted with (abstract).
- The hierarchy of artifact types that the artifact belongs to.
- The property groups applicable to the artifact type. Click a property group to view its details.
- Any directly associated sub-artifacts.
- The description showing the label used in the user interface and a description of the artifact type.
- The set of properties applicable to the artifact type divided into the following:
  - **Properties Summary** - These properties are directly associated with the artifact type in the model.
  - **Properties inherited from property groups** - Lists the set of properties applicable to the artifact defined by property groups assigned to the artifact type. Each applicable property group displays in its own table.
  - **Properties inherited from parent artifacts** - Lists the set of properties inherited from artifact types higher in the hierarchy, Each artifact type displays in its own table.
  - Each table shows the following information about each property:
    - **Name** - Click the property name to view its details.
    - **Type** - The property type. For details, see "Property Types" on page 12.
    - **Cardinality** - The number of times the property can occur for an artifact type with the

following possible values:

- [0..1] - Optional property that can occur only once.
- [0..*] - Optional property that can occur multiple times.
- [1..1] - Required property that only occurs once.
- [1..*] - Required property that must occur at least once.



Artifact `bacServerArtifact` **(system)**

artifactBase
'---registryArtifact
'---bacServerArtifact

Implemented property groups

systemProperties

Description

BAC / UCMDB Server — Represents a BAC / UCMDB Server

Properties summary		
Name	**Type**	**Cardinality**
+ baseUrl	: nameUrlPair [url, name]	[1..1]
+ encryptedPassword	: text	[0..1]
+ environment	: taxonomy [val, name, taxonomyURI]	[0..1]
+ runtimePolicy	: relation ⟶ to runtimePolicyOf [wsPolicyArtifact]	[0..*]
+ ucmdbEncryptedPassword	: text	[0..1]
+ ucmdbUsername	: text	[0..1]
+ username	: text	[0..1]

Properties inherited from property groups

Properties inherited from systemProperties		
Name	**Type**	**Cardinality**
+ _artifactType (Deprecated)	: taxonomy [val, name, taxonomyURI]	[1..*]
+ _deleted	: boolean	[1..1]
+ _domainId	: text	[1..1]
+ _owner	: text	[1..1]
+ _revision	: integer	[1..1]
+ _revisionCreator	: text	[1..1]
+ _revisionTimestamp	: date	[1..1]
+ _uuid	: uuid	[1..1]
+ categoryBag	: categoryBag [categoryGroups.categories, categoryGroups, categoryGroups.categories.name, categories.taxonomyURI, categories, categoryGroups.categories.val, categoryGroups.taxonomyURI, categoryGroups.categories.taxonomyURI, categories.val, categories.name]	[0..1]
+ description	: text	[0..1]
+ identifierBag	: identifierBag [categories.taxonomyURI, categories, categories.val, categories.name]	[0..1]
+ keyword	: taxonomy [val, name, taxonomyURI]	[0..*]
+ name	: text	[1..1]

Properties inherited from parent artifacts

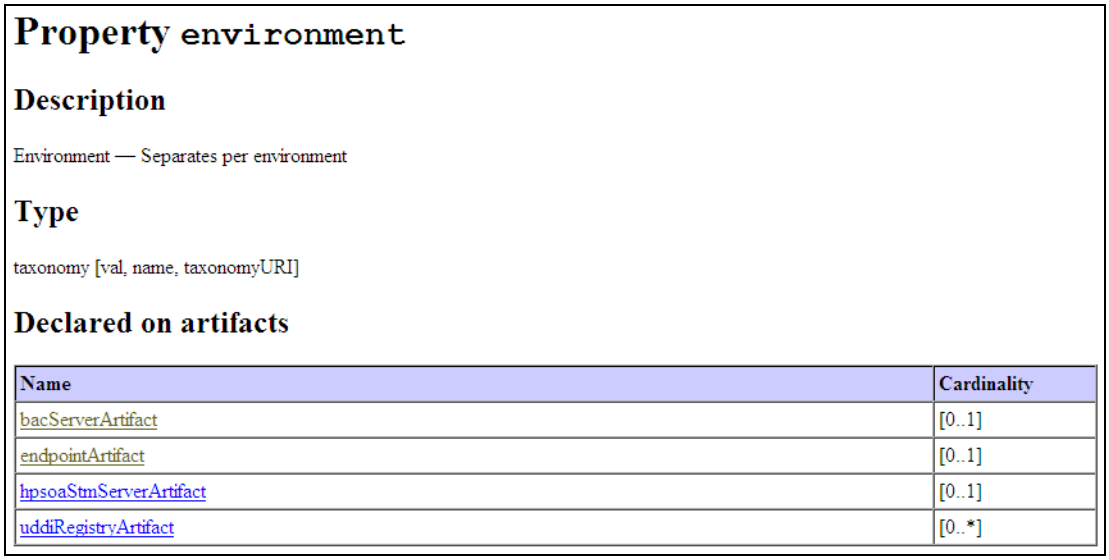
Properties inherited from artifactBase		
Name	**Type**	**Cardinality**

Properties inherited from registryArtifact		
Name	**Type**	**Cardinality**
+ documentation	: relation ⟶ to documentationOf [documentationArtifact]	[0..*]
+ publishLocation	: text	[0..1]
+ registers	: relation ⟵ from registeredIn [*externalEntityArtifact*, uddiEntityArtifact, bacEntityArtifact]	[0..*]

Property Documentation

The documentation for a property displays the following information:

- The title showing the property localname.
- The description showing the label used in the user interface and a description of the property.
- The property type. For more details, see "Property Types" on the next page.
- The set of artifact types and property groups that the property belongs to displaying the following information:
  - **Name** - Click the artifact name to view its details.
  - **Cardinality** - The number of times the property can occur for an artifact type with the following possible values:
    - [0..1] - Optional property that can occur only once.
    - [0..*] - Optional property that can occur multiple times.
    - [1..1] - Required property that only occurs once.
    - [1..*] - Required property that must occur at least once.



Property `environment`

Description

Environment — Separates per environment

Type

taxonomy [val, name, taxonomyURI]

Declared on artifacts

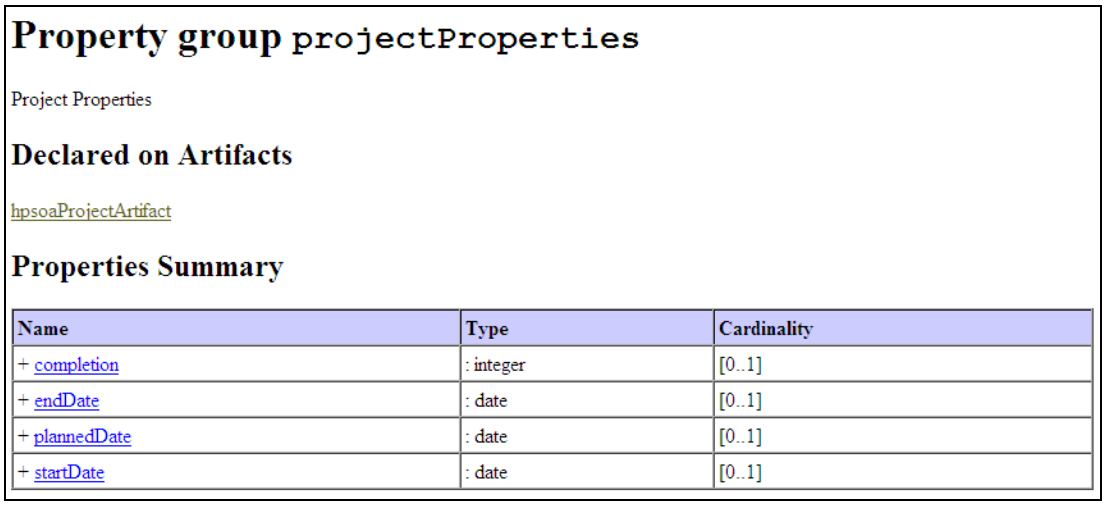
Name	Cardinality
bacServerArtifact	[0..1]
endpointArtifact	[0..1]
hpsoaStmServerArtifact	[0..1]
uddiRegistryArtifact	[0..*]

Property Group Documentation

The documentation for a property groups displays the following information:

- The title showing the property group localname.
- The description showing the property group label used in the user interface.

- The set of artifact types that the property group applies to. Click an artifact type name to view its details.
- The set of properties in the group displaying the following information:
  - **Name** - Click the property name to view its details.
  - **Type** - The property type. For details, see "Property Types" below.
  - **Cardinality** - The number of times the property can occur for an artifact type with the following possible values:
    - [0..1] - Optional property that can occur only once.
    - [0..*] - Optional property that can occur multiple times.
    - [1..1] - Required property that only occurs once.
    - [1..*] - Required property that must occur at least once.



Property group `projectProperties`

Project Properties

Declared on Artifacts

hpsoaProjectArtifact

Properties Summary

Name	Type	Cardinality
+ completion	: integer	[0..1]
+ endDate	: date	[0..1]
+ plannedDate	: date	[0..1]
+ startDate	: date	[0..1]

Property Types

Systinet uses the following property types.

Property Types

Type	Description
address	A full postal address
boolean	Boolean value - TRUE or FALSE
category	Used to assign several categories from a taxonomy to the artifact

Property Types, continued

Type	Description
categoryBag	Categorizes an artifact by taxonomies
dailyInterval	A time interval defined by a start day and end day, e.g. Monday to Friday
dateTime	A specific date and time
documentRelationship	Used to reference other artifacts etc.
identifierBag	Identifies the artifact by taxonomy. For categorization, use category bag instead
instanceDetail	Property type used by UDDI integration.
integer	Integer number
double	A double precision floating point number
nameUrlPair	URL with an optional name assigned
nameValuePair	A name and value pair
plainText	One-line text, suitable for textual information such as names
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	One-line text, suitable for machine readable information such as e-mail addresses
textarea	Multi-line text, suitable for information such as descriptions
xqueryParameter	Property type used to display parameters for executing an XQuery
encryptedPassword	Encrypted Password

Chapter 2: Model Changes

The following major changes were made to the SDM Model between HP SOA Systinet 4.x and Systinet 10.01:

- **REST Service model**: new set of artifacts and relationships
- **Web Service model**: renamed existing artifacts.
- **RGIF**: new set of artifacts, deprecate old artifacts.

The following sections list the specific changes to the model:

- "Collection Changes" below
- "Artifacts" on the next page
- "Properties" on page 16

Collection Changes

- `abstractEndpointArtifact`: now contains `endpointArtifact` and `httpEndpointArtifact`.
- `abstractOperationArtifact`: now contains `hpsoaOperationArtifact`, `httpRequestArtifact` and `httpResponseArtifact`.
- `hpsoaAppComponentArtifact`: now contains `hpsoaApplicationArtifact`, `hpsoaComponentArtifact` and `webArtifact`.
- `interfaceArtifact`: now contains 2 new artifacts `swaggerArtifact` and `wadlArtifact`.
- `implementationArtifact`: now contains new artifact `restServiceArtifact`.
- `r_messageProcessingPolicy`: now contains only `r_universalPolicyArtifact` and `r_runtimePolicyArtifact`.
- `r_deviceArtifact`: now contains new artifact `r_proxyAdapterArtifact`.
- `r_proxyArtifact`: now contains new artifacts `r_universalProxyArtifact` and `r_runtimePolicyParamArtifact`.
- `schemaArtifact`: now contains 2 new artifacts `jsonDefinitionArtifact` and `xmlDefinitionArtifact`.
- `sloArtifact`: now contains 2 new artifacts `contractOfferingArtifact` and

contractObjectivesArtifact.

- systemArtifact: now contains new artifacts iconArtifact and r_scriptArtifact.

Artifacts

added:

- contractObjectivesArtifact
- contractOfferingArtifact
- communicationSampleArtifact
- hpsoaRequirementArtifact
- httpEndpointArtifact
- httpHeaderArtifact
- httpStatusCodeArtifact
- httpRequestArtifact
- httpResponseArtifact
- iconArtifact
- jsonDefinitionArtifact
- restServiceArtifact
- r_universalPolicyArtifact
- r_runtimePolicyArtifact
- r_proxyAdapterArtifact
- r_universalProxyArtifact
- r_runtimePolicyParamArtifact
- r_scriptArtifact
- swaggerArtifact
- wadlArtifact
- xmlDefinitionArtifact

- fileTransferArtifact
- fileEndpointArtifact

removed artifacts:

- hpsoaContractBase
- r_datapowerXI50Artifact
- r_xi50WebServiceProxyArtifact
- r_xi50ProcessingPolicyArtifact
- r_l7GatewayArtifact
- r_l7ProxyArtifact
- r_l7ProcessingPolicyArtifact

deprecated:

- contractRequestArtifact
- contractRequestArtifact

changed:

- endpointArtifact: display name is changed from Endpoint to SOAP Endpoint
- hpsoaApplicationArtifact: extends from new artifact hpsoaAppComponentArtifact
- hpsoaComponentArtifact: extends from new hpsoaAppComponentArtifact instead of from implementationArtifact
- hpsoaOperationArtifact: display name is changed from Operation to SOAP Operation
- implementationArtifact: display name is changed from Implementation to Application Interface
- r_messageRoutingArtifact: display name is changed from Message Routing to RGIF
- schemaArtifact: extends from new artifact dataObjectArtifact
- xmlSchemaArtifact: display name is changed from XML Schema to XSD

Properties

removed:

- ***properties***:
  - accessPoint
  - r_jmsProvider
  - r_transport
  - resourceRevision
  - outOfSync
  - r_managementInterfaceUrl
  - r_webInterfaceURL
  - r_frontendURL
  - r_initialDeploymentTemplate
  - uddiApiVersion
  - fullVersion
  - portDocumentRelationship
- **relationships:**
  - http request/ composedOf: composedOf remove target type HTTP Request
  - restService/ assignedTo: assignedTo remove target type REST Service

changed:

- **relationships:**
  - http request/ r_operation : r_operation add target type HTTP Request
  - http endpoint/ endpointOf: endpointOf add target type HTTP Endpoint

Chapter 3: Breadcrumb Configuration

Breadcrumb is defined by rules in the breadcrumb configuration. The configuration is a single XML document persisted as a system setting, namely **platform.ui.breadcrumb.configuration**.

Administrator user is able to modify those rules. The format of the breadcrumb configuration is as below:

```
<breadcrumbs maxItems="5">
<artifact type="artifactTypeLocalName1">
<previous type="artifactTypeLocalName2" relation="relationshipLocalName1_2" />
<previous type="artifactTypeLocalName3" relation="relationshipLocalName1_3" />
...
</artifact>
<artifact type="artifactTypeLocalName2">
<previous type="artifactTypeLocalName4" relation="relationshipLocalName2_4" />
<previous type="artifactTypeLocalName5" relation="relationshipLocalName2_5" />
...
</artifact>
<artifact type="artifactTypeLocalName3">
<previous type="artifactTypeLocalName6" relation="relationshipLocalName3_6" />
<previous type="artifactTypeLocalName7" relation="relationshipLocalName3_7" />
...
</artifact>
...
</breadcrumbs>
```

Both type and relation attributes of the previous element is optional. Default value ANY is used if not provided.

There are maximum of **maxItems** on the breadcrumb path, from the right to left.

Changes made on this configuration is effected immediately, server start is not required.

Chapter 4: Taxonomy Schema

Each *taxonomy* is represented as an XML document with root element `taxonomy`.

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 the HP SOA Registry Foundation documentation.

Note: 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 Systinet are compatible with taxonomies in HP SOA Registry Foundation. They are represented as XML documents using the same schema shown in "Taxonomy Schema Reference" on page 23.

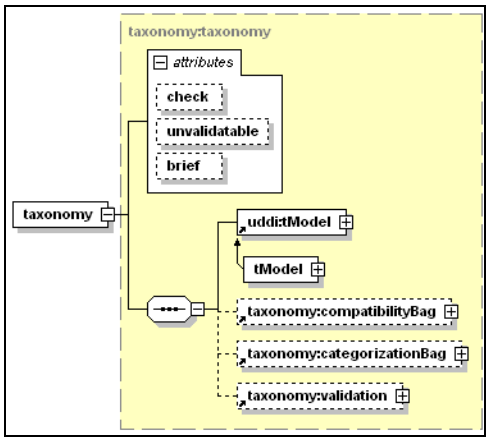
However, this schema defines elements used for other purposes, such as listing and referencing taxonomies on API's, and since Systinet only uses a subset of possible taxonomies, it is only possible to import taxonomies satisfying additional constraints. These are described in "Taxonomy Schema Reference" on page 23.

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

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

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

taxonomy

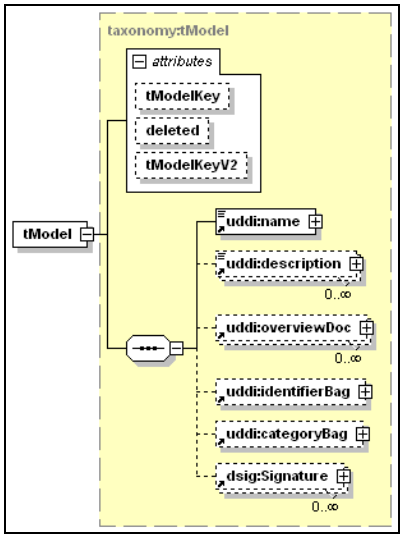


Attributes

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

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

tModel

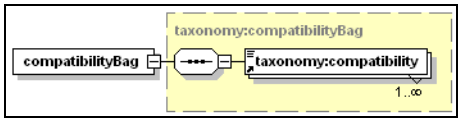


Attributes

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

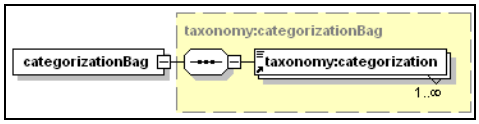
compatibilityBag (and compatibility)



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

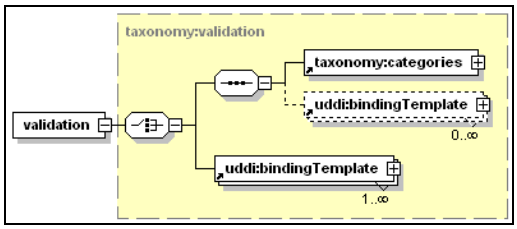
- tModel
- businessEntity
- businessService
- bindingTemplate

categorizationBag (and categorization)



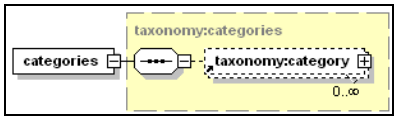
taxonomy:categorizationBag contains one or more taxonomy:categorization elements.

validation



In 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.

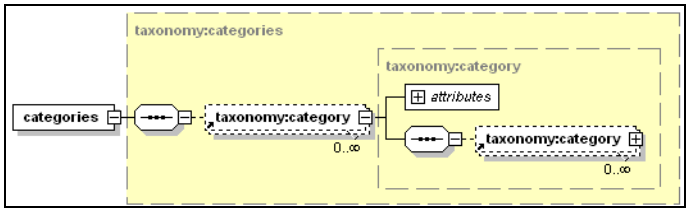
categories



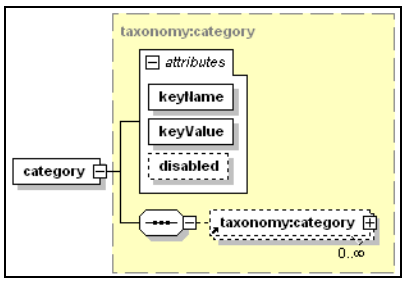
A `taxonomy:categories` element contains zero or more `taxonomy:category` elements defining the categories in a taxonomy. It can be empty, but until possible categories are added, no artifacts can be created that use the taxonomy for a property type.

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 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



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.

Taxonomy Schema Reference

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema id="taxonomy"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://systinet.com/uddi/taxonomy/v3/5.0"
    xmlns:taxonomy="http://systinet.com/uddi/taxonomy/v3/5.0"
    xmlns:uddi="urn:uddi-org:api_v3"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.00">
  <!--
      Copyright 2001-2005 Systinet Corp. All rights reserved.
      Use is subject to license terms.
  -->
  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xsd:import namespace="urn:uddi-org:api_v3" schemaLocation="uddi_v3.xsd"/>

  <xsd:element name="compatibility" type="taxonomy:compatibility"
final="restriction"/>
  <xsd:simpleType name="compatibility">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="businessEntity"/>
      <xsd:enumeration value="businessService"/>
      <xsd:enumeration value="bindingTemplate"/>
      <xsd:enumeration value="tModel"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:element name="categorization" type="taxonomy:categorization"
final="restriction"/>
  <xsd:simpleType name="categorization">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="categorization"/>
      <xsd:enumeration value="categorizationGroup"/>
      <xsd:enumeration value="identifier"/>
      <xsd:enumeration value="relationship"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:element name="taxonomy" type="taxonomy:taxonomy" final="restriction"/>
  <xsd:complexType name="taxonomy" final="restriction">
    <xsd:sequence>
      <xsd:element ref="uddi:tModel"/>
      <xsd:element ref="taxonomy:compatibilityBag" minOccurs="0"/>
```

```
      <xsd:element ref="taxonomy:categorizationBag" minOccurs="0"/>
      <xsd:element ref="taxonomy:validation" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="check" type="xsd:boolean" use="optional"/>
    <xsd:attribute name="unvalidatable" type="xsd:boolean" use="optional"/>
    <xsd:attribute name="brief" type="xsd:boolean" use="optional"/>
  </xsd:complexType>

  <xsd:element name="taxonomyDetail" type="taxonomy:taxonomyDetail"
final="restriction"/>
  <xsd:complexType name="taxonomyDetail" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:taxonomy" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="truncated" type="uddi:truncated" use="optional"/>
  </xsd:complexType>

  <xsd:element name="validation" type="taxonomy:validation" final="restriction"/>
  <xsd:complexType name="validation" final="restriction">
    <xsd:choice>
      <xsd:sequence>
        <xsd:element ref="taxonomy:categories"/>
        <xsd:element ref="uddi:bindingTemplate" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:element ref="uddi:bindingTemplate" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="categories" type="taxonomy:categories" final="restriction"/>
  <xsd:complexType name="categories" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:category" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="category" type="taxonomy:category" final="restriction"/>
  <xsd:complexType name="category" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:category" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="keyName" type="uddi:keyName" use="required"/>
    <xsd:attribute name="keyValue" type="uddi:keyValue" use="required"/>
    <xsd:attribute name="disabled" type="xsd:boolean" use="optional"/>
  </xsd:complexType>

  <xsd:element name="compatibilityBag" type="taxonomy:compatibilityBag"
final="restriction"/>
  <xsd:complexType name="compatibilityBag" final="restriction">
```

```
    <xsd:sequence>
      <xsd:element ref="taxonomy:compatibility" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="categorizationBag" type="taxonomy:categorizationBag"
final="restriction"/>
  <xsd:complexType name="categorizationBag" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:categorization" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="find_taxonomy" type="taxonomy:find_taxonomy"
final="restriction"/>
  <xsd:complexType name="find_taxonomy" final="restriction">
    <xsd:complexContent>
      <xsd:extension base="uddi:find_tModel">
        <xsd:sequence>
          <xsd:element ref="taxonomy:compatibilityBag" minOccurs="0"/>
          <xsd:element ref="taxonomy:categorizationBag" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="check" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="unvalidatable" type="xsd:boolean" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="taxonomyInfo" type="taxonomy:taxonomyInfo"
final="restriction"/>
  <xsd:complexType name="taxonomyInfo" final="restriction">
    <xsd:complexContent>
      <xsd:extension base="uddi:tModelInfo">
        <xsd:sequence>
          <xsd:element ref="taxonomy:categorizationBag"/>
          <xsd:element ref="taxonomy:compatibilityBag"/>
        </xsd:sequence>
        <xsd:attribute name="check" type="xsd:boolean" use="required"/>
        <xsd:attribute name="unvalidatable" type="xsd:boolean" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="taxonomyInfos" type="taxonomy:taxonomyInfos"
final="restriction"/>
  <xsd:complexType name="taxonomyInfos" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:taxonomyInfo" minOccurs="0"
maxOccurs="unbounded"/>
```

```
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="taxonomyList" type="taxonomy:taxonomyList"
final="restriction"/>
  <xsd:complexType name="taxonomyList" final="restriction">
    <xsd:sequence>
      <xsd:element ref="uddi:listDescription" minOccurs="0"/>
      <xsd:element ref="taxonomy:taxonomyInfos" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="truncated" type="xsd:boolean" use="optional"/>
  </xsd:complexType>

  <xsd:element name="get_taxonomy" type="taxonomy:get_taxonomy"
final="restriction"/>
  <xsd:complexType name="get_taxonomy" final="restriction">
    <xsd:complexContent>
      <xsd:extension base="uddi:get_tModelDetail">
        <xsd:attribute name="brief" type="xsd:boolean" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="delete_taxonomy" type="taxonomy:delete_taxonomy"
final="restriction"/>
  <xsd:complexType name="delete_taxonomy" final="restriction">
    <xsd:complexContent>
      <xsd:extension base="uddi:delete_tModel"/>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="save_taxonomy" type="taxonomy:save_taxonomy"
final="restriction"/>
  <xsd:complexType name="save_taxonomy" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:authInfo" minOccurs="0"/>
      <xsd:element ref="taxonomy:taxonomy" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="authInfo" type="taxonomy:authInfo" final="restriction"/>
  <xsd:simpleType name="authInfo">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

  <xsd:element name="tModel" type="taxonomy:tModel"
substitutionGroup="uddi:tModel"/>
  <xsd:complexType name="tModel">
    <xsd:complexContent>
```

```
      <xsd:extension base="uddi:tModel">
        <xsd:attribute name="tModelKeyV2" type="uddi:tModelKey" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="download_taxonomy" type="taxonomy:download_taxonomy"
final="restriction"/>
  <xsd:complexType name="download_taxonomy" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:authInfo" minOccurs="0"/>
      <xsd:element ref="uddi:tModelKey"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="upload_taxonomy" type="taxonomy:upload_taxonomy"
final="restriction"/>
  <xsd:complexType name="upload_taxonomy" final="restriction">
    <xsd:sequence>
      <xsd:element ref="taxonomy:authInfo" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Chapter 5: Versioning Schema

Systinet uses versioning strategy configurations in XML format described by the following versioning schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://hp.com/2009/11/systinet/platform/versioning/schema"
    targetNamespace="http://hp.com/2009/11/systinet/platform/versioning/schema"
    elementFormDefault="qualified">

  <xs:complexType name="mask">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:ID" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="maskref">
    <xs:attribute name="ref" type="xs:IDREF"/>
  </xs:complexType>

  <xs:simpleType name="separator">
    <xs:restriction base="xs:string">
      <xs:length value="1"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="group">
    <xs:simpleContent>
      <xs:extension base="xs:positiveInteger">
        <xs:attribute name="separator" type="separator" default="."/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="groupLocation">
    <xs:simpleContent>
      <xs:extension base="group">
        <xs:attribute name="prefix" type="xs:string" default=""/>
        <xs:attribute name="suffix" type="xs:string" default=""/>
        <xs:attribute name="mandatory" type="xs:boolean" default="true"/>
        <xs:attribute name="primary" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
```

```
  </xs:complexType>

  <xs:complexType name="groups">
    <xs:sequence>
      <xs:element name="group" type="groupLocation" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="maskOrMaskref">
    <xs:sequence>
      <xs:choice>
        <xs:element name="mask" type="xs:string"/>
        <xs:element name="maskref" type="maskref"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="sourceMask">
    <xs:complexContent>
      <xs:extension base="maskOrMaskref">
        <xs:sequence>
          <xs:element name="group" type="group" minOccurs="1"
maxOccurs="unbounded"/>
          <xs:element name="property" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="operator">
    <xs:restriction base="xs:string">
      <xs:enumeration value="prefix"/>
      <xs:enumeration value="suffix"/>
      <xs:enumeration value="contains"/>
      <xs:enumeration value="full"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="other">
    <xs:complexContent>
      <xs:extension base="sourceMask">
        <xs:sequence>
          <xs:element name="match" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="operator" type="operator" default="prefix"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:complexType name="source">
    <xs:sequence>
      <xs:element name="version" type="sourceMask"/>
      <xs:element name="other" type="other"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="location">
    <xs:complexContent>
      <xs:extension base="maskOrMaskref">
        <xs:sequence>
          <xs:element name="version" type="groups"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="masks">
    <xs:sequence>
      <xs:element name="mask" type="mask" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="schema">
    <xs:sequence>
      <xs:element name="masks" type="masks" minOccurs="0"/>
      <xs:element name="source" type="source"/>
      <xs:element name="location" type="location"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:element name="versioningSchema" type="schema"/>

</xs:schema>
```

Chapter 6: Policy Artifacts

Policy Manager entities (Technical Policy and Assertion) are represented in Systinet by artifacts. These artifacts are Technical Policy Artifact (hpsoaTechnicalPolicyArtifact) and Assertion Artifact (assertionArtifact).

Policy Manager entities are related to each other. A technical policy can reference both technical polices and assertions. The references are included in the entity data. The technical policy data contains references to other technical policies and to assertions. Since the entity data is in the artifacts, the references are included in the artifact data.

In addition these references between policy manager entities are represented by relations between Policy Manager artifacts. The details of the relations are listed in the following table.

Source (Referencing) Artifact	Outgoing Relation Property on Source	Incoming Relation Property on Target	Target (Referenced) Artifact
hpsoaTechnicalPolicyArtifact	r_referencedTechnicalPolicy	r_referencedTechnicalPolicyInverse	hpsoaTechnicalPolicyArtifact
hpsoaTechnicalPolicyArtifact	r_referencedAssertion	r_referencedAssertionInverse	assertionArtifact

The relations are created during the creation and update of the source artifact. The relations just reflect the references in the data. In the case that a user creates or deletes relations, the references in the artifact data stay unchanged. This creates an inconsistency between the references in the data and the relations between artifacts. The relations are created, based on the references, when the source artifact is saved (created or updated).

Chapter 7: Policy Schema

The policy schema structure defines technical policies. Systinet uses the WS-Policy specification 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.

Note: In WS-Policy terms, a technical policy = WS-Policy + name + documentation. Systinet policies are covered by the WS-PolicyAttachment specification.

Policy structure consists of a `wsp:Policy` element wrapping any number of assertion elements, as shown in "Policy Definition Structure". Systinet does not support `wsp:All` or `wsp:ExactlyOne`.

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 param="value" />
  <ex:Assertion3/>
  <ex:Assertion4/>
</wsp:Policy>
```

Chapter 8: Assertion Schema

Assertions are Systinet artifacts. They are stored in the 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. Create and edit assertions using *Assertion Editor*, a component of *Systinet Workbench*. For details, see the *Assertion Editor Guide*.

The following sections cover each type and component of the assertion schema:

- "Assertion Definition" below

  The assertion definition, wrapping metadata and details.

- "Assertion Document Details" on page 36

  Details, or components, of the assertion definition.

Assertion Definition

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

Note: You can view the XML serialization of an assertion in the Administration UI. Open the detail page for the assertion and select the Technical Details tab.

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

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

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

The assertion definition has the following structure:

Assertion Definition Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<rest:resource
xlink:href="http://b217:8080/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/systinet/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/systinet/policy/report)qname(ns:Message)">
     <wsim:Message AssertionID="BP1001"
xmlns:wsim="http://systinet.com/2005/10/systinet/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 Systinet 2 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'/>`.

  Note: It takes approximately 10 minutes for changes in the SDM configuration to propagate to 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 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.

  Note: 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 43.

Assertion Document Details

“UDDI BE 01 Assertion XML Document” is the raw XML document of the UDDI BE 01 assertion.

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>
```

Assertion documents contain the following elements:

- **`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/>.

- **`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.

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

- **`pm:Assertion/pm:Parameter/@Type`** Type of the parameter's value.

- **`pm:Assertion/pm:Parameter/@Taxonomy`** A taxonomy with values that the parameter can adopt. The taxonomy is specified using its tModelKey. This attribute is only required when `Type` has the `pm:taxonomy` value (with pm being the `xmlns:pm="http://systinet.com/2005/10/soa/policy` namespace), otherwise it is ignored (and optional).

- **`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.

- **`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.

- **`pm:Assertion/pm:Parameter/@Optional`**This attribute tells whether the parameter is optional, that is, if it can be omitted from the assertion instance.
- **`pm:Assertion/pm:Validation`**The implementation, as described in "Implementations" on page 43.

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

- "Reference Templates" below
- "Parameters" on the next page
- "Implementations" on page 43, 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 “UDDI BE 01 Assertion XML Document”.). 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, and so on. 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.

“Assertion With Parameter” 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 the previous page) 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.

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).

- `pm:Parameter/@Type`

Type of the parameter's value. Supported values are most of built-in W3C Schema data types (see http://www.w3.org/TR/xmlschema-2/#built-in-datatypes):

- xs:string
- xs:boolean
- xs:float
- xs:double
- xs:duration
- xs:dateTime
- xs:time
- xs:date
- xs:gYearMonth
- xs:gYear
- xs:gMonthDay
- xs:gDay
- xs:gMonth
- xs:hexBinary
- xs:base64Binary
- xs:anyURI
- xs:QName
- xs:integer
- xs:long
- xs:short
- xs:byte
- xs:unsignedLong
- xs:unsignedInt

- xs:unsignedShort
- xs:unsignedByte

Where xs is the `xmlns:xs="http://www.w3.org/2001/XMLSchema"` namespace.

Also supported is the value `pm:taxonomy` (with pm being the `xmlns:pm="http://systinet.com/2005/10/soa/policy` namespace), which specifies that the parameter will take on values from the taxonomy specified by the @Taxonomy attribute.

- `pm:Parameter/@Taxonomy`

  Taxonomy whose values the parameter adopts. Specified with the taxonomy tModelKey. The attribute is required only when Type has the `pm:taxonomy` value, otherwise it is be ignored (and optional). Actual parameter values are specified with keyValues in policy documents.

The following examples demonstrate the use of a taxonomy-based parameter and the corresponding policy document:

Parameter with Taxonomy Type

```
<pm:Parameter Name="artifactType" Optional="true" Type="pm:taxonomy"
    Taxonomy="uddi:systinet.com:soa:model:taxonomies:artifactTypes"
    XPointer="xpointer(@artifactType)"
    xmlns:pm="http://systinet.com/2005/10/soa/policy">
  <pm:Description>Artifact type to restrict applicability.</pm:Description>
</pm:Parameter>
```

Policy Document with a Taxonomy-Based Assertion

```
<wsp:Policy xmlns:ws="...">
  <up:MyAssertion
artifactType="urn:com:systinet:soa:model:artifacts:soa:applicationArtifact"
    xmlns:up="..."/>
</wsp:Policy>
```

The following 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.

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

In this case, a single XPointer referencing the `up:ConnectionTimeout/@value` attribute is not enough, because 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, “Parameter with ValueXPointer Set at 5000”is a parameter with the ValueXPointer set at 5000. This results in the policy document in “Policy Document with ValueXPointer in Parameter Set to 5000”. By contrast, if the developer leaves the ValueXPointer blank, the resulting policy document is “Policy Document with Empty ValueXPointer in Parameter”.

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>
```

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>
```

Policy Document with Empty ValueXPointer in Parameter

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

Table “XPointer Combinations and Results” shows the XML representations of various XPointer and ValueXPointer combinations, for optional and required attributes, and whether the value is defined or not. “XPointer” is a correctly defined XPointer.

Note: 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>]}*)

XPointer Combinations and Results

Optional	Value	XPointer	ValueXPointer	Result in Policy Schema
Yes/No	'ABC'	@P	—	<a P="ABC"/>

Optional	Value	XPointer	ValueXPointer	Result in Policy Schema
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.)

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 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 the next page

  A description of all source types to which an implementation may apply.

- "XPath Assertions" on page 46

  XPath validation handlers.

- "XQuery Assertions" on page 46

  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 “Source Types Applying to General Resources” for a list of these `SourceType`values and their associated artifacts and namespaces.
- An artifact type. These share the namespace `xmlns:a="http://systinet.com/2005/05/soa/model/artifact"`. A list of these `SourceType` values and their matching artifact types is given in the following table.

Source Types Applying to General Resources

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

SourceTypes Applying Artifacts

Artifact Type	SourceType Value
Agreement	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) agreementArtifact
Application	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) hpsoaApplicationArtifact
Business Policy	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) businessPolicyArtifact
Business Service	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) businessServiceArtifact

Artifact Type	SourceType Value
Consumption Request	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) contractRequestArtifact
Contact	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)contactArtifact
Contract	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)contractArtifact
Conversation Document	xmlns(a=http://systinet.com/2005/10/soa/policy/report)Conversation
Documentation	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) documentationArtifact
HTTP Message Document	xmlns(a=http://systinet.com/2005/10/soa/policy/report)Message
Person	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)personArtifact
Policy	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)policyArtifact
Registry	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)registryArtifact
Report	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)reportArtifact
SOAP Service	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) webServiceArtifact
SLO	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)sloArtifact
Schema	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)schemaArtifact
Taxonomy	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) taxonomyArtifact
UDDI Channel	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) uddiChannelArtifact
UDDI Entity	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) uddiEntityArtifact
UDDI Registry	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) uddiRegistryArtifact
WS-Policy	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)wsPolicyArtifact
WSDL	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)wsdlArtifact
Web Application	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)webArtifact
XML Schema	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) xmlSchemaArtifact

Artifact Type	SourceType Value
XML Service	xmlns(a=http://systinet.com/2005/05/soa/model/artifact) xmlServiceArtifact
XSLT	xmlns(a=http://systinet.com/2005/05/soa/model/artifact)xsltArtifact

XPath Assertions

Example “XPath Expression” 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.

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 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 follows:

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 “XQuery Expression” 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 Systinet 2 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 “XQuery Expression” there is no logical need to call `assertionFailed` more than once, since the artifact either has one lifecycle stage or none at all. In “XQuery Reporting Multiple Failures”, 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.

XQuery Reporting Multiple Failures

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

Note: 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 “XQuery Expression”.

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).
metadata.source.is.subdocument	Detects subdocuments. Returns "false" if document is standalone, "true" if document is part of a larger document.

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 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)
  ```

Chapter 9: Publishing Invalid WSDLs

If WSDL documents are not complete or valid, Systinet will handle them according to the following table:

Usecase	Behaviour
File is not well-formed	In the case of popular extension use (XSD, WSDL...), the error message "Failed to upload data. The XSD/WSDL... appears to be non well-formed." will be presented. Otherwise, it will be published as a document artifact.
WSDL elements use undefined namespace	Display of error message "Created WSDL: ... . WSDL namespace is invalid." Also attempt to publish WSDL + imported WSDLs + imported XSDs creating the relationship among as many of them as possible. **Note:** publishing a WSDL may be in recursion as a WSDL may import other WSDL.
WSDL contains invalid references	For each invalid (non existed or non well-formed) referred resource: - If not well- formed, there will be the error item in the publishing report 'Unrecognized data content, 'WSDL' expected for ...' - A warning item for the resource which refers to that invalid resource: Failed reference 'text.xsd': Unrecognized data content, 'WSDL' expected for ...
WSDL contains valid references	WSDL + other imported WSDLs/XSDs will published creating the relationship among them.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Reference Guide (Systinet 10.01)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to docteam_systinet@hp.com.

We appreciate your feedback!