ebXML RegRep Profile for XML Schema Version 4.0

Committee Specification Draft 01-1

03 January 2012

Specification URIs

This version:

http://docs.oasis-open.org/regrep/regrep-profile-xsd/v4.0/csd01/regrep-profile-xsd.html
http://docs.oasis-open.org/regrep/regrep-profile-xsd/v4.0/csd01/regrep-profile-xsd.odt (Authoratative)
http://docs.oasis-open.org/regrep/regrep-profile-xsd/v4.0/csd01/regrep-profile-xsd.pdf

Previous version:

TBD

Latest version:

TBD

Technical Committee:

OASIS ebXML Registry TC

Chairs:

Kathryn Breininger (Kathryn.r.Breininger@boeing.com), Boeing
Farrukh Najmi (farrukh@wellfleetsoftware.com), Wellfleet Software

Editors:

Farrukh Najmi, (farrukh@wellfleetsoftware.com), Wellfleet Software
Oliver Newell (olivern@ll.mit.edu), MIT Lincoln Laboratories

Additional artifacts:

None

Related work:

This specification depends upon the ebXML RegRep 4 specification and the ebXML RegRep Profile for XML Namespaces specification.

Declared XML namespaces:

urn:oasis:names:tc:ebxml-regrep:profile:xsd

Abstract:

This document defines the ebXML RegRep profile for publishing, management, discovery and reuse of XML Schema documents.

Status:

This document is a draft specification for review, revision and approval by the OASIS ebXML RegRep TC.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/regrep/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/regrep/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[regrep-profile-xsd-v4.0]

OASIS ebXML RegRep Profile for XML Schema Version 4.0. 03 January 2012. Committee Specification Draft 01. http://docs.oasis-open.org/regrep/regrep-profile-xsd/v4.0/csd01/regrep-profile-xsd.html.

Notices

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

Index of Tables

1 Introduction

ebXML RegRep is a standard defining the service interfaces, protocols and information model for an integrated registry and repository. The repository stores digital content while the registry stores metadata that describes the content in the repository. This document defines the ebXML RegRep profile for publishing, management, discovery and reuse of XML Schema documents.

1.1 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF [RFC 2119].

The term XSD file may be used interchangeably with the term XML Schema file.

1.2 Normative References

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[regrep-overview-v4.0] *OASIS ebXML RegRep Version 4.0 Part 0: Overview Document*. 15 September 2011. Candidate OASIS Standard 01. http://docs.oasis-open.org/regrep/regrep-core/v4.0/cos01/regrep-core-overview-v4.0-cos01.html.

[regrep-xmlns-v4.0] *OASIS ebXML RegRep Profile for XML Namespaces Version 4.0*. Committee Specification Draft 01. http://docs.oasis-open.org/regrep/regrep-profile-xmlns/v4.0/csd01/regrep-profile-xmlns.html

[XPATH2] XML Path Language (XPath) 2.0, W3C Recommendation 23 January 2007
http://www.w3.org/TR/xpath20

[XPATHFUNC] XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Recommendation 23 January 2007
http://www.w3.org/TR/xpath-functions

1.3 Non-normative References

[UML] Unified Modeling Language
http://www.uml.org
http://www.omg.org/cgi-bin/doc?formal/03-03-01

1.4 Namespaces

1.4.1 Namespaces Defined

The following namespaces are defined by this specification.

Prefix	Namespace URI	Defining Specification / Description
rr-xsd	urn:oasis:names:tc:ebxml-regrep:profile:xsd	ebXML RegRep Profile for XML Schema

Table 1: Namespaces Defined

1.4.2 Namespaces Referenced

The following is a list of namespaces referenced by this specification. This list is not exhaustive and may be incomplete.

Namespace Prefix	Namespace URI	Defining Specification
lcm	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:4.0	ebXML RegRep Part 3: XML Schema file xsd/lcm.xsd Schema used by the LifecycleManager interface.
query	urn:oasis:names:tc:ebxml-regrep:xsd:query:4.0	ebXML RegRep Part 3: XML Schema file xsd/query.xsd Schema used by the QueryManager interface.
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0	ebXML RegRep Part 3: XML Schema file xsd/rim.xsd Schema used for information model objects specified by [regrep-rim-v4.0].
rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:4.0	ebXML RegRep Part 3: XML Schema file xsd/rs.xsd Common schema used by registry protocols defined by [regrep-rs-v4.0].
lcm	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:4.0	ebXML RegRep Part 3: XML Schema file xsd/lcm.xsd Schema used by the LifecycleManager interface.
query	urn:oasis:names:tc:ebxml-regrep:xsd:query:4.0	ebXML RegRep Part 3: XML Schema file xsd/query.xsd Schema used by the QueryManager interface.
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0	ebXML RegRep Part 3: XML Schema file xsd/rim.xsd Schema used for information model objects specified by [regrep-rim-v4.0].
rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:4.0	ebXML RegRep Part 3: XML Schema file xsd/rs.xsd Common schema used by registry protocols defined by [regrep-rs-v4.0].
spi	urn:oasis:names:tc:ebxml-regrep:xsd:spi:4.0	ebXML RegRep Part 3: XML Schema file xsd/spi.xsd Schema used by the service provider interfaces defined by [regrep-rs-v4.0].
xlink	http://www.w3.org/1999/xlink	XML Linking Language (XLink) Version 1.1
xsd	http://www.w3.org/2001/XMLSchema	XML Schema [XML Schema Part 1], [XML Schema Part 2] specification
xsi	"http://www.w3.org/2001/XMLSchema-instance	W3C XML Schema specification [XML Schema Part 1], [XML Schema Part 2].

Table 2: Namespaces Referenced

2 Conformance

This section defines the requirements for a server implementation claiming to conform to ebXML RegRep Profile for XML Schema specification.

TBD

3 Publish Profile

This section specifies how XML Schema documents are published to an ebXML RegRep server.

3.1 Metadata Representing an XML Schema File

A client publishes an XML Schema file by publishing an ExtrinsicObjectType instance as defined by **[regrep-rim-v4.0]** with the additional constraints specified below:

- The id attribute of the ExtrinsicObjectType instance representing the XML Schema document MUST be generated from the path component of the URL as specified by section 3.3 of [RFC1738] for the XML Schema document resource. This needs to be better defined???

- The lid attribute value must be the same as the id attribute This needs to be evaluated further on how it impacts versioning???

The following is an example from cataloging rim.xsd as specified in **[regrep-xsd-v4.0]:**

```
<ExtrinsicObjectType
  id="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  lid="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:XSD"

  ...
/>
```

3.2 Models for Managing XML Schema File

This specification supports the following two models for managing the XML Schema files:

- **External Repository** - XML Schema files are managed outside the Repository of the RegRep server and only their metadata is managed within the Registry of the RegRep server

- **Internal Repository** - XML Schema files are managed as repsoitory items within the Repository of the RegRep server and their corresponding metadata is managed within the Registry of the RegRep server

3.3 Publish Using External Repository

The ExtrinsicObjectType for the XML Schema file contains a rim:RepositoryItemRef sub-element to reference the XML Schema file by its URL.

The following is an example from cataloging rim.xsd as specified in **[regrep-xsd-v4.0]** using the external repository model:

```
<ExtrinsicObjectType
  id="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  lid="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:XSD"

  <RepositoryItemRef xlink:href="http://docs.oasis-open.org/regrep/regrep-
core/v4.0/cos01/xsd/rim.xsd"/>
  ...
/>
```

3.4 Publish Using Internal Repository

The ExtrinsicObjectType for the XML Schema file contains a rim:RepositoryItem sub-element to contain the XML Schema file.

The following is an example from cataloging rim.xsd as specified in **[regrep-xsd-v4.0]** using the internal repository model:

```
<ExtrinsicObjectType
  id="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  lid="regrep/regrep-core/v4.0/cos01/xsd/rim.xsd"
  objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:XSD"

  <RepositoryItem>...binary encoding of repository item...</RepositoryItem>
  ...
/>
```

3.5 Publishing XML Schema File Dependencies

When publishing an XML Schema file there is no requirement that schema files that are dependencies of the file being published be also published in the same transaction (or ever be published at all). This gives much flexibility in how XML Schema files get published. Note however that in order for the XSDDependenciesQuery and XSDUsageQuery to work properly a server SHOULD have all dependencies of a schema published as well.

3.6 Publishing XML Schema Using a Web Crawler

A specialized web crawler may be used to crawl a web site containing XML Schema documents. Such a web crawler may then publish the XML Schema files that it encounters to the RegRep server. This specification does not specify such a web-crawler. However, it is likely that such a web crawler would use the interfaces and protocols specified in **[regrep-rs-v4.0].**

4 Cataloging Profile

This section defines how XML Schema documents are cataloged when published to the RegRep server.

The XML Schema cataloger conforms to the Cataloger interface specified by **[regrep-rs-v4.0]**. The XML Schema cataloger catalogs the ExtrinsicObjhectType instance representing the XML Schema file and updates it as defined by this section. The cataloger processes the XML Schema file content regardless of whether it is represented as a RepositoryItem or RepositoryItemRef within the ExtrinsicObjectType instance for the XML Schema file.

4.1 Cataloging Name

The filename component of the path component within the URL for the XML Schema file MUST be cataloged into the ./rim:Name/rim:LocalizedString/@value where the rim:LocalizedString/@lang is left unspecified.

The following is an example from cataloging rim.xsd as specified in **[regrep-xsd-v4.0]:**

```
<Name>
      <rim:LocalizedString value="rim.xsd"/>
</Name>
```

4.2 Locator Slot

TBD??

4.3 Cataloging Other Document Nodes

The following table summarizes all nodes in an XML Schema document that are cataloged and specifies the rules for how they are cataloged. Each cataloging rule is followed by an example of the cataloged metadata produced by the cataloging rule. Unless otherwise specified the example is from cataloging rim.xsd as specified in **[regrep-xsd-v4.0].**

Source Node XPATH	Cataloging Rule
./xsd:annotation/xsd:documentation	• Each matched node maps to rim:LocalizedString element for the rim:Description element of ExtrinsicObjectType instance • The xsd:documentation/@xml:lang attribute, if present, is mapped to the rim:LocalizedString/@lang attribute • The content of the xsd:documentation node is mapped to rim:LocalizedString/@value attribute value

`<Description>`
`<LocalizedString xml:lang="en" value="The schema for OASIS ebXML Registry Information Model"/>`
`</Description>`	

./@targetNamespace	• Maps to a StringValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**targetNamespace**" • The slot's value is the targetNamespace node content

Source Node XPATH	Cataloging Rule

```
  <Slot name="urn:oasis:names:tc:ebxml-
regrep:profile:xsd:slot:targetNamespace">
        <SlotValue xsi:type="StringValueType">
            <Value>urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0</Value>
        </SlotValue>
 </Slot>
```

./@xsi:schemaLocation	• Maps to a MapValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**schemaLocationsMap**" • There is a Map/Entry for each schema location specified as content of the xsi:schemaLocation node • The EntryKey for each Entry is a StringValueType whose value is the schema namespace URI • The EntryValue for each Entry is a StringValueType whose value is the schema URL

```
    <Slot name="urn:oasis:names:tc:ebxml-
regrep:profile:xsd:slot:schemaLocationsMap">
        <SlotValue xsi:type="MapValueType">
            <Map>
                <Entry>
                    <EntryKey xsi:type="StringValueType">
                        <Value>http://www.w3.org/2001/XMLSchema</Value>
                    </EntryKey>
                    <EntryValue xsi:type="StringValueType">
                        <Value>http://www.w3.org/2001/XMLSchema.xsd</Value>
                    </EntryValue>
                </Entry>
            </Map>
        </SlotValue>
    </Slot>
```

./xsd:import	• All xsd:import nodes map to a single MapValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**importsMap**" • There is a Map/Entry for each xsd:import node • The EntryKey for each Entry is a StringValueType whose value is the value of ./xsd:import/@namespace child node • The EntryValue for each Entry is a StringValueType whose value is the value of ./xsd:import/@schemaLocation child node

The following example is from cataloging lcm.xsd as specified in **[regrep-xsd-v4.0].**

```
    <Slot name="urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:importsMap">
        <SlotValue xsi:type="MapValueType">
            <Map>
                <Entry>
                    <EntryKey xsi:type="StringValueType">
                        <Value>urn:oasis:names:tc:ebxml-
regrep:xsd:rs:4.0</Value>
                    </EntryKey>
                    <EntryValue xsi:type="StringValueType">
                        <Value>http://docs.oasis-open.org/regrep/regrep-
core/v4.0/cos01/xsd/rs.xsd</Value>
                    </EntryValue>
                </Entry>
                <Entry>
```

Source Node XPATH	Cataloging Rule

```
                    <EntryKey xsi:type="StringValueType">
                        <Value>urn:oasis:names:tc:ebxml-
regrep:xsd:rim:4.0</Value>
                    </EntryKey>
                    <EntryValue xsi:type="StringValueType">
                        <Value>http://docs.oasis-open.org/regrep/regrep-
core/v4.0/cos01/xsd/rim.xsd</Value>
                    </EntryValue>
                </Entry>
            </Map>
        </SlotValue>
    </Slot>
```

Source Node XPATH	Cataloging Rule
./xsd:include	• All xsd:include nodes map to a single CollectionValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**includes**" • The collectionType is "urn:oasis:names:tc:ebxml-regrep:**CollectionType:Set**" • There is a Element for each xsd:include node whose xsi:type is StringValueType • The Value for each element is the content of the xsd:include node

The following example is from cataloging http://schemas.opengis.net/ows/0.4.0/owsAll.xsd

```
<Slot name="urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:includes">
    <SlotValue xsi:type="CollectionValueType"
collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:Set">
        <Element xsi:type="StringValueType">
            <Value>http://schemas.opengis.net/ows/0.4.0/owsGetCapabilities.xs
d</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>http://schemas.opengis.net/ows/0.4.0/owsExceptionReport.xs
d</Value>
        </Element>
    </SlotValue>
</Slot>
```

Source Node XPATH	Cataloging Rule
./xsd:simpleType/@name | ./xsd:complexType/@name	• Only global (top-level) types are matched • All matched nodes map to a single CollectionValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**declaredTypes**" • The collectionType is “urn:oasis:names:tc:ebxml-regrep:CollectionType:Set” • There is an Element for each matched node whose type is StringValueType • The Value for the Element is the value of the matched node (the name)

The following example is from cataloging lcm.xsd as specified in **[regrep-xsd-v4.0].**

```
    <Slot name="urn:oasis:names:tc:ebxml-
regrep:profile:xsd:slot:declaredTypes">
        <SlotValue xsi:type="CollectionValueType"
collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:Set">
            <Element xsi:type="StringValueType">
                <Value>UpdateActionType</Value>
            </Element>
            <Element xsi:type="StringValueType">
                <Value>mode</Value>
            </Element>
```

Source Node XPATH	Cataloging Rule

```
        </SlotValue>
    </Slot>
```

./xsd:element/@name	• Only global (top-level) elements are matched • All matched nodes map to a single CollectionValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**declaredElements**" • The collectionType is "urn:oasis:names:tc:ebxml-regrep:CollectionType:Set" • There is an Element for each matched node whose type is StringValueType • The Value for the Element is the value of the matched node (the name)

The following example is from cataloging lcm.xsd as specified in **[regrep-xsd-v4.0].**

```
    <Slot name="urn:oasis:names:tc:ebxml-
regrep:profile:xsd:slot:declaredElements">
        <SlotValue xsi:type="CollectionValueType"
collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:Set">
            <Element xsi:type="StringValueType">
                <Value>RemoveObjectsRequest</Value>
            </Element>
            <Element xsi:type="StringValueType">
                <Value>SubmitObjectsRequest</Value>
            </Element>
            <Element xsi:type="StringValueType">
                <Value>UpdateObjectsRequest</Value>
            </Element>
        </SlotValue>
    </Slot>
```

./xsd:attribute/@name	• Only global (top-level) attributes are matched • All matched nodes map to a single CollectionValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**declaredAttributes**" • The collectionType is "urn:oasis:names:tc:ebxml-regrep:CollectionType:Set" • There is an Element for each matched node whose type is StringValueType • The Value for the Element is the value of the matched node (the name)

The following example is from cataloging http://schemas.opengis.net/xlink/1.0.0/xlinks.xsd

```
<Slot name="urn:oasis:names:tc:ebxml-
regrep:profile:xsd:slot:declaredAttributes">
    <SlotValue xsi:type="CollectionValueType"
collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:Set">
        <Element xsi:type="StringValueType">
            <Value>to</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>actuate</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>arcrole</Value>
```

Source Node XPATH	Cataloging Rule

```
        </Element>
        <Element xsi:type="StringValueType">
            <Value>title</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>role</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>label</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>show</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>from</Value>
        </Element>
        <Element xsi:type="StringValueType">
            <Value>href</Value>
        </Element>
    </SlotValue>
</Slot>
```

Source Node XPATH	Cataloging Rule
.//xsd:extension/@base | .//xsd:restriction/@base | .//xsd:element/@type | .//xsd:element/@ref | .//xsd:attribute/@type | .//xsd:attribute/@ref	• Catalogs all references to QNames that are in a namespace other than the targetNamespace for the XML Schema document. Thus, of the nodes that are matched, only those nodes are mapped that are QNames within a different namspace than the targetNamespace of the XML Schema document. In other words, local references are not cataloged. • All mapped nodes are mapped to a single CollectionValueType Slot with name: "urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:**references**" • The collectionType is "urn:oasis:names:tc:ebxml-regrep:**CollectionType:Set**" • There is an Element for each matched node whose type is StringValueType • The Value for the Element is the QName of the referenced type, element or attribute. The format of the QName MUST be as specified in 4.4 QName Representation Format

The following example is from cataloging lcm.xsd as specified in **[regrep-xsd-v4.0].**

```
    <Slot name="urn:oasis:names:tc:ebxml-regrep:profile:xsd:slot:references">
        <SlotValue xsi:type="CollectionValueType"
collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:Set">
            <Element xsi:type="StringValueType">
                <Value>{urn:oasis:names:tc:ebxml-
regrep:xsd:rim:4.0}RegistryObjectList</Value>
            </Element>
            <Element xsi:type="StringValueType">
                <Value>{urn:oasis:names:tc:ebxml-
regrep:xsd:rim:4.0}ValueType</Value>
            </Element>
            ...
        </SlotValue>
    </Slot>
```

4.4 QName Representation Format

A `QName` represents a **qualified name** as defined in the XML specifications: XML Schema Part2: Datatypes specification, Namespaces in XML, Namespaces in XML Errata.

This specification requires representing a QName in various situation in a specific format as described in this section.

The Qname is represented as: "{" + Namespace URI + "}" + local part. If the Namespace URI is null then only the local part is included in the representation.

For example to specify rim:RegistryObjectType as Qname use: {urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0}RegistryObjectType

5 Discovery Profile

This section specifies the canonical queries and canonical query functions that may be used to discovery Discovery of XML Schema documents.

5.1 Canonical Query: XSDDependenciesQuery

The canonical query XSDDependenciesQuery allows clients to discover XML Schema files that are used by the specified XML Schema file as a dependency. This query is useful to get a list of XML Schema files that are needed in order to support specified XML Schema file.

5.1.1 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
id	Specifies the id for an ExtrinsicObject that represents an XML Schema file.	string		0..1
levels	Number of dependency levels to search. Use 1 to match immediate dependencies, use positive integer N to match direct and indirect dependencies upto N levels. Use -1 or 0 to match all direct and indirect dependencies.	integer	1	0..1

5.1.2 Query Semantics

- The id parameter MUST be specified
- The query traverses the dependency tree for the XML Schema file matched by the id parameter up to the specified number of levels and includes each XML Schema file as a dependency matched by the query
- This query returns a set of ExtrinsicObjects where each ExtrinsicObject represents an XML Schema file matches by the query

Issue: The query currently does not preserve dependency hierarchy as a tree since matched objects are returned as a flat list. Should this be improved??

5.2 Canonical Query: XSDDiscoveryQuery

The canonical query XSDDiscoveryQuery allows clients to discover XML Schema files matching specified parameters.

5.2.1 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
declaresAttribute	Specifies a regular expression $declaresAttribute. Matches XML Schema files that have a node that matches the following XPATH expression ./xsd:attribute/@name[fn:matches(., $declaresAttribute)]	string		0..1
declaresElement	Specifies a regular expression $declaresElement. Matches XML Schema files that have a node	string		0..1

	that matches the following XPATH expression: ./xsd:element/@name[fn:matches(., $declaresElement)]			
declaresType	Specifies a regular expression $declaresType. Matches XML Schema files that have a node that matches the following XPATH expression: ./xsd:attribute/@name[fn:matches(., $declaresType)] | ./xsd:complexType/@name[fn:matches(., $declaresType)]	string		0..1
description	Specifies a regular expression $description. Matches XML Schema files that have a node that matches the following XPATH expression: ./xsd:documentation[fn:matches(., $description)]	string		0..1
importsNamespace	Specifies a regular expression $importsNamespace. Matches XML Schema files that have a node that matches the following XPATH expression: ./xsd:import/@namespace[fn:matches(., $importsNamespace)]	string		0..1
includesSchema	Specifies a regular expression $includesSchema. Matches XML Schema files that have a node that matches the following XPATH expression: ./xsd:include/@schemaLocation[fn:matches(., $includesSchema)]	string		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
name	Specifies a regular expression that matches the filename of an XML Schema file	string		0..1
targetNamespace	Specifies a regular expression $targetNamespace. Matches XML Schema files that have a node that matches the following XPATH expression: ./@targetNamespace[fn:matches(., $targetNamespace)]	string		0..1
xsiSchemaLocationNamespace	Specifies a regular expression $xsiSchemaLocationNamespace. Matches XML Schema files that have a node that matches the following XPATH expression: ./@xsi:schemaLocation[fn:matches(., fun:string-join($xsiSchemaLocationNamespace, '\s'))]	string		0..1
xsiSchemaLocationValue	Specifies a regular expression $xsiSchemaLocationValue. Matches XML Schema files that have a node that matches the following XPATH expression:	string		0..1

	./@xsi:schemaLocation[fn:matches(., fun:string-join('\s', $xsiSchemaLocationValue))]			

5.2.2 Query Semantics

- This query has several optional parameters
- Each parameter implies a predicate within the underlying query
- Predicates for each supplied parameter are combined using with an implicit LOGICAL AND if matchOnAnyParameter is unspecified or false. If it is specified as true then predicates for each supplied parameters are combined using a LOGICAL OR
- If an optional parameter is not supplied then its corresponding predicate MUST NOT be included in the underlying query
- This query returns a set of ExtrinsicObjects where each ExtrinsicObject represents an XML Schema file matches by the query

5.3 Canonical Query: XSDUsageQuery

The canonical query XSDUsageQuery allows clients to discover XML Schema files that use a particular XML Schema file, or alternatively, use a type, attribute or element in a schema file. This query is useful in determining which XML Schema files will be impacted when one makes a change to an XML Schema file or the definition of a type, attribute or element within an XML Schema file

5.3.1 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
id	Specifies a regular expression $id. Matches XML Schema files that reference an XML Schema file whose URL matches the specified id.	string		0..1
qname	Specifies a regular expression $qname. Matches XML Schema files that have a reference to a Qname that matches $qname. The Qname represnetation format that the regular expression expects to match MUST be as specified in 4.4 QName Representation Format	string		0..1

5.3.2 Query Semantics

- Either the id or qname parameter MUST be specified
- Both parameters MUST NOT be specified
- This query returns a set of ExtrinsicObjects where each ExtrinsicObject represents an XML Schema file matches by the query

Appendix A. Acknowledgements

The following individuals have contributed significantly towards the creation of this specification and are gratefully acknowledged.

Technical Committee Contributors:

- Kathryn Breininger, Boeing
- Carl Mattocks, MetLife
- Farrukh Najmi, Wellfleet Software
- Oliver Newell, MIT Lincoln Labs
- Nikola Stojanovic, Individual

External Contributors:

- Kohsuke Kawaguchi, Individual
- Brett Levasseur, MIT Lincoln Labs
- Aleksei Valikov, Disy Informationssysteme GmbH

Appendix B. Revision History

Revision	Date	Editor	Changes Made
CSD01-1	03 January 2012	Farrukh Najmi, Oliver Newell	Initial version for 4.0.