INTERNATIONAL STANDARD

ISO 15000-3

First edition
2023-01

Electronic business eXtensible Markup Language (ebXML) —

Part 3: Registry and repository

Reference number
ISO 15000-3:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the OASIS ebXML (Electronic business eXtensible Markup Language) Registry Technical Committee (as the “OASIS ebXML RegRep Version 4.0 specification”) and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration* and adopted under the "fast-track procedure".

A list of all parts in the ISO 15000 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

ebXML RegRep (registry and repository) is a standard defining service interfaces, protocols and information model for an integrated registry and repository. The repository stores electronic content while the registry stores metadata that describes the content in the repository.

To explain what an ebXML RegRep is we use the following familiar analogy. The ebXML RegRep is to electronic content, what your local library is to books and other published content. To make this analogy clearer one could look at comparisons enumerated in Table 1:

Your Local Library	ebXML RegRep
Manages books and all types of published material	Manages all types of electronic content
Has book shelves containing books and other published material	Has a "repository" containing all types of electronic content
Has a card catalog that describes the published material that is available in the book shelves	Has a "registry" that describes the electronic content that is available in the repository
Multiple libraries can voluntarily participate in a cooperative network and offer a unified service	Multiple ebXML RegRep instances can voluntarily participate in a cooperative network and offer a unified service

Table 1: ebXML RegRep comparison with your local library

An ebXML RegRep is capable of storing any type of electronic content such as XML documents, text documents, images, sound and video. Instances of such content are classified as “RepositoryItem” instances. Repository items are stored in a content *repository* provided by ebXML RegRep.

In addition to repository items ebXML RegRep also stores standardized metadata. Instances of such metadata are classified as “RegistryObject” instances (or one of its sub-types, as described later in this document). Registry objects are stored in a *registry* provided by ebXML RegRep. A registry object is like a card in the card catalog. All registry objects conform to a standard just like the cards in the card catalog conform to a standard. Every repository item shall have a registry object that describes it, just like every book must have a card in the card catalog. At the same time there are registry objects that do not describe any repository item and serve the purpose of defining capabilities of an ebXML RegRep instance.

This document describes Electronic business eXtensible Markup Language (ebXML) Part 3: Registry and Repository. The main clauses are:

a) The Registry Information Model *(*ebRIM*)* which defines the types of metadata that can be stored in an ebXML RegRep server (Clause 4).
b) The Registry Services and Protocols (ebRS) which define the services provided by an ebXML RegRep server and the protocols used by clients to interact with these services (Clause 5).
c) Conformance to ebXML Registry and Repository (Clause 6).

This document also includes the following annexes:

a) Annex A lists the standard exceptions that may be returned by various protocols defined in this document.
b) Annex B provides a listing of namespaces that are defined in this document.
c) Annex C provides a listing of namespaces that are referenced in this document.

Separately provided are:

a) A set of XML schema files at https://standards.iso.org/iso/15000/-3/ed-1/en/xsd/ define XML schema types and elements referenced in the core specification clauses.

b) A set of WSDL files at https://standards.iso.org/iso/15000/-3/ed-1/en/wsdl/ define the WSDL interfaces, bindings and services referenced in the core specification clauses.

c) A set of XML files at https://standards.iso.org/iso/15000/-3/ed-1/en/xml/ that contain the canonical data that shall be supported within every ebXML RegRep server.

Electronic business eXtensible Markup Language (ebXML) —Part 3: Registry and Repository

1 Scope

This part of ISO 15000 specifies service interfaces, protocols and information model for an integrated registry and repository. The repository stores electronic content while the registry stores metadata that describes the content in the repository.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2828, *Internet Security Glossary*, 2000. Edited by R. Shirey. https://www.rfc-editor.org/info/rfc2828.

ISO. Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation, Schematron, International Standard ISO/IEC 19757-3, Geneva, Switzerland : ISO.

OASIS. *Web Services Security: Web Services Security: SAML Token Profile 1.1*. 1 February 2006. http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SAMLTokenProfile.pdf

OASIS. *Web Services Security: WS-Security Kerberos Token Profile 1.1*, February 2006. http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf

OASIS. *Web Services Security: SOAP Message Security 1.0*, 2004. Edited by Anthony Nadalin, et al Available from http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

OASIS. *Web Services Security UsernameToken Profile 1.0*, 2004. Edited by P. Hallam-Baker, et al. Available from http://docs.oasis-open.org/wss/2004/01/.

OASIS. *Web Services Security X.509 Certificate Token Profile*, 2004. Edited by P. Hallam-Baker, et al Available from http://docs.oasis-open.org/wss/2004/01/.

OASIS. *Web Services Security: SOAP Message Security 1.1*, 2005. Edited by Anthony Nadalin, et al Available from http://docs.oasis-open.org/wss/v1.1/.

OASIS. *Web Services Security UsernameToken Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from http://docs.oasis-open.org/wss/v1.1/.

OASIS. *Web Services Security X.509 Certificate Token Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from http://docs.oasis-open.org/wss/v1.1/.

OASIS. *eXtensible Access Control Markup Language (XACML) Version 2.01*. Edited by T. Moses. Available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

WORLD WIDE WEB CONSORTIUM (W3C). *SOAP Version 1.2 Part 1: Messaging Framework*. M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen, Editors, W3C Recommendation, June 24, 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

WORLD WIDE WEB CONSORTIUM (W3C). *SOAP Version 1.2 Part 2: Adjuncts*, M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen, Editors, W3C Recommendation, June 24, 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - Core*, M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, May 9, 2006, http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. Latest version available at http://www.w3.org/TR/ws-addr-core/.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - SOAP Binding*, M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, May 9, 2006, http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - Metadata*, M. Gudgin, M. Hadley, T. Rogers, Ü. Yalçinalp, Editors, W3C Recommendation, September 4, 2007, http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Description Language (WSDL) Version 1.1*, E. Christinsen et Al., Editors, W3C Note, March, 2001, https://www.w3.org/TR/2001/NOTE-wsdl-20010315.

WORLD WIDE WEB CONSORTIUM(W3C). *XML Schema Part 1: Structures Second Edition*, 2004. Edited by H. Thompson, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

WORLD WIDE WIDE CONSORTIUM(W3C). *XML Schema Part 2: Datatypes Second Edition*, 2004. Edited by P. Biron, et al. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

WORLD WIDE WEB CONSORTIUM. *Extensible Markup Language (XML) 1.0* (Second Edition), 2000. Edited by T. Bray, et al. Available from http://www.w3.org/TR/2000/REC-xml-20001006.

WORD WIDE WEB CONSORTIUM. *XSL Transformations (XSLT) Version 1.0*, 1999. Edited by J. Clark.. Available from http://www.w3.org/TR/1999/REC-xslt-19991116.

3 Terms and definitions

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp
— IEC Electropedia: available at https://www.electropedia.org/

3.1
access control
authority definitions and their enforcement for various types of actors regarding access to a *registry object* (3.21)

3.2
association
registry object (3.21) that represents directional relationship between two *registry objects* (3.21)

3.3
auditable event
long-term record of what actions caused changes in the state of a *registry object* (3.21)

3.4
canonical artefact
set of *registry object* (3.21) values defined by the standard

3.5
cataloging
registry service (3.23) that creates and stores additional metadata related to a *registry object* (3.21)

3.6
classification
registry object (3.21) that classifies another *registry object* (3.21) by assigning a *classification node* (3.7) to it

3.7
classification node
registry object (3.21) that represents a specific value that is part of a *classification scheme* (3.8)

3.8
classification scheme
registry object (3.21) formed as a parent-child hierarchy of *classification nodes* (3.7)

3.9
external identity
identity of a *registry object* (3.21) as per known identification scheme

3.10
federation
set of *registry server* (3.22) instances that form a loosely coupled union

3.11
governance
process of controlling the lifecycle of *registry objects* (3.21)

3.12
identity
globally unique identity that is different between different versions of a *registry object* (3.21)

3.13
logical identity
globally unique identity of a *registry object* (3.21) that does not change during its lifetime

3.14
messaging
secure and reliable exchange of messages between a *registry client* (3.20) and a *registry server* (3.22)

3.15
native language support
support to express properties of a *registry object* (3.21) in different languages

3.16
notification
registry object (3.21) sent to a *registry client* (3.20) as an effect of registry events matching a relevant *subscription* (3.28)

3.17
object types classification scheme
canonical classification scheme that represents types of objects that a *registry server* (3.22) supports

3.18
provenance
set of data and processes that represents ownership and history of custodianship of a *registry object* (3.21)

3.19
query definition
*registry objec*t (3.21) that represents query that could be invoked by a *registry client* (3.20)

3.20
registry client
client that communicates with a *registry server* (3.22) via ebXML RegRep protocols

3.21
registry object
metadata stored in a *registry server* (3.22)

3.22
registry server
server that provides *registry services* (3.23)

3.23
registry service
service that provides storage and discovery of electronic artefacts (repository) and their metadata (registry)

3.24
replication
registry service (3.23) where a copy of a *registry object* (3.21) is stored and maintained in another *registry server* (3.22)

3.25
repository item
electronic artefact stored in a repository

3.26
service
*registry objec*t (3.21) that represents services offered by an actor

3.27
slot
typed name-value property that provides extensibility of a *registry objec*t (3.21)

3.28
subscription
*registry objec*t (3.21) that represents the interest of a *registry client* (3.20) interest in receiving *notifications* (3.16) when certain events happen in a *registry server* (3.22).

3.28
validation
registry service (3.23) that checks whether a *registry object* (3.21) is valid or not

3.30
version
distinct state of a *registry object* (3.21)

4 Registry information model

4.1 Introduction

4.1.1 Overview

This clause defines the ebXML RegRep registry information model (ebRIM).

4.1.2 XML Schema

The ebXML Registry Information Model is defined as an XML Schema in https://standards.iso.org/iso/15000/-3/ed-1/en/xsd/rim.xsd. It defines the metadata types and their relationships within this specification.

4.1.3 Information model types: inheritance view

The central type in the model is the RegistryObjectType. An instance of RegistryObjectType represents an ebRIM metadata object.

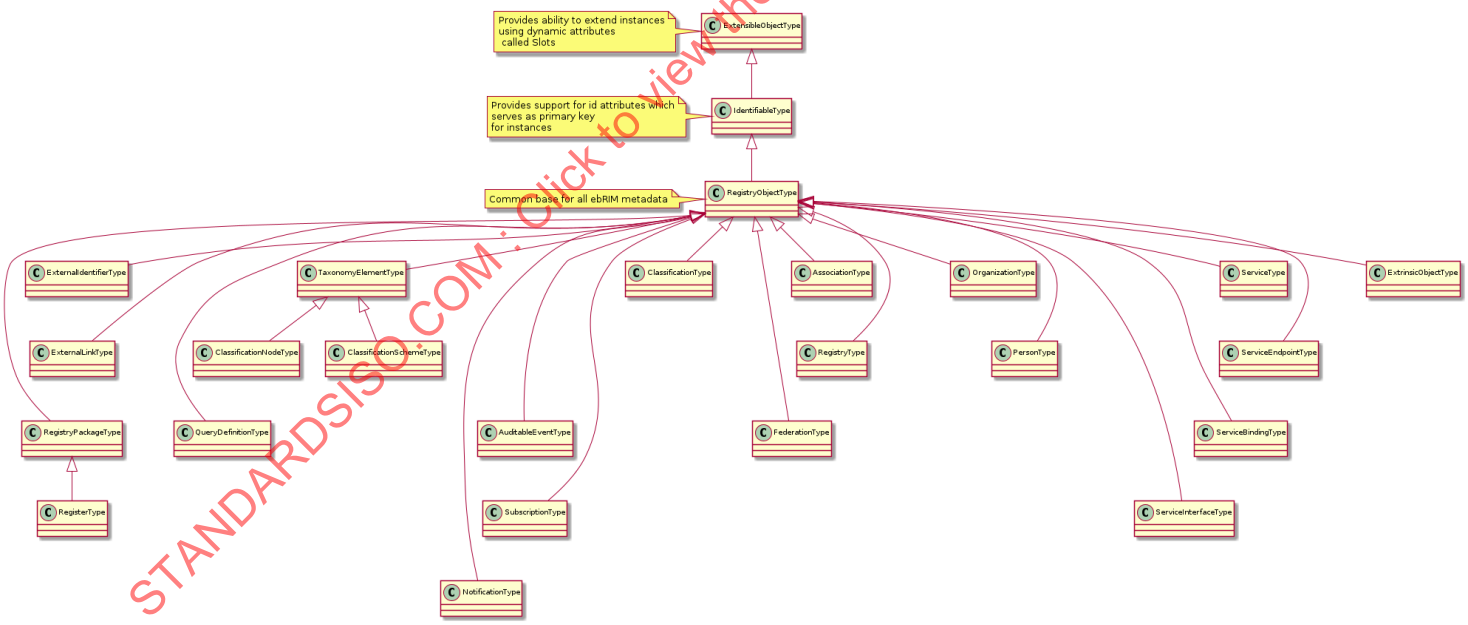

Figure 1 Information model inheritance view

Figure 1 shows the inheritance or "Is-A" relationships between the various types derived from RegistryObjectType in the information model. Note that it does not show the other types of relationships, such as "Has-A" relationships, as they will be presented in subsequent diagrams. The attributes and elements of each type are also not shown to conserve page space. Detailed description of attributes and elements of each type will be displayed in tabular form within the detailed description of each type.

4.1.4 Extending ebRIM

The XML Schema for ebRIM uses XML Schema type substitution feature to allow use of schema type extensions.

A deployment or profile specification of ebXML RegRep may define new types that extend the types defined in this specification as long as the XML Schema for ebRIM supports such extension.

A server may support the schema type extensibility feature. The following requirements are defined for a server that supports the schema type extensibility feature:

a) The server protocols as defined by the registry services (see clause 5) shall support extended types in a manner equivalent to pre-defined types. Specifically they shall support submit, update, versioning and removal of extended types derived directly or indirectly from RegistryObjectType

b) The server shall be able to faithfully persist instances of extended types including all extension attributes and elements without any information loss

c) The server shall be able to faithfully return instances of extension types including extension attributes and elements within a query response without any information loss

d) This specification does not prescribe how a server may support the addition of new extension types to the server

4.1.5 Canonical ClassificationSchemes

ClassificationSchemes are defined in detail in the Classification Information Model (see clause 4.4). They are used by the specification for a wide variety of purposes within this specification.

This specification uses several standard ClassificationSchemes referred to as *canonical ClassificationSchemes*. The values defined within canonical ClassificationSchemes are defined using standard ClassificationNodes that are referred to as *canonical ClassificationNodes*.

The directory https://standards.iso.org/iso/15000/-3/ed-1/en/xml/minDB/ contains the canonical ClassificationSchemes defined by this specification. The canonical ClassificationSchemes and ClassificationNodes are typically described using the rim:Description element within these files.

These canonical ClassificationSchemes shall be present in all conforming ebXML RegRep servers. These Canonical ClassificationSchemes may be extended by adding additional ClassificationNodes. However, a ClassificationNode defined normatively in the canonical ClassificationScheme definitions shall not be modified within a registry. In particular they shall preserve their canonical id attributes in all servers.

4.2 Core Information Model

4.2.1 Overview

The core information model is centered around the RegistryObjectType type as shown in Figure 2 below. Each type will be defined in detail in subsequent clauses.

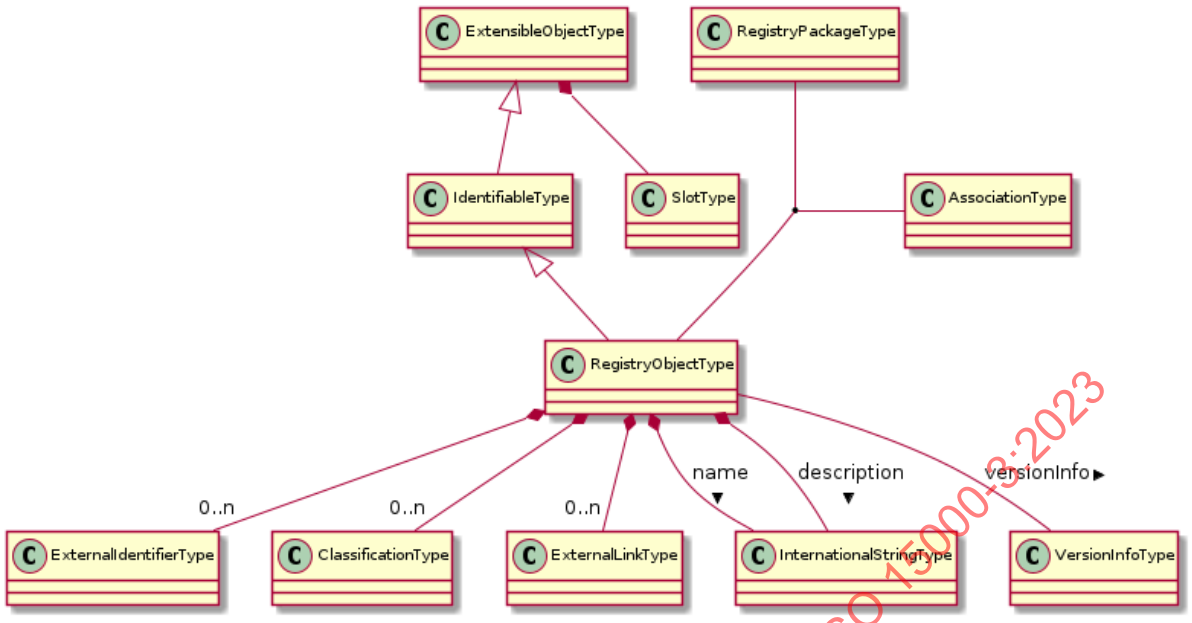

Figure 2 Core Information Model

4.2.2 InternationalStringType

4.2.2.1 Overview

The InternationalStringType type is used throughout the schema whenever a textual value needs to be represented in one or more local languages.

The InternationalStringType has a sequence of LocalizedString instances, where each LocalizedString instance is specific to a particular locale.

4.2.2.2 XML schema type definition

```
<complexType name="InternationalStringType">
  <sequence>
    <element name="LocalizedString" type="tns:LocalizedStringType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
```

EXAMPLE:

```
<rim:Name>
  <rim:LocalizedString
    xml:lang="en-US" value="freebXMLRegistry"/>
</rim:Name>
```

4.2.2.3 Description

The InternationStringType is defined in Table 2.

Node	Type	Cardinality	Default Value	Specified By	Mutable

LocalizedString	LocalizedStringType	0..*		Client	Yes

Table 2 InternationalStringType

Element LocalizedString - An InternationalStringType instance may have zero or more LocalizedString elements where each defines a string value within a specific local language

4.2.3 LocalizedStringType

4.2.3.1 Overview

This type allows the definition of a string value using the specified local language. It is used within the InternationalStringType as the type of the LocalizedString sub-element. The value of the xml:lang attribute shall be set to a tag for identifying a language as defined in RFC 4646. Note that the character set for all LocalizedStringType instances in an XML document is defined by the charset attribute within the *Content-Type* mime header for the XML document as shown in example below:

4.2.3.2 XML schema type definition

```
<complexType name="LocalizedStringType">
  <attribute ref="xml:lang" default="en-US" use="optional"/>
  <attribute name="value" type="tns:FreeFormText" use="required"/>
</complexType>
```

EXAMPLE:

```
<rim:LocalizedString
   xml:lang="en-US" value="freebXMLRegistry"/>
```

4.2.3.3 Description

The LocalizedStringType is described in Table 3.

Node	Type	Cardinality	Default Value	Specified By	Mutable
xml:lang	xs:language	0..1	en-US	Client	Yes
value	rim:FreeFormText	1		Client	Yes

Table 3 LocalizedStringType

a) Attribute xml:lang - Each LocalizedStringType instance may have a *xml:lang* attribute that specifies the language used by that LocalizedStringType instance. The xml:lang attribute and legal values for it are defined by Extensible Markup Language (XML) .

b) Attribute value - Each LocalizedStringType instance shall have a *value* attribute that specifies the string value used by that LocalizedStringType instance

4.2.4 ExtensibleObjectType

4.2.4.1 Overview

This type is the root type for most other types in rim.xsd. It allows extension properties called slots to be added to instances of this type using Slot sub-elements.

4.2.4.2 XML schema type definition

```
<complexType name="ExtensibleObjectType" abstract="true">
```



```
  <sequence>
    <element name="Slot" type="tns:SlotType" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

EXAMPLE: The following example shows how an OrganizationType instance which is of type ExtensibleObjectType may use Slot sub-elements to define a tax payer id for the organization.

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
  <rim:Slot name="urn:foo:slot:taxPayerId">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>1234567890</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
  ...
</rim:RegistryObject>
```

4.2.4.3 Description

The ExtensibleObjectType is described in Table 4.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Slot	SlotType	0..*			

Table 4 ExtensibleObjectType

Element Slot – Allows an extension property to be added to any ExtensibleObjectType instance

4.2.5 SlotType

4.2.5.1 Overview

Base Type: ExtensibleObjectType

The SlotType represents an extensible property for a RegistryObjectType instance . It can contain any type of information that may be represented in an XML document. It is an important extensibility mechanism with ebRIM.

A SlotType instance has a name and a value. The value is of type ValueType. ValueType is abstract and has several concrete sub-types defined within this specification.

Note that SlotType extends ExtensibleObjectType which means that a SlotType element may itself have SlotType sub-elements.

4.2.5.2 XML schema type definition

```
<complexType name="SlotType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="SlotValue" type="tns:ValueType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
      <attribute name="name" type="tns:LongText" use="required"/>
      <attribute name="type" type="tns:LongText" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

The following example shows how a GML geometry value may be specified as a Slot.

EXAMPLE

```
<rim:Slot
  name="geographicBoundingBox"
  type="urn:ogc:def:dataType:ISO-19107:GM_Geometry">
  <rim:SlotValue xsi:type="rim:AnyValueType">
    <gml:Envelope srsName="urn:ogc:def:crs:OGC:2:WGS84">
      <!--BB: POLYGON((0 0, 30 0, 30 30, 0 30, 0 0))-->
      <gml:lowerCorner>0 0</gml:lowerCorner>
      <gml:upperCorner>30 30</gml:upperCorner>
    </gml:Envelope>
  </rim:SlotValue>
</rim:Slot>
```

4.2.5.3 Description

Table 5 shows the description of SlotType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Name	LongText	1		Client	Yes
SlotValue	ValueType	0..1		Client	Yes
type	LongText	0..1		Client	Yes

Table 5 SlotType

a) Attribute name – The name of this SlotType instance. The name of the slot shall be unique within the universe of slot names for all sibling slots within its parent object

b) Element SlotValue – This element is the container for the actual value for the SlotType instance

c) Attribute type – A string that specifies the type for the SlotType instance. The type may be used to assign a category for the SlotType instance

4.2.6 ValueType

4.2.6.1 Overview

This type is abstract base type for the value of a SlotType instance.

4.2.6.2 XML schema type definition

```
<complexType name="ValueType" abstract="true">
</complexType>
```

4.2.6.3 Description

The ValueType is an abstract base type that does not define any attributes or elements. This specification defines several concrete sub-types that extend ValueType

a) AnyValueType – This concrete sub-type of ValueType is used as a container for any well-formed XML element value in any namespace

b) BooleanValueType - This concrete sub-type of ValueType is used as a container for a boolean value

c) CollectionValueType - This concrete sub-type of ValueType is used as a container for a collection of values. It may be used to represent a SlotValue that is a collection of values where each value is represented by a ValueType instance

Attribute collectionType – Defines the type of collection for the CollectionValueType. Must be an objectReferenceType that references a ClassificationNode in the canonical ClassificationScheme CollectionTypeScheme. A server shall enforce the following semantics associated with the following canonical collection types:

1) List – Server shall maintain the order of the values in the collection
2) Set – Server shall not allow duplicate values in the collection
3) Sorted Set – Server shall not allow duplicate values in the collection and shall maintain a sort order according to the alphanumeric ordering of its elements according to the default locale associated with the server
4) Bag – Server shall allow duplicate values and may not maintain order of values

a) DateTimeValueType - This concrete sub-type of ValueType is used as a container for a dateTime value

b) DurationValueType - This concrete sub-type of ValueType is used as a container for a duration value

c) FloatValueType - This concrete sub-type of ValueType is used as a container for a float value

d) IntegerValueType - This concrete sub-type of ValueType is used as a container for an integer value

e) InternationalStringValueType - This concrete sub-type of ValueType is used as a container for an InternationalSringType value capable of holding strings in multiple locales

f) MapValueType - This concrete sub-type of ValueType is used as a container for a map value. A map consists of Entry sub-elements where each Entry consists of an EntryKey and EntryValue both of which are of type ValueType

g) SlotValueType – This concrete sub-type of ValueType is used as a container for a SlotType value

h) StringValueType – This concrete sub-type of ValueType is used as a container for a string value

i) VocabularyTermValueType - This concrete sub-type of ValueType is used as a container for a VocabularyTermType value. It is used to reference a term in some externally defined coded vocabulary (e.g. Dublin Core)

4.2.7 IdentifiableObjectType

4.2.7.1 Overview

Base Type: ExtensibleObjectType

This type extends ExtensibleObjectType and allows its instances to be uniquely identifiable by a unique id.

4.2.7.2 XML schema type definition

```
<complexType name="IdentifiableType" abstract="true">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="id" type="string" use="required"/>
    </extension>
  </complexContent>
```

```
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
  ...
</rim:RegistryObject>
```

4.2.7.3 Description

Table 6 shows the description of IdentifiableObjectType.

Node	Type	Cardinality	Default Value	Specified By
id	xs:string	1		Client

Table 6 IdentifiableObjectType

Attribute id – Specifies the unique identifier for an IdentifiableType instance.

4.2.8 RegistryObjectType

4.2.8.1 Overview

Base Type: IdentifiableType

This type extends IdentifiableObjectType and is the common base type for all *queryable* metadata elements in ebRIM.

4.2.8.2 XML schema type definition

```
<complexType name="RegistryObjectType">
  <complexContent>
    <extension base="tns:IdentifiableType">
      <sequence>
        <element name="Name" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
        <element name="Description" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
        <element name="VersionInfo" type="tns:VersionInfoType" minOccurs="0"
maxOccurs="1"/>
        <element name="Classification" type="tns:ClassificationType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="ExternalIdentifier" type="tns:ExternalIdentifierType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="ExternalLink" type="tns:ExternalLinkType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="lid" type="string" use="optional"/>
      <attribute name="objectType" type="tns:objectReferenceType" use="optional"/>
      <attribute name="owner" type="string" use="optional"/>
      <attribute name="status" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

4.2.8.3 Description

Table 7 shows the description of RegistryObjectType.

Node	Type	Cardinality	Default Value	Specified By	Mutable

Classification	ClassificationType	0..*		Client	Yes
Description	InternationalStringType	0..1		Client	Yes
ExternalIdentifer	ExternalIdentifierType	0..*		Client	Yes
ExternalLink	ExternalLinkType	0..*		Client	Yes
lid	string	0..1		Client or Server	No
Name	InternationalStringType	0..1		Client	Yes
objectType	objectReferenceType	0..1		Client or Server	No
owner	string	0..1		Server	Yes
status	objectReferenceType	0..1		Server	Yes
VersionInfo	VersionInfoType	0..1		Server	No

Table 7 RegistryObjectType

a) Element Classification - A RegistryObjectType instance may have zero or more ClassificationType instances that are composed within the RegistryObject. A ClassificationType instance classify the RegistryObject using a value within a ClassificationScheme

b) Element Description - A RegistryObjectType instance may have textual description in a human readable and user-friendly form. This element is of type InternationalStringType and therefor capable of containing textual values in multiple local languages and character sets.

c) Element ExternalIdentifier - A RegistryObjectType instance may have zero or more ExternalIdentifier instances that are composed within the RegistryObject. An ExternalIdentifier instance represents an alternate identifier for the RegistryObject in addition to the identifier specified by its id attribute value.

d) Attribute lid - A RegistryObjectType instance shall have a lid (Logical Id) attribute. The lid is used to refer to a logical RegistryObject in a version independent manner.

   1) All versions of a RegistryObject shall have the same value for the lid attribute. Note that this is in contrast with the id attribute that shall be unique for each version of the same logical RegistryObject.

   2) The lid attribute shall be specified by the client when creating the original version of a RegistryObject.

   3) The lid attribute specified when submitting the original version of a RegistryObject shall be globally unique and shall not be already in use as lid by another object.

e) Element Name - A RegistryObjectType instance may have a human readable name. The name does not need to be unique with respect to other RegistryObjectType instances. This element is of type InternationalStringType and therefor capable of containing textual values in multiple local languages and character sets.

f) Attribute objectType - A RegistryObjectType instance has an *objectType* attribute.

   1) The value of the objectType attribute shall be a reference to a ClassificationNode in the canonical ObjectType ClassificationScheme.

2) A server shall support the object types as defined by the canonical ObjectType ClassificationScheme. The canonical ObjectType ClassificationScheme may easily be extended by adding additional ClassificationNodes to the canonical ObjectType ClassificationScheme.

3) The *objectType* attribute shall be assigned by the server for all RegistryObjectType instances that are not instances of ExtrinsicObjectType.

4) The *objectType* attribute may be assigned by the client for all RegistryObjectType instances that are instances of ExtrinsicObjectType

5) If the client does not specify an objectType for an ExtrinsicObject then the server shall set its value to the id of the ClassificationNode representing ExtrinsicObject within the canonical ObjectType ClassificationScheme.

6) A server shall set the correct objectType on a RegistryObject when returning it as a response to a client request.

h) Attribute owner – Specifies the identifier associated with the registered user that owns the RegistryObjectType instance. It is used for access control and may be referenced within custom access control policies.

i) Attribute status - A RegistryObjectType instance shall have a life cycle status indicator. The status is assigned by the server. Profiles may define additional status values if needed as slots on the RegistryObjectType instance. Such slots should have a type attribute with value "urn:oasis:names:tc:ebxml-regrep:rim:Slot:type:status".

1) A server shall set the correct status on a RegistryObject when returning it as a response to a client request.

2) A client should not set the status on a RegistryObject when submitting the object as this is the responsibility of the server.

3) A server shall ignore the status on a RegistryObject when it is set by the client during submission or update of the object.

4) The value of the status attribute should be a reference to a ClassificationNode in the canonical StatusType ClassificationScheme.

5) A Registry shall support the status types as defined by the StatusType ClassificationScheme. The canonical StatusType ClassificationScheme may easily be extended by adding additional ClassificationNodes to the canonical StatusType ClassificationScheme.

j) Element VersionInfo - Provides information about the specific version of a RegistryObjectType instance. The VersionInfo element is set by the server.

A server shall set a VersionInfo element for a RegistryObjectType instance. The VersionInfo element shall contain a versionName attribute whose value shall be unique for all versions of that logical RegistryObjectType.

4.2.9 VersionInfoType

4.2.9.1 Overview

This type represents information about a specific version of a RegistryObject or RepositoryItem. It is used as type for the RegistryObjectType/VersionInfo and ExtrinsicObjectType/ContentVersionInfo elements in the rim.xsd schema.

4.2.9.2 XML schema type definition

```
<complexType name="VersionInfoType">
  <attribute name="versionName"
    type="tns:String16" use="optional" default="1.1"/>
  <attribute name="userVersionName" type="string" use="optional"/>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"  ...>
  ...
  <rim:VersionInfo versionName="1.1" userVersionName="1.1"/>
  ...
</rim:RegistryObject>
```

4.2.9.3 Description

Table 8 shows the description of VersionInfoType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
userVersionName	LongText	0..1		Client	Yes
versionName	String16	0..1		Server	No

Table 8 VersionInfoType

a) Attribute userVersionName - Represents a client-specified version name associated with the VersionInfo for a specific RegistryObject version

   1) A client may directly provide a value for the userVersionName attribute when submitting or updating an object

   2) A server shall persist any client specified userVersionName for an object without altering it in any form

b) Attribute versionName - Represents the registry assigned version name identifying the VersionInfo for a specific RegistryObject version.

   1) The value for this attribute should not be specified by the client

   2) A server may silently ignore the value for this attribute if specified by the client

   3) The value for this attribute shall be automatically generated by the server and shall be defined for RegistryObjectType instances returned by server responses. The server is free to choose any scheme for generating the value for this attribute as long as the value is uniquely identifies a version for objects that have the same lid attribute value.

4.2.10 objectReferenceType

4.2.10.1 Overview

Base Type: xs:string

A RegistryObjectType instance typically has several references to other RegistryObjectType instances. These references are represented by attributes of type rim:objectReferenceType within the XML Schema for ebXML RegRep.

The RegistryObjectType instance that has a reference to another RegistryObjectType instance is referred to as the *reference source* object. The RegistryObjectType instance that is being referenced is referred to as the *reference target* object.

4.2.10.2 XML schema type definition

```
<simpleType name="objectReferenceType">
  <restriction base="string"/>
</simpleType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  primaryContact="urn:acme:person:Danyal" ...>
  ...
</rim:RegistryObject>
```

4.2.10.3 Description

4.2.10.3.1 Local and Remote References

The reference source and target objects may be in different ebXML RegRep servers. In such cases the reference is referred to as a *remote reference*.

4.2.10.3.2 Static and Dynamic References

When a reference is fixed to a specific reference target it is referred to as a *static reference*. This specification also supports a *dynamic reference* where the reference target is determined dynamically by a query at the time the reference is resolved. Such a reference is referred to as a *dynamic reference*.

Both static and dynamic references may be to a local or remote object. Static references to local reference targets are the most typical form of reference.

4.2.10.3.3 Encoding of objectReferenceType

A client shall specify values for reference attributes of type objectReferenceType to be encoded as described below:

a) A static reference to a local reference target should be encoded as the value of the id attribute of the reference target.

The following example shows the reference attribute named primaryContact within Organization element. Its value is the value of the id attribute of a Person element.

EXAMPLE 1:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  primaryContact="urn:acme:person:Danyal" ...>
  ...
</rim:RegistryObject>

<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
</rim:RegistryObject>
```


b) A dynamic reference to a local reference target should be encoded to contain the id of a DynamicObjectRefType instance. The reference target is determined by the singleton result returned by the Query within the DynamicObjectRef instance.

The following example shows the reference attribute named primaryContact within Organization element. Its value is the value of the *id* attribute of a DynamicObjectRefType instance. The DynamicObjectRefType instance has a *Query* that gets the latest version of the object identified by the *lid* parameter of the Query. The query when invoked matches the latest version of the Person object representing Danyal.

EXAMPLE 2:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  primaryContact="urn:acme:dynamicRef:LatestVersionOfDanyal" ...>
  ...
  ...
</rim:RegistryObject>

<rim:ObjectRef xsi:type="rim:ObjectRefType"
  id="urn:acme:dynamicRef:LatestVersionOfDanyal">
  <rim:Query queryDefinition="urn:acme:QueryDefinition:FindLatestVersion">
    <rim:Slot name="lid">
      <rim:SlotValue xsi:type="rim:StringValueType">
        <rim:Value>urn:acme:person:Danyal</rim:Value>
      </rim:SlotValue>
    </rim:Slot>
  </rim:Query>
</rim:ObjectRef>

<rim:RegistryObject xsi:type="rim:PersonType"
  lid="urn:acme:person:Danyal" id="urn:acme:person:Danyal:1.8"...>
  <!-- latest version of object with lid "urn:acme:person:Danyal" -->
  ...
</rim:RegistryObject>
```

c) A static or dynamic reference to a local reference target may be encoded to contain a Canonical URL for the local object as defined by the REST binding in the registry services (see clause 5).

d) A static or dynamic reference to a remote reference target shall be encoded to contain a Canonical URL for the local object as defined by the REST binding in the registry services (see clause 5).

The following example shows the reference attribute named primaryContact within Organization element. Its value is the HTTP GET URL for a remote PersonType instance. Note that the URL is not encoded to handle special characters for sake of clarity.

EXAMPLE 3:

```
<!-- Following object is in local server -->
<rim:RegistryObject xsi:type="rim:OrganizationType"

primaryContact="http://www.remoteRegistry.com/query?id=urn:remoteServer:person:Danyal" ...
>
  ...
  ...
</rim:RegistryObject>

<!-- Following object is in a remote server -->
<rim:RegistryObject xsi:type="rim:PersonType"
  id="urn:remoteServer:person:Danyal" ...>
  ...
</rim:RegistryObject>
```


4.2.11 ObjectRefType

4.2.11.1 Overview

Base Type: ExtensibleObjectType

This type represents an object reference as does the objectReferenceType. However, the two are used in different situations. The objectReferenceType is used as the type for all reference attributes in ebRIM. The ObjectRefType is used as type for elements rather than attributes. This type is used when there is a need to have multiple object references within a schema type. An example of this is the ObjectRefList element which is used in several places in the schema where a list of references to RegistryObjectType instances is needed.

4.2.11.2 XML schema type definition

```
<complexType name="ObjectRefType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="id" type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="ObjectRefListType">
  <sequence>
    <element name="ObjectRef"
      type="tns:ObjectRefType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<element name="ObjectRefList" type="tns:ObjectRefListType"/>
```

4.2.11.3 Description

Table 9 shows the description of ObjectRefType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
id	objectReferenceType	1		Client	Yes

Table 9 ObjectRefType

Attribute *id* - Every ObjectRef instance shall have an *id* attribute. The *id* attribute shall contain the value of the *id* attribute of the RegistryObject being referenced.

4.2.12 DynamicObjectRefType

4.2.12.1 Overview

Base Type: ObjectRefType

This type represents a dynamic object reference. It extends the ObjectRefType and add a Query sub-element. This query is used to determine the reference target at the time the reference is resolved.

4.2.12.2 XML schema type definition

```
<complexType name="DynamicObjectRefType">
  <complexContent>
    <extension base="tns:ObjectRefType">
      <sequence>
```



```
        <element name="Query" type="tns:QueryType"
          minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

4.2.12.3 **Description**

Table 10 shows the description of DynamicObjectRefType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Query	QueryType	1		Client	Yes

Table 10 DynamicObjectRefType

Element Query – Specifies the query that shall be invoked in order to determine the reference target.

1) This query shall match zero or one RegistryObjectType instances.

2) When the query matches zero RegistryObjectType instances, the dynamic object reference is considered to be unresolved.

3) A server shall return a ConfigurationException fault message if the query matches more than 1 RegistryObjectType instances.

4.2.13 ExtrinsicObjectType

4.2.13.1 **Overview**

Base Type: RegistryObjectType

This type is a common base type for new extended types defined by profiles of ebRIM or by clients. The ExtrinsicObjectType also allows arbitrary content to be associated with it. Such arbitrary content is referred to as a Repository Item.

4.2.13.2 **XML schema type definition**

```
<complexType name="ExtrinsicObjectType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ContentVersionInfo" type="tns:VersionInfoType"
          minOccurs="0" maxOccurs="1"/>
        <choice minOccurs="0" maxOccurs="1">
          <element name="RepositoryItemRef" type="tns:SimpleLinkType"/>
          <element name="RepositoryItem"
            xmime:expectedContentTypes="*/*" type="base64Binary">
        </choice>
      </sequence>
      <attribute name="mimeType" type="tns:LongText" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="text/xml"
  objectType="urn:freebxml:registry:sample:profile:cpp:objectType:cppa:CPP"
```



```
  lid="urn:freebxml:registry:sample:profile:cpp:instance:cpp1"
  id="urn:freebxml:registry:sample:profile:cpp:instance:cpp1" >
    <ContentVersionInfo versionName="311" userVersionName="1.1"/>
    <RepositoryItem>...binary encoding of repository item</RepositoryItem>
</rim:RegistryObject>
```

4.2.13.3 **Description**

Table 11 shows the description of ExtrinsicObjectType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
ContentVersionInfo	VersionInfoType	0..1		Server	No
mimeType	LongText	0..1	application/octet-stream	Client	No
RepositoryItem	xs:base64Binary	0..1		Client	Yes
RepositoryItemRef	SimpleLinkType	0..1		Client	No

Table 11 ExtrinsicObjectType

a) Element ContentVersionInfo - Provides information about the specific version of a RepositoryItem that is associated with an ExtrinsicObjectType instance. The ContentVersionInfo element is set by the server.

   1) A server shall not set a ContentVersionInfo element for an ExtrinsicObjectType instance that does not have a RepositoryItem.

   2) A server shall set a ContentVersionInfo element for an ExtrinsicObjectType instance that has a RepositoryItem. The ContentVersionInfo element shall contain a versionName attribute whose value shall be unique for all versions of that RepositoryItem.

b) Attribute mimeType - An ExtrinsicObjectType instance may have a mimeType attribute defined. The mimeType provides information on the type of repository item cataloged by the ExtrinsicObject instance. The value of this attribute should be a registered MIME media type at http://www.iana.org/assignments/media-types.

c) Element repositoryItem – Provides a base64 binary encoded representation of the repository item associated with the ExtrinsicObjectType instance (if any).

d) Element repositoryItemRef – This element may be specified as an alternative to the repositoryItem element. Its type is SimpleLinkType. It uses xlink:simpleAttrs to specify a reference to a file on the client's local file system. This provides client libraries an alternative way to specify local files as repository item. The client library shall convert a repositoryItemRef element to a repositoryItem element prior to submitting it to the server.

4.2.14 CommentType

4.2.14.1 **Overview**

Base Type: ExtrinsicObjectType

This type represents a comment that may be associated with a RegistryObjectType instance. A comment associated with a RegistryObject models the familiar yellow POST-IT note metaphor used in attaching comments to paper documents.

4.2.14.2 XML schema type definition

```
<complexType name="CommentType">
  <complexContent>
    <extension base="tns:ExtrinsicObjectType">
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:CommentType"
  lid="urn:freebxml:registry:sample:comment1"
  id="urn:freebxml:registry:sample:comment1" >
  <rim:Description>
    <rim:LocalizedString
      xml:lang="en-US" value="This change request is rejected because it is too
        complex a change."/>
  </rim:Description>
</rim:RegistryObject>
```

4.2.14.3 Description

No new attributes or elements are added by this type. The following requirements are defined for this type:

a) An authorized client may attach one or more comments to any RegistryObjectType instance using an Association between the RegistryObjectType instance and the CommentType instance

   Since a CommentType is itself a RegistryObjectType, a client may attach one or more comments to any CommentType instance

b) The type of the Association shall reference the canonical HasComment ClassificationNode within the Canonical AssociationType ClassificationScheme

c) The sourceObject of the Association shall be the RegistryObjectType instance

d) The targetObject of the Association shall be the CommentType instance

4.2.15 RegistryPackageType

4.2.15.1 Overview

Base Type: RegistryObjectType

This type allows for grouping of related RegistryObjectType instances. It serves a similar role as a folder in the familiar file-folder metaphor available in most operating systems.

a) A RegistyObjectType instance may be a member of multiple RegistryPackageType instances.

b) A RegistryPackageType instance may have multiple RegistryObjectType instances as its members.

c) Membership of a RegistryObjectType instance in a RegistryPackageType instance is established via an AssociationType instance where the type attribute references the canonical “HasMember” AssociationType within the canonical AssociationTypeScheme ClassificationScheme.

d) As a convenience, the RegistryPackageType allows a RegistryObjectList to be specified by the client as a sub-element during submission of a RegistryPackage. The RegistryObjectList contains the set of RegistryObjectType instances that are members of the RegistryPackageType instance.

4.2.15.2 XML schema type definition

```
<complexType name="RegistryPackageType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="RegistryObjectList" type="tns:RegistryObjectListType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

4.2.15.3 Examples

The following example shows the use of a RegistryObjectList to specify the members of a RegistryPackageType instance during submission.

EXAMPLE 1

```
<rim:RegistryObject xsi:type="rim:RegistryPackageType"
   id="urn:acme:RegistryPackage:photos" ...>
  ...
  <rim:RegistryObjectList>
    <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="image/jpeg"
id="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
      <repositoryItem>
        ...binary encoding of photo repository item
      </repositoryItem>
    </rim:RegistryObject>
  </rim:RegistryObjectList>
</rim:RegistryObject>
```

The following example shows the equivalent syntax for representing the membership relationship between a RegistryPackage and its members. This representation uses “HasMember” AssociationType instances to establish the membership relationship.

EXAMPLE 2

```
<rim:RegistryObject xsi:type="rim:RegistryPackageType"
  id="urn:acme:RegistryPackage:photos" .../>

<rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="image/jpeg"
id="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
  <repositoryItem>
    ...binary encoding of photo repository item
  </repositoryItem>
</rim:RegistryObject>

<Association
  sourceObject="urn:acme:RegistryPackage:photos"
  targetObject="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"/>
```

4.2.15.4 Description

Table 12 shows the description of RegistryPackageType.

Node	Type	Cardinality	Default Value	Specified By	Mutable

RegistryObjectList	RegistryObjectList-Type	0..1		Client	Yes

Table 12 RegistryPackageType

Element RegistryObjectList – This element allows clients to specify members of the RegistryPackage instance using a simpler alternative to "HasMember" AssociationType instances.

A server shall replace the RegistryObjectList with AssociationType instances such that each RegistryObjectType instance is replaced with an AssociationType instance with type "urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember", with sourceObject specifying the id of the RegistryPackage instance and with targetObject specifying the id of the RegistryObjectType instance

4.2.16 ExternalIdentifierType

4.2.16.1 Overview

Base Type: RegistryObjectType

This type allows any number of additional identifiers to be specified for a RegistryObjectType instance. The identifier value is defined using the *value* attribute within the context of a ClassificationScheme referenced via the *identificationScheme* attribute.

4.2.16.2 XML schema type definition

```
<complexType name="ExternalIdentifierType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="registryObject"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="identificationScheme"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="value" type="tns:LongText" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

4.2.16.3 Example

The following example shows an Organization instance with its tax payer id specified using an ExternalIdentifierType instance.

EXAMPLE

```
<rim:RegistryObject xsi:type="rim:OrganizationType" ...>
  ...
  <rim:ExternalIdentifier ...
    identificationScheme="urn:acme:ClassificationScheme:TaxPayerId"
    value="1234567890"/>
  </rim:ExternalIdentifier>
  ...
</rim:RegistryObject>
```

4.2.16.4 Description

Table 13 shows the description of ExternalIdentifierType.

Node	Type	Cardinality	Default Value	Specified By	Mutable

identificationScheme	objectReferenceType	1		Client	Yes
registryObject	objectReferenceType	0..1		Client	No
value	LongText	1		Client	Yes

Table 13 ExternalIdentifierType

a) Attribute identificationScheme - Each ExternalIdentifier instance shall have an identificationScheme attribute that references a ClassificationScheme. This ClassificationScheme defines the namespace within which an identifier is defined using the value attribute for the RegistryObjectType instance referenced by the RegistryObject attribute.

b) Attribute registryObject - Each ExternalIdentifier instance may have a *registryObject* attribute specified. This attribute references the parent RegistryObjectType instance for which this is an ExternalIdentifier.

   1) This attribute shall be specified when a client submits an ExternalIdentifier separately from its parent RegistryObjectType instance

   2) This attribute may be unspecified when a client submits an ExternalIdentifier as a sub-element of its parent RegistryObjectType instance. In such cases the server shall set this attributes value to the value of the id attribute of the parent RegistryObjectType instance.

   3) Attribute value - Each ExternalIdentifier instance shall have a *value* attribute that provides the identifier value for this ExternalIdentifier (e.g., the tax payer id in example above).

4.2.17 ExternalLinkType

4.2.17.1 Overview

Base Type: RegistryObjectType

This type allows a link to external content to be added to a RegistryObjectType instance.

4.2.17.2 XML schema type definition

```
<complexType name="ExternalLinkType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ExternalRef"
          type="tns:SimpleLinkType" minOccurs="1" maxOccurs="1"/>
      </sequence>
      <attribute name="registryObject"
        type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

4.2.17.3 Example

The following example shows an Organization instance with an ExternalLink that links to its web site URL via its ExternalRef sub-element.

EXAMPLE

```
<rim:RegistryObject xsi:type="rim:OrganizationType" ...>
  ...
  <rim:ExternalLink ...
```



```
    objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:WSDL"
    mimeType="text/xml"/>
    <ExternalRef xlink:href="http://www.acme.com"/>
  </rim:ExternalLink>
  ...
</rim:RegistryObject>
```

4.2.17.4 **Description**

Table 14 shows the description of ExternalLinkType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
ExternalRef	SimpleLinkType	1		Client	Yes
registryObject	objectReferenceType	0..1		Client or Server	No

Table 14 ExternalLinkType

a) Element ExternalRef - Each ExternalLink instance shall have an ExternalRef sub-element defined. This element provides a URI to the external resource pointed to by this ExternalLink instance.

b) Attribute registryObject – references the parent RegistryObjectType instance within which the ExtrnalLinkType instance is composed. The value shall be provided by client when an ExtrenalLink is submitted separate from its parent object. The value shall be set by the server if the ExternalLink is submitted as part of the submission of its parent object.

4.3 Association information model

4.3.1 Overview

A RegistryObjectType instance may be associated or related with zero or more RegistryObjectType instances. The information model defines the AssociationType type, an instance of which may be used to associate any two RegistryObjectType instances. It also defines an Association element for that type.

In Figure 3 below, an AsssociationType instance with type “urn:oasis:names:tc:ebxml-regrep:AssociationType:Supersedes” is used to indicate that the NAICS2001 ClassificationScheme supercedes the NAICS1997 ClassificationScheme.

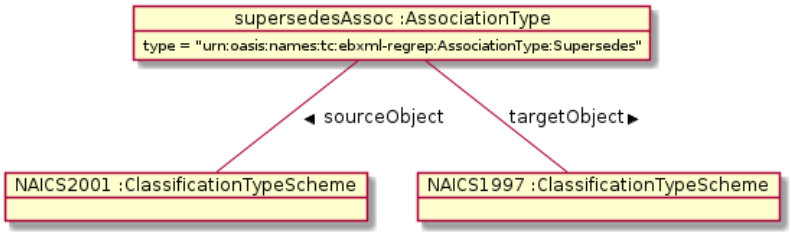

Figure 3 association example

4.3.2 Source and target objects

An AssociationType instance represents an association between a source RegistryObjectType instance and a target RegistryObjectType instance. These are referred to as *sourceObject* and *targetObject* for the AssociationType instance. It is important which object is the sourceObject and which is the targetObject as it determines the directional semantics of an Association.

4.3.3 Type of an association

An AssociationType instance shall have a type attribute that identifies the type of that association. The value of this attribute is typically the id of a ClassificationNode under the canonical AssociationType ClassificationScheme.

4.3.4 AssociationType

Base Type: RegistryObjectType

4.3.4.1 XML schema type definition

```
<complexType name="AssociationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="type"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="sourceObject"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="targetObject"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: an Organization instance that has an "OffersService" association with a Service that it offers.

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:acme:Organization:acme-inc" ... />
<rim:RegistryObject xsi:type="rim:ServiceType"
  id="urn:acme:Service:stock-quote" ... />
<rim:RegistryObject xsi:type="rim:AssociationType"
  id="urn:acme:Association:acme-example-relationship"
  sourceObject="urn:acme:Organization:acme-inc"
  targetObject="urn:acme:Service:stock-quote"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:OffersService" .../>
```

4.3.4.2 Description

Table 15 shows the description of AssociationType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
sourceObject	objectReferenceType	1		Client	Yes
targetObject	objectReferenceType	1		Client	Yes
type	objectReferenceType	1		Client	Yes

Table 15 AssociationType

a) Attribute sourceObject - Each Association shall have a *sourceObject* attribute that references the RegistryObjectType instance that is the source of that Association.

b) Attribute targetObject - Each Association shall have a *targetObject* attribute that references the RegistryObjectType instance that is the target of that Association.

c) Attribute type - Each Association shall have a t*ype* attribute that identifies the type of that association.

1) The value of the type attribute shall be a reference to a ClassificationNode within the canonical AssociationType ClassificationScheme.

2) A server shall support the canonical association types as defined by the canonical AssociationType ClassificationScheme. Deployments and profiles may extend the canonical AssociationType ClassificationScheme by adding additional ClassificationNodes to it.

4.3.5 Access control

A client may create an AssociationType instance between *any* two RegistryObjectType instances assuming the access control policies associated with the source and target object permit the client to create a reference to them. The default access control policy permits any client to create a reference to an object.

4.4 Classification information model

4.4.1 Overview

The ebRIM information model supports classification of RegistryObjectType instances using values defined by a taxonomy or controlled vocabulary. A taxonomy is represented in ebRIM by the ClassificationSchemeType type. Values in a taxonomy are represented by the ClassificationNode type. A classification instance is represented in ebRIM by the ClassificationType type.

This specification specifies a set of canonical ClassificationSchemes. Deployments and profiles may extends these canonical ClassificationSchemes by adding additional ClassificationNodes to them. They may also define new ClassificationSchemes. A RegistryObjectType instance may be classified using *any* ClassificationNode in *any* ClassificationScheme supported by the server. A RegistryObjectType instance may have any number of classifications defined for it.

A general ClassificationScheme can be viewed as a tree structure where the ClassificationScheme is the root and ClassificationNodes are either intermediate or leaf nodes in the tree.

Figure 4 shows RegistryObjectType instances representing Organizations as grey boxes. Each Organization represents an automobile manufacturer. Organization is classified by the ClassificationNode named “Automotive” under the ClassificationScheme instance with name “IndustryScheme”. Furthermore, the US Automobile manufacturers are classified by the “US” ClassificationNode under the ClassificationScheme with name “GeographyScheme”. Similarly, a European automobile manufacturer is classified by the “Europe” ClassificationNode under the ClassificationScheme with name “GeographyScheme”.

Figure 4 Classification example shows how a RegistryObject may be classified by multiple ClassificationNodeType instances under multiple ClassificationScheme instances (e.g., IndustryScheme, GeographyScheme).

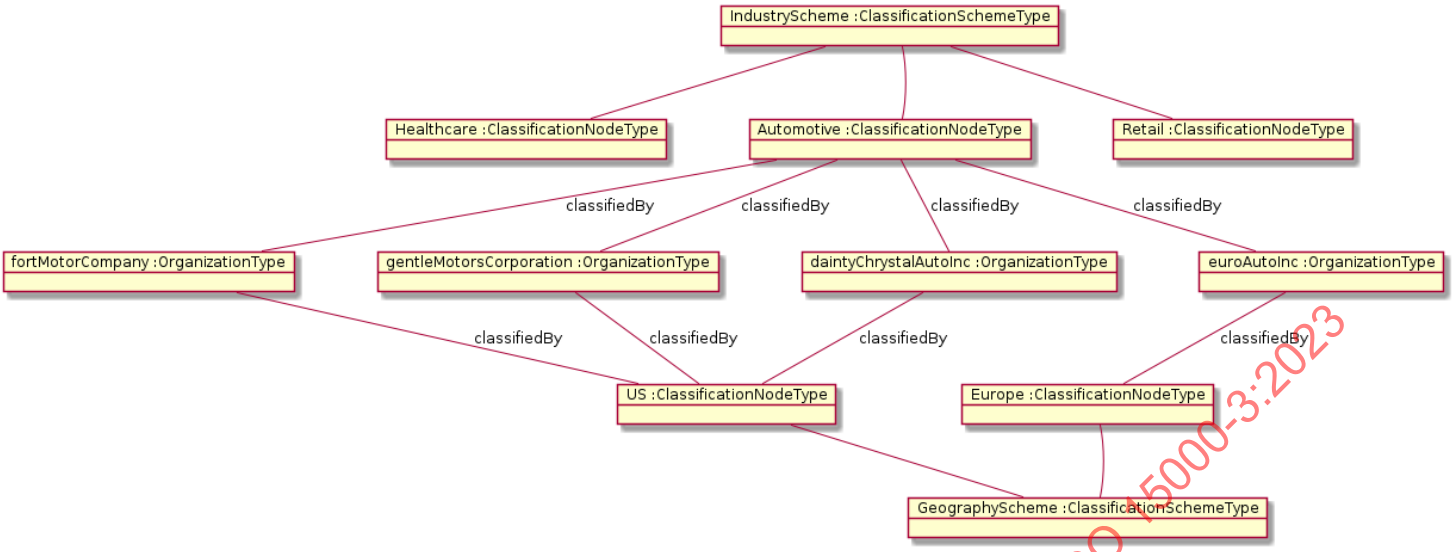

Figure 4 Classification example

Figure 5 shows the classification information model.

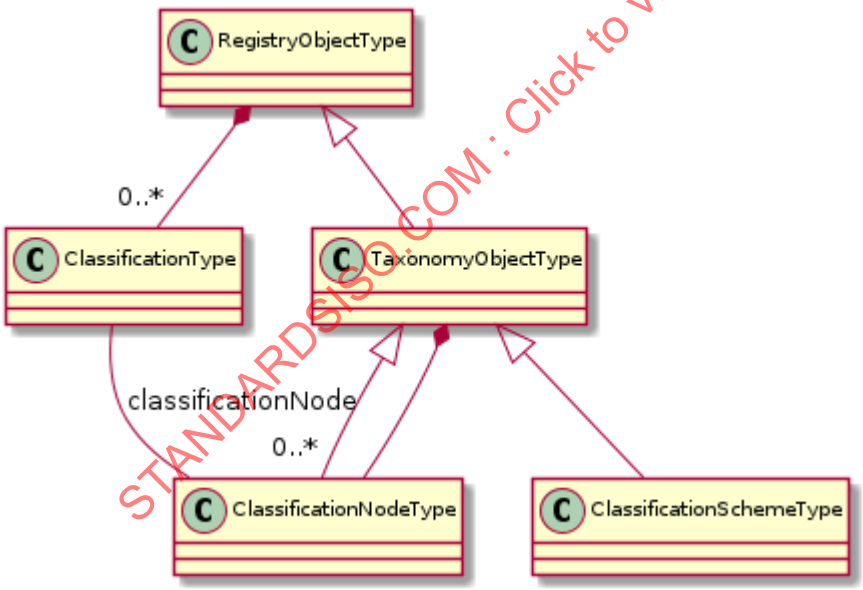

Figure 5 Classification information model

4.4.2 TaxonomyElementType

4.4.2.1 Overview

Base Type: RegistryObjectType

This abstract type is the common base type for ClassificationSchemeType and ClassificationNodeType.

4.4.2.2 XML schema type definition

```
<complexType name="TaxonomyElementType" abstract="true">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ClassificationNode" type="tns:ClassificationNodeType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

4.4.2.3 Description

Table 16 TaxonomyElementType shows the description of TaxonomyElementType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
ClassificationNode	ClassificationNodeType	0..*		Client	Yes

Table 16 TaxonomyElementType

Element ClassificationNode – This element represents a ClassificationNode child of a parent TaxonomyElementType instance. A TaxonomyElementType instance may have any number of ClassificationNode child elements.

4.4.3 ClassificationSchemeType

4.4.3.1 Overview

Base Type: TaxonomyElementType

A ClassificationScheme instance represents a taxonomy.

The taxonomy hierarchy may be defined internally to the server using instances of ClassificationNodeType type, or it may be defined externally to the server, in which case the structure and values of the taxonomy elements are not known to the Registry.

In the first case the classification scheme is said to be *internal* and in the second case the classification scheme is said to be *external*.

4.4.3.2 XML schema type definition

```
<complexType name="ClassificationSchemeType">
  <complexContent>
    <extension base="tns:TaxonomyElementType">
      <attribute name="isInternal" type="boolean" use="required"/>
      <attribute name="nodeType"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: a ClassificationScheme representing gender values.

```
<rim:RegistryObject xsi:type="rim:ClassificationSchemeType"
  id="urn:acme:GenderScheme" isInternal="true"
  nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode" ...>
```

```
  <Name>
    <LocalizedString value="GenderScheme"/>
  </Name>
  <rim:ClassificationNode id="urn:acme:Gender:Male" code="Male" .../>
  <rim:ClassificationNode id="urn:acme:Gender:Female" code="Female" .../>
  <rim:ClassificationNode id="urn:acme:Gender:Other" code="Other" .../>
</rim:RegistryObject>
```

4.4.3.3 **Description**

Table 17 ClassificationSchemeType shows the description of ClassificationSchemeType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
isInternal	xs:boolean	1		Client	No
nodeType	objectReferenceType	1		Client	No

Table 17 ClassificationSchemeType

a) Attribute isInternal - When submitting a ClassificationSchemeType instance the client shall declare whether the ClassificationSchemeType instance represents an internal or an external taxonomy. This allows the server to validate the subsequent submissions of ClassificationNodeType and ClassificationType instances in order to maintain the type of ClassificationScheme consistent throughout its lifecycle.

b) Attribute nodeType - When submitting a ClassificationScheme instance the client shall declare the structure of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The value of the nodeType attribute shall be a reference to a ClassificationNodeType instance within the canonical NodeType ClassificationScheme. A server shall support the node types as defined by the canonical NodeType ClassificationScheme. The canonical NodeType ClassificationScheme may easily be extended by adding additional ClassificationNodes to it.

Table 18 NodeType classification scheme lists the canonical ClassificationNode defined as values for the NodeType ClassificationScheme:

Name	Description
UniqueCode	Indicates that the code for each ClassificationNode in the ClassificationScheme is unique within the scope of the ClassificationScheme
EmbeddedPath	Indicates that the code assigned to each node of the taxonomy also encodes its path.
NonUniqueCode	Indicates that the code for each ClassificationNode in the ClassificationScheme is not unique within the scope of the ClassificationScheme. For example, in a geography taxonomy Moscow could be under both Russia and the USA, where there are five cities of that name in different states.

Table 18 NodeType classification scheme

4.4.4 ClassificationNodeType

4.4.4.1 **Overview**

Base Type: TaxonomyElementType

ClassificationNodeType instances are used to define values for a taxonomy represented by ClassificationSchemeType instance.

4.4.4.2 XML schema type definition

```
<complexType name="ClassificationNodeType">
  <complexContent>
    <extension base="tns:TaxonomyElementType">
      <attribute name="parent" type="tns:objectReferenceType" use="optional"/>
      <attribute name="path" type="string" use="optional"/>
      <attribute name="code" type="tns:LongText" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

4.4.4.3 Description

Table 19 ClassificationNodeType shows the description of ClassificationNodeType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
code	LongText	1		Client	No
parent	objectReferenceType	0..1		Client	No
path	xs:string	0..1		Registry	No

Table 19 ClassificationNodeType

a) Attribute code - A ClassificationNodeType instance shall have a *code* attribute. The code attribute contains a code that represents a value within a ClassificationScheme.

   The code attribute of a ClassificationNodeType instance shall be unique with respect to all sibling ClassificationNodes that are immediate children of the same parent TaxonomyElementType instance.

b) Attribute parent - A ClassificationNodeType instance may have a *parent* attribute. The parent attribute references the parent TaxonomyElementType instance. This is either another ClassificationNodeType instance or the ClassificationSchemeType instance.

c) Attribute path - A ClassificationNodeType instance may have a *path* attribute. The path attribute represents a hierarchical path from the root ClassificationSchemeType to the ClassificationNodeType instance. The syntax of the path attribute value is defined in 4.4.4.4.

   1) A server shall set the path attribute for any ClassificationNodeType instance when it is submitted by a client.

   2) The path attribute shall be ignored by the server if it is specified by the client during the submission of the ClassificationNodeType instance.

   3) The path attribute of a ClassificationNode shall be unique within a server.

4.4.4.4 Canonical Path Syntax

The path attribute of the ClassificationNodeType instance contains an absolute path in a canonical representation that uniquely identifies the path leading from the root ClassificationSchemeType instance to that ClassificationNodeType instance.

The canonical path representation is defined by the following BNF grammar:

```
canonicalPath ::= '/' rootTaxonomyElementId nodePath
nodePath   ::=   '/' nodeCode
      |     '/' nodeCode ( nodePath )?
```

In the above grammar, rootTaxonomyElementId is the id attribute of the root ClassificationSchemeType or ClassificationNodeType instance, and nodeCode is defined by NCName production as defined by http://www.w3.org/TR/REC-xml-names/#NT-NCName.

The following canonical path represents the *path* attribute value for the ClassificationNode with code "Male" in the sample Gender ClassificationScheme presented earlier.

EXAMPLE:

```
/urn:acme:GenderScheme/Male
```

4.4.5 ClassificationType

4.4.5.1 Overview

Base Type: RegistryObjectType

A ClassificationType instance classifies a RegistryObjectType instance by using a value defined within a particular ClassificationScheme. An internal Classification specifies the value by referencing the ClassificationNodeType instance within a ClassificationSchemeType instance. An external Classification specifies the value using a string value that is defined in some external specification represented by an external ClassificationSchemeType instance.

4.4.5.2 XML schema type definition

```
<complexType name="ClassificationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="classificationScheme"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="classifiedObject"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="classificationNode"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="nodeRepresentation"
        type="tns:LongText" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: a Person instance is classified using the sample Gender ClassificationScheme used in earlier examples.

```
<rim:RegistryObject xsi:type="rim:PersonType"
  id="urn:acme:person:Danyal" ...>
  ...
  <Classification classifiedObject="urn:acme:person:Danyal"
    classificationNode="urn:acme:Gender:Male"
  ...
</rim:RegistryObject>
```

4.4.5.3 Description

Table 20 ClassificationType shows the description of ClassificationType.

Node	Type	Cardinality	Default Value	Specified By	Mutable

classificationNode	objectReferenceType	0..1		Client	No
classifiedObject	objectReferenceType	0..1		Client	No
classificationScheme	objectReferenceType	0..1		Client	No
nodeRepresentation	LongText	0..1		Client	No

Table 20 ClassificationType

a) Attribute classificationNode - If the ClassificationType instance represents an internal classification, then the *classificationNode* attribute is required.

   The *classificationNode* value shall reference a ClassificationNodeType instance.

b) Attribute classifiedObject - For both internal and external classifications, the *classifiedObject* attribute is required and it references the RegistryObjectType instance that is classified by this Classification.

c) Attribute classificationScheme - If the ClassificationType instance represents an external classification, then the *classificationScheme* attribute is required.

   The classificationScheme value shall reference a ClassificationScheme instance.

d) Attribute nodeRepresentation - If the ClassificationType instance represents an external classification, then the *nodeRepresentation* attribute is required. It is a representation of a taxonomy value from a classification scheme.

e) A canonical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:Classification:context” may be optionally specified to provide additional context for a ClassificationType instance

4.5 Provenance information model

4.5.1 Overview

The term provenance in the English language implies the origin and history of ownership and custodianship of things of value. When applied to the ebXML RegRep, provenance implies information about the origin, history of ownership, custodianship, and other relationships between entities such as people, organizations and information represented by RegistryObjectType instances.

The ebRIM information model supports types and relationships that may be used to represent the

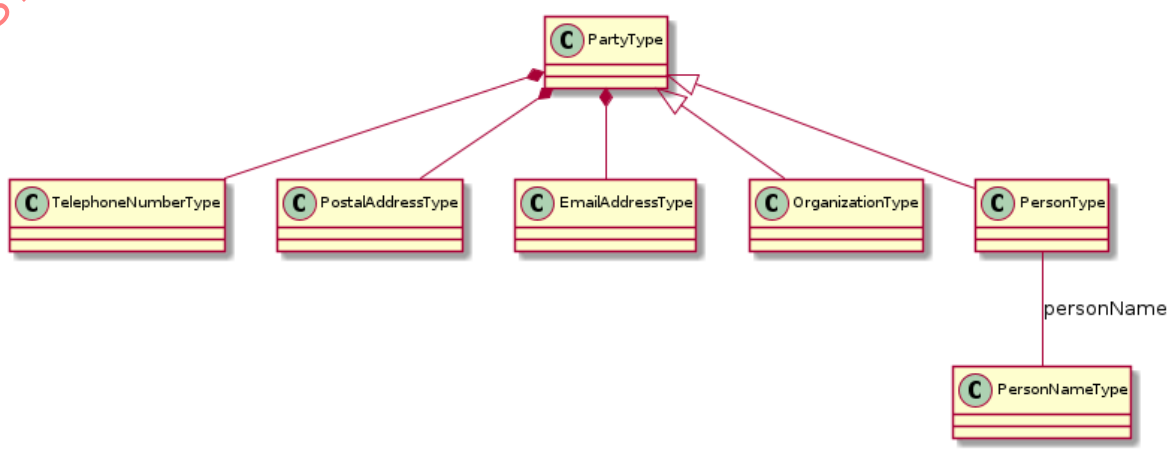

Figure 6 provenance information model

provenance of RegistryObjectType instances. Figure 6 presents the significant types defined by the provenance information model.

4.5.2 PostalAddressType

4.5.2.1 Overview

Base Type: ExtensibleObjectType

This type represents a postal or mailing address.

4.5.2.2 XML schema type definition

```
<complexType name="PostalAddressType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="city" type="tns:ShortText" use="optional"/>
      <attribute name="country" type="tns:ShortText" use="optional"/>
      <attribute name="postalCode" type="tns:ShortText" use="optional"/>
      <attribute name="stateOrProvince" type="tns:ShortText" use="optional"/>
      <attribute name="street" type="tns:ShortText" use="optional"/>
      <attribute name="streetNumber" type="tns:String32" use="optional"/>
      <attribute name="type" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:PostalAddress streetNumber="10" street="Street 1" city="Islamabad"
    stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
  ...
</rim:RegistryObject>
```

4.5.2.3 Description

Table 21 PostalAddressType shows the description of PostalAddressType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
city	ShortText	No		Client	Yes
country	ShortText	No		Client	Yes
postalCode	ShortText	No		Client	Yes
stateOrProvince	ShortText	No		Client	Yes
street	ShortText	No		Client	Yes
streetNumber	String32	No		Client	Yes

Table 21 PostalAddressType

a) Attribute city - A PostalAddressType instance may have a *city* attribute identifying the city for that address.

b) Attribute country - A PostalAddressType instance may have a *country* attribute identifying the country for that address.

c) Attribute postalCode - A PostalAddressType instance may have a *postalCode* attribute identifying the postal code (e.g., zip code) for that address.

d) Attribute stateOrProvince - A PostalAddressType instance may have a *stateOrProvince* attribute identifying the state, province or region for that address.

e) Attribute street - A PostalAddressType instance may have a *street* attribute identifying the street name for that address.

f) Attribute streetNumber - A PostalAddressType instance may have a *streetNumber* attribute identifying the street number (e.g., 65) for the street address.

4.5.3 TelephoneNumberType

4.5.3.1 Overview

Base Type: ExtensibleObjectType

This type defines attributes of a telephone number.

4.5.3.2 XML schema type definition

```
<complexType name="TelephoneNumberType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="areaCode" type="tns:String8" use="optional"/>
      <attribute name="countryCode" type="tns:String8" use="optional"/>
      <attribute name="extension" type="tns:String8" use="optional"/>
      <attribute name="number" type="tns:String16" use="optional"/>
      <attribute name="type" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:TelephoneNumber countryCode="92" areaCode="51" number="123-4567"
    type="urn:oasis:names:tc:ebxml-regrep:PhoneType:MobilePhone"/>
  ...
</rim:RegistryObject>
```

4.5.3.3 Description

Table 22 TelephoneNumberType shows the description of TelephoneNumberType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
areaCode	String8	0..1		Client	
countryCode	String8	0..1		Client	
extension	String8	0..1		Client	
number	String16	0..1		Client	
type	objectRef	0..1		Client	

Table 22 TelephoneNumberType

a) Attribute areaCode - A TelephoneNumberType instance may have an *areaCode* attribute that provides the area code for that telephone number.

b) Attribute countryCode - A TelephoneNumberType instance may have a *countryCode* attribute that provides the country code for that telephone number.

c) Attribute extension - A TelephoneNumberType instance may have an *extension* attribute that provides the extension number, if any, for that telephone number.

d) Attribute number - A TelephoneNumberType instance may have a *number* attribute that provides the local number (without area code, country code and extension) for that telephone number.

e) Attribute type - A TelephoneNumberType instance may have a *type* attribute that provides the type for the TelephoneNumber. The value of the phoneType attribute shall be a reference to a ClassificationNode in the canonical PhoneType ClassificationScheme.

4.5.4 EmailAddressType

4.5.4.1 Overview

Base Type: ExtensibleObjectType

This type defines attributes of an email address.

4.5.4.2 XML schema type definition

```
<complexType name="EmailAddressType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="address" type="tns:ShortText" use="required"/>
      <attribute name="type" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:EmailAddress address="danyal@play.com"
    type="urn:oasis:names:tc:ebxml-regrep:EmailType:HomeEmail"/>
  ...
</rim:RegistryObject>
```

4.5.4.3 Description

Table 23 EmailAddressTypeshows the description of EmailAddressType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
address	ShortText	1		Client	Yes
type	objectReferenceType	0..1		Client	Yes

Table 23 EmailAddressType

a) Attribute address - An EmailAddressType instance shall have an *address* attribute that provides the actual email address.

b) Attribute type - An EmailAddressType instance may have a *type* attribute that provides the type for that email address. The value of the type attribute shall be a reference to a ClassificationNode in the canonical EmailType ClassificationScheme.

4.5.5 PartyType

4.5.5.1 Overview

Base Type: RegistryObjectType

This abstract type represents a party that has contact information such as PostalAddress, EmailAddress, TelephoneNumber etc. It is used as a common base type for PersonType and OrganizationType.

4.5.5.2 XML schema type definition

```
<complexType name="PartyType" abstract="true">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="PostalAddress" type="tns:PostalAddressType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="TelephoneNumber" type="tns:TelephoneNumberType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="EmailAddress" type="tns:EmailAddressType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

4.5.5.3 Description

Table 24 shows the description of PartyType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
EmailAddress	EmailAddressType	0..*		Client	Yes
PostalAddress	PostalAddressType	0..*		Client	Yes
TelephoneNumber	TelephoneNumberType	0..*		Client	Yes

Table 24 PartyType

a) Element EmailAddress - A PartyType instance may have any number of EmailAddress sub-elements. Each EmailAddress provides an email address for that PartyType instance. A PartyType instance should have at least one EmailAddress.
b) Element PostalAddress - A PartyType instance may have any number of PostalAddress sub-elements. Each PostalAddress element provides a postal address for that PartyType instance. A PartyType instance should have at least one PostalAddress.
c) Element TelephoneNumber - A PartyType instance may have any number of T*elephoneNumber* sub-elements. Each TelephoneNumber element provides a TelephoneNumber for that PartyType instance. A PartyType instance should have at least one TelephoneNumber.

4.5.6 PersonType

4.5.6.1 Overview

Base Type: PartyType

This type represents a person.

4.5.6.2 XML schema type definition

```
<complexType name="PersonType">
  <complexContent>
    <extension base="tns:PartyType">
      <sequence>
        <element name="PersonName" type="tns:PersonNameType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  <rim:PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
  <rim:PostalAddress streetNumber="10" street="Street 1" city="Islamabad"
    stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
  <rim:TelephoneNumber countryCode="92" areaCode="51" number="123-4567"
    type="urn:oasis:names:tc:ebxml-regrep:PhoneType:MobilePhone"/>
  <rim:EmailAddress address="danyal@play.com"
    type="urn:oasis:names:tc:ebxml-regrep:EmailType:HomeEmail"/>
</rim:RegistryObject>
```

4.5.6.3 Description

Table 25 PersonType shows the description of PersonType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
PersonName	PersonNameType	0..1		Client	No

Table 25 PersonType

Element PersonName – A PersonType instance should have a P*ersonName* sub-element that provides the name for that person.

4.5.7 PersonNameType

4.5.7.1 Overview

Base Type: ExtensibleObjectType

This represents the name for a PersonType instance.

4.5.7.2 XML schema type definition

```
<complexType name="PersonNameType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="firstName" type="tns:ShortText" use="optional"/>
      <attribute name="middleName" type="tns:ShortText" use="optional"/>
```



```
      <attribute name="lastName" type="tns:ShortText" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
  ...
</rim:RegistryObject>
```

4.5.7.3 Description

Table 26 shows the description of PersonNameType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
firstName	ShortText	0..1		Client	Yes
lastName	ShortText	0..1		Client	Yes
middleName	ShortText	0..1		Client	Yes

Table 26 PersonNameType

a) Attribute firstName - A PersonName instance should have a *firstName* attribute that is the given name of the person.
b) Attribute lastName - A PersonName instance should have a *lastName* attribute that is the family name of the person.
c) Attribute middleName - A PersonName instance should have a *middleName* attribute that is the middle name of the person.

4.5.8 OrganizationType

4.5.8.1 Overview

Base Type: PartyType

This type represents an organization or entity.

4.5.8.2 XML schema type definition

```
<complexType name="OrganizationType">
  <complexContent>
    <extension base="tns:PartyType">
      <sequence>
        <element name="Organization" type="tns:OrganizationType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="primaryContact" type="tns:objectReferenceType"
        use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:acme:Organization:acme"
```

```
  primaryContact="urn:acme:person:Danyal" ...>
  <rim:PostalAddress streetNumber="1" street="Grand Trunk Rd."
   city="Hasan Abdal"
   stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
  <rim:TelephoneNumber countryCode="92" areaCode="52" number="123-4567"
    type="urn:oasis:names:tc:ebxml-regrep:PhoneType:OfficePhone"/>
  <rim:EmailAddress address="info@acme.com"
    type="urn:oasis:names:tc:ebxml-regrep:EmailType:OfficeEmail"/>
</rim:RegistryObject>
```

4.5.8.3 Description

Table 27 shows the description of ObjectRefType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Organization	OrganizationType	0..*		Client	
primaryContact	objectReferenceType	0..1		Client	

Table 27 OrganizationType

a) Element Organization – This element allows clients to specify sub-organizations of the Organization instance using a simpler alternative to specifying "HasMember" AssociationType instances between the parent and child Organizations.

   A server shall replace any nested Organization elements within an OrganizationType instance with AssociationType instances such that each nested Organization element is replaced with an AssociationType instance with type "urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember", with sourceObject specifying the id of the parent OrganizationType instance and with targetObject specifying the id of the nested Organization element

b) Attribute primaryContact - An OrganizationType instance should have a *primaryContact* attribute that references the Person instance for the person that is the primary contact for that organization.

4.5.9 Associating organization with persons

There are many situations where a person is related to an organization. Such relationship should be defined by AssociationType instances between an OrganizationType instance and a PersonType instance as follows:

a) The type attribute of the Association references the canonical ClassificationNode with id "urn:oasis:names:tc:ebxml-regrep:AssociationType:AffiliatedWith" or one of its descendants.

b) The sourceObject references the PersonType instance.

c) The targetObject references the OrganizationType instance.

4.5.10 Associating organization with organizations

There are many situations where an organization is related to another organization. Such relationship should be defined by AssociationType instances between an OrganizationType instance and another OrganizationType instance.

a) To represent parent-child relationship between organizations the type attribute of the Association should reference the canonical ClassificationNode with id "urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember" or one of its descendants.
b) The sourceObject should reference the parent OrganizationType instance.

c) The targetObject should reference the child OrganizationType instance.

4.5.11 Associating organizations with registry objects

An organization may be associated with zero or more RegistryObjectType instances. Each such association is modeled in ebRIM using an Association instance between an Organization instance and a RegistryObjectType instance.

Associations between Organizations and RegistryObjectType instances do not entitle organizations to any special privileges with respect to those instances. Such privileges are defined by the Access Control Policies defined for the RegistryObjectType instances as described in the Access Control Information Model clause.

4.6 Service information model

4.6.1 Overview

This clause describes the parts of the information model that support the description of services within an ebXML RegRep server. Although service information model aligns with the Web Services Description Language (WSDL) Version 2.0 model, it may be used to describe any type of service in addition to web services. The model is shown in Figure 7.

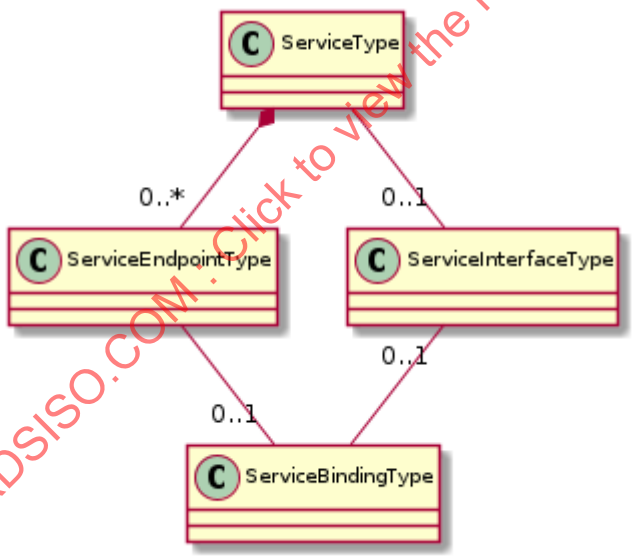

Figure 7 Service information model

4.6.2 ServiceType

4.6.2.1 Overview

Base Type: RegistryObjectType

This type represent a logical service. Physical service endpoints are represented by the ServiceEndpointType type. A ServiceType instance typically contains ServiceEndpoint sub-elements where each ServiceEndpoint sub-element represents an alternate endpoint for a service.

4.6.2.2 XML schema type definition

```
<complexType name="ServiceType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ServiceEndpoint" type="tns:ServiceEndpointType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="serviceInterface"
        type="tns:objectReferenceType" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:ServiceType"
  id="urn:acme:Service:StockQuoteService" ...>
  ...
  <rim:ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:free" .../>
  <rim:ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:premium" .../>
</rim:RegistryObject>
```

4.6.2.3 Description

Table 28 shows the description of ServiceType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
ServiceEndpoint	ServiceEndpointType	0..*		Client	Yes
serviceInterface	objectReferenceType	0..1		Client	Yes

Table 28 ServiceType

a) Element ServiceEndpoint – Represents a physical endpoint for the service that may be used by clients to access the service.
b) Attribute serviceInterface – References the abstract interface description for the service. It shall reference a ServiceInterfaceType instance if specified.

4.6.3 ServiceEndpointType

4.6.3.1 Overview

Base Type: RegistryObjectType

This type represents a physical endpoint for the service that may be used by clients to access a service.

4.6.3.2 XML schema type definition

```
<complexType name="ServiceEndpointType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="address" type="anyURI" use="optional" />
      <attribute name="serviceBinding"
        type="tns:objectReferenceType" use="optional" />
    </extension>
  </complexContent>
```



```
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:ServiceType"
  id="urn:acme:Service:StockQuoteService" ...>
  ...
  <rim:ServiceEndpoint id="urn:acme:ServiceEndpoint:StockQuoteService:free"
    address="http://acme.com/StockQuoteService/free"
    serviceBinding="urn:acme:ServiceBinding:soap:StockQuoteService">
</rim:RegistryObject>
```

4.6.3.3 Description

Table 29 shows the description of ServiceEndpointType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
address	xs:anyURI	0..1		Client	Yes
serviceBinding	objectReferenceType	0..1		Client	Yes

Table 29 ServiceEndpointType

a) Attribute address – Represents the endpoint address URI that a client of the service endpoint may use to access the service endpoint
b) Attribute serviceBinding – References the ServiceBindingType instance that represents protocol-specific binding information for the ServiceEndpointType instance. It shall reference a ServiceBindingType instance.

4.6.4 ServiceBindingType

4.6.4.1 Overview

Base Type: RegistryObjectType

This type represents protocol-specific binding information for a ServiceEndpointType instance. Example of a protocol-specific binding is a SOAP binding.

4.6.4.2 XML schema type definition

```
<complexType name="ServiceBindingType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="serviceInterface"
        type="tns:objectReferenceType" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:ServiceBindingType"
  id="urn:acme:ServiceBinding:soap:StockQuoteService"
  serviceInterface="urn:acme:ServiceInterface:StockQuoteService" .../>
  ...
</rim:RegistryObject>
```

4.6.4.3 Description

Table 30 shows the description of ServiceBindingType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
serviceInterface	objectReferenceType	0..1		Client	Yes

Table 30 ServiceBindingType

a) Attribute serviceInterface – References a ServiceInterfaceType instance which represents the abstract service interface for the service. It shall reference a ServiceInterfaceType instance if specified.

4.6.5 ServiceInterfaceType

4.6.5.1 Overview

Base Type: RegistryObjectType

This type represents an abstract service interface for a service.

4.6.5.2 XML schema type definition

```
<complexType name="ServiceInterfaceType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE

```
<rim:RegistryObject xsi:type="rim:ServiceInterfaceType"
  id="urn:acme:ServiceInterface:StockQuoteService" .../>
  ...
</rim:RegistryObject>
```

4.6.5.3 Description

No attributes or elements beyond those inherited from RegistryObjectType are defined for this type.

4.7 Query information model

4.7.1 Overview

This clause describes the information model for defining and invoking parameterized queries in ebXML RegRep. The following significant types are defined by the Query Information Model:

a) QueryDefinitionType - Represents the definition of a parameterized query
b) QueryType – Represents the invocation of a parameterized query

Several canonical QueryDefinitionType instances are defined by the ebRS specification. Profiles of ebXML RegRep may define additional QueryDefinitionType instances as canonical queries for that profile. Deployments may also define additional QueryDefinitionType instances. Finally, clients may submit additional QueryDefinitionType instances.

A QueryDefinitionType instance may be invoked using a QueryType instance. The ebRS Query protocol allows clients to invoke a QueryDefinitionType instance using a QueryType instance within the Query protocol.

Figure 8 presents the significant types defined by the Query information model.

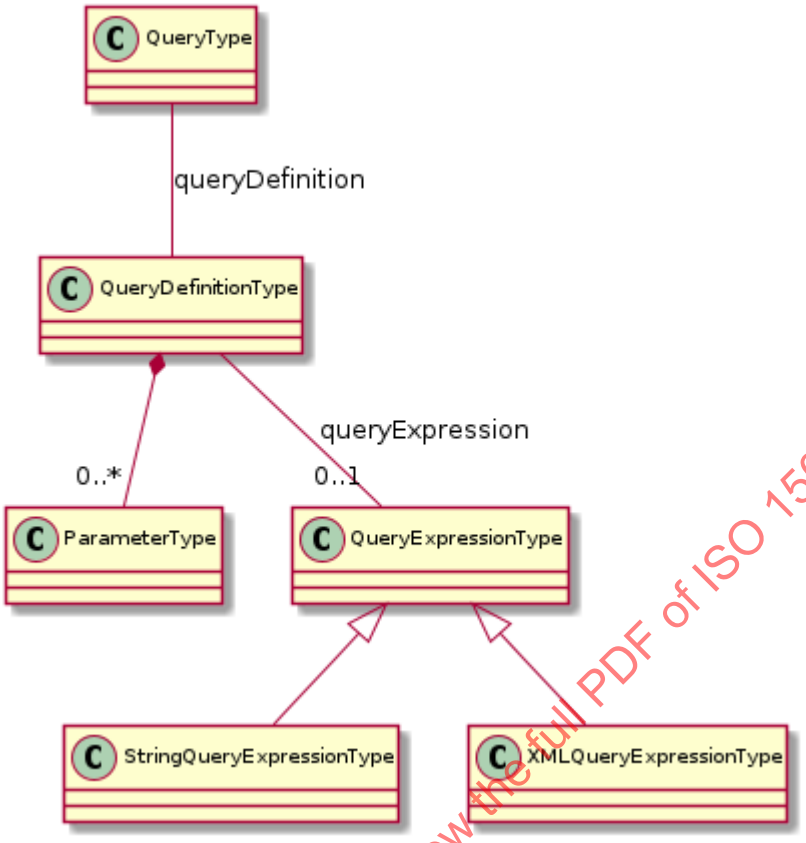

Figure 8 query information model

4.7.2 QueryDefinitionType

4.7.2.1 Overview

Base Type: RegistryObjectType

This type represents the definition of a parameterized query. The definition of a query includes the definition of its supported parameters and the definition of a parameterized query expression.

4.7.2.2 XML schema type definition

```
<complexType name="QueryDefinitionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="Parameter"
          type="tns:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="QueryExpression"
          type="tns:QueryExpressionType" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
```

```
  id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Parameter parameterName="id"
    minOccurs="1" maxOccurs="1" defaultValue="%">
  </Parameter>
  <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <Value>
    SELECT Object(ro) FROM ...
    </Value>
  </QueryExpression>
</rim:RegistryObject>
```

4.7.2.3 Description

Table 31 shows the description of QueryDefinitionType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Parameter	ParameterType	0..*		Client	Yes
QueryExpression	QueryExpressionType	0..1		Client	Yes

Table 31 QueryDefinitionType

a) Element Parameter – Represents the definition of a query parameter for the QueryDefinitionType instance. A QueryDefinitionType instance may have any number of Parameter sub-elements.
b) Element QueryExpression – Represents a query expression for the parameterized query. It may be omitted if the query is implemented as a Query plugin as defined by ebRS.

4.7.3 ParameterType

4.7.3.1 Overview

Base Type: ExtensibleObjectType

This type represents the definition of a parameter within a QueryDefinitionType.

4.7.3.2 XML schema type definition

```
<complexType name="ParameterType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="Name" type="tns:InternationalStringType"
          minOccurs="1" maxOccurs="1"/>
        <element name="Description" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
      <attribute name="parameterName" type="string" use="required"/>
      <attribute name="dataType" type="string" use="required" />
      <attribute name="defaultValue" type="string" use="optional"/>
      <attribute name="minOccurs" type="nonNegativeInteger" default="1"/>
      <attribute name="maxOccurs" type="nonNegativeInteger" default="1"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
  id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
```



```
  <rim:Parameter parameterName="id" dataType="string" minOccurs="1"
    maxOccurs="1" defaultValue="%" />
  ...
  <rim:QueryExpression .../>
</rim:RegistryObject>
```

4.7.3.3 **Description**

Table 32 shows the description of ParameterType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
dataType	xs:string	1		Client	Yes
defaultValue	xs:string	0..1		Client	Yes
Description	InternationalStringType	0..1		Client	Yes
minOccurs	xs:nonNegativeInteger	0..1	1	Client	Yes
maxOccurs	xs:nonNegativeInteger	0..1	1	Client	Yes
Name	InternationalStringType	1		Client	Yes
parameterName	xs:string	1		Client	Yes

Table 32 ParameterType

a) Attribute dataType – Specifies the data type for the parameter.

   1) The dataType shall be “string” for parameters whose values are represented by a string value.

   2) The dataType shall be “boolean” for parameters whose values are represented by a boolean value.

   3) The dataType shall be “taxonomyElement” for parameters whose value is the id of a TaxonomyElement.

b) Attribute defaultValue - Specifies the default value for the parameter. This value shall be used as parameter value when the query is invoked if the client does not specify a value for this parameter.

c) Element Description - Specifies a human-friendly description of the parameter that indicates what the parameter value represents and what kind of value is allowed. The description may be provided in multiple local languages and character sets.

d) Attribute minOccurs – Specifies the minimum number of values allowed for the parameter.

e) Attribute maxOccurs - Specifies the maximum number of values allowed for the parameter.

f) Element Name - Specifies a human-friendly name for the parameter. The name may be provided in multiple local languages and character sets.

g) Attribute parameterName – Specifies the canonical name of the parameter. The canonicalName identifies the parameter in a locale-insensitive manner

   1) should match a declared parameter name within the query expression for the QueryDefinitionType instance

2) The parameterName shall be unique across the universe of all sibling ParameterType instances within a QueryDefinitionType instance

4.7.4 QueryExpressionType

4.7.4.1 Overview

Base Type: ExtensibleObjectType

This type represents a query expression in a specified query language that may be used by the server to invoke a query.

The QueryExpressionType is the abstract root of a type hierarchy for the following more specialized sub-types:

a) StringQueryExpressionType – This type may be used to represent non-XML query syntaxes such as SQL-92 and EJBQL.
b) XMLQueryExpressionType - This type may be used to represent XML query syntaxes such as OGC Filter Query.

This specification does not specify a specific query expression syntax that a server must support.

4.7.4.2 XML schema type definition

```
<complexType name="QueryExpressionType" abstract="true">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="queryLanguage"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

4.7.4.3 Description

Table 33 shows the description of QueryExpressionType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
queryLanguage	objectReferenceType	1		Client	Yes

Table 33 QueryExpressionType

Attribute queryLanguage – Specifies the query language used by the QueryExpressionType instance. It shall be a reference to a ClassificationNode in the canonical Query Language ClassificationScheme whose id is “urn:oasis:names:tc:ebxml-regrep:classificationScheme:QueryLanguage”.

4.7.5 StringQueryExpressionType

4.7.5.1 Overview

Base Type: QueryExpressionType

This type is used to represent non-XML query syntaxes such as SQL-92 and EJBQL.

4.7.5.2 XML schema type definition

```
<complexType name="StringQueryExpressionType">
  <complexContent>
    <extension base="tns:QueryExpressionType">
      <sequence>
        <element name="Value" type="string" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
  id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Parameter ... />
  ...
  <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <Value>
    SELECT Object(ro) FROM RegistryObjectType WHERE ...
    </Value>
  </rim:QueryExpression>
</rim:RegistryObject>
```

4.7.5.3 Description

Table 34 shows the description of StringQueryExpressionType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Value	xs:string	1		Client	Yes

Table 34 StringQueryExpressionType

Element Value – Specifies the string value representing the actual query expression within the query language specified by the queryLanguage attribute inherited from base type QueryExpressionType.

4.7.6 XMLQueryExpressionType

4.7.6.1 Overview

Base Type: QueryExpressionType

This type is used to represent XML query syntaxes such as OGC Filter Query.

4.7.6.2 XML schema type definition

```
<complexType name="XMLQueryExpressionType">
  <complexContent>
    <extension base="tns:QueryExpressionType">
      <sequence>
        <any namespace="##other"
          processContents="lax" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
```

```
  <rim:Parameter ... />
  ...
  <rim:QueryExpression xsi:type="rim:XMLQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <ogc:Filter>
    ...
    </ogc:Filter>
  </rim:QueryExpression>
</rim:RegistryObject>
```

4.7.6.3 Description

An XMLQueryExpressionType instance may contain any XML element from a namespace other than the name space for rim.xsd. In the example above we use an ogc:Filter element to represent an OGC Filter query.

4.7.7 QueryType

Base Type: ExtensibleObjectType

This type represents the invocation of a parameterized query.

4.7.7.1 XML schema type definition

```
<complexType name="QueryType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="queryDefinition"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<rim:RegistryObject xsi:type="rim:QueryType"
  queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Slot name="id">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>urn:acme:person:Danyal</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
</rim:RegistryObject>
```

4.7.7.2 Description

Table 35 shows the description of QueryType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
queryDefinition	objectReferenceType	1		Client	Yes

Table 35 QueryType

a) Attribute queryDefinition – References the parameterized query to be invoked by the server.

   The value of this attribute shall be a reference to a QueryDefinitionType instance that is supported by the server.

b) Element Slot (Inherited) - Each Slot element specifies a parameter value for a parameter supported by the QueryDefinitionType instance.

1) The slot name shall match a parameterName attribute within a Parameter's definition within the QueryDefinitionType instance.

2) The slot value's type shall match the dataType attribute for the Parameter's definition within the QueryDefinitionType instance.

3) A server shall not treat the order of parameters as significant.

4.8 Event information model

4.8.1 Overview

This clause defines the information model types that supports the Event Notification feature for ebXML RegRep. These types include the following:

a) AuditableEventType – Represents a server event that is typically a consequence of a client request.
b) SubscriptionType – Represents a client's subscription to receive notification of AuditableEventType instances based upon a specified selection criteria.
c) QueryType – Represents a query invocation that is used to select events of interest within a SubscriptionType instance. This type has been specified previously in the Query Information Model.
d) NotificationType – Represents a notification sent by the server to a client regarding an event that matches the criteria specified by the client within a SubscriptionType instance.

Figure 9 shows how a Subscription may be defined that uses a QueryType instance as a selector query to select the AuditableEvents of interest to the subscriber. The Subscription may also have zero or more DeliveryInfoType elements that specify the subscriber's endpoint to deliver the selected events to. The endpoint may be a REST or SOAP service endpoint or it may be an email address endpoint in case notification is to be delivered via email.

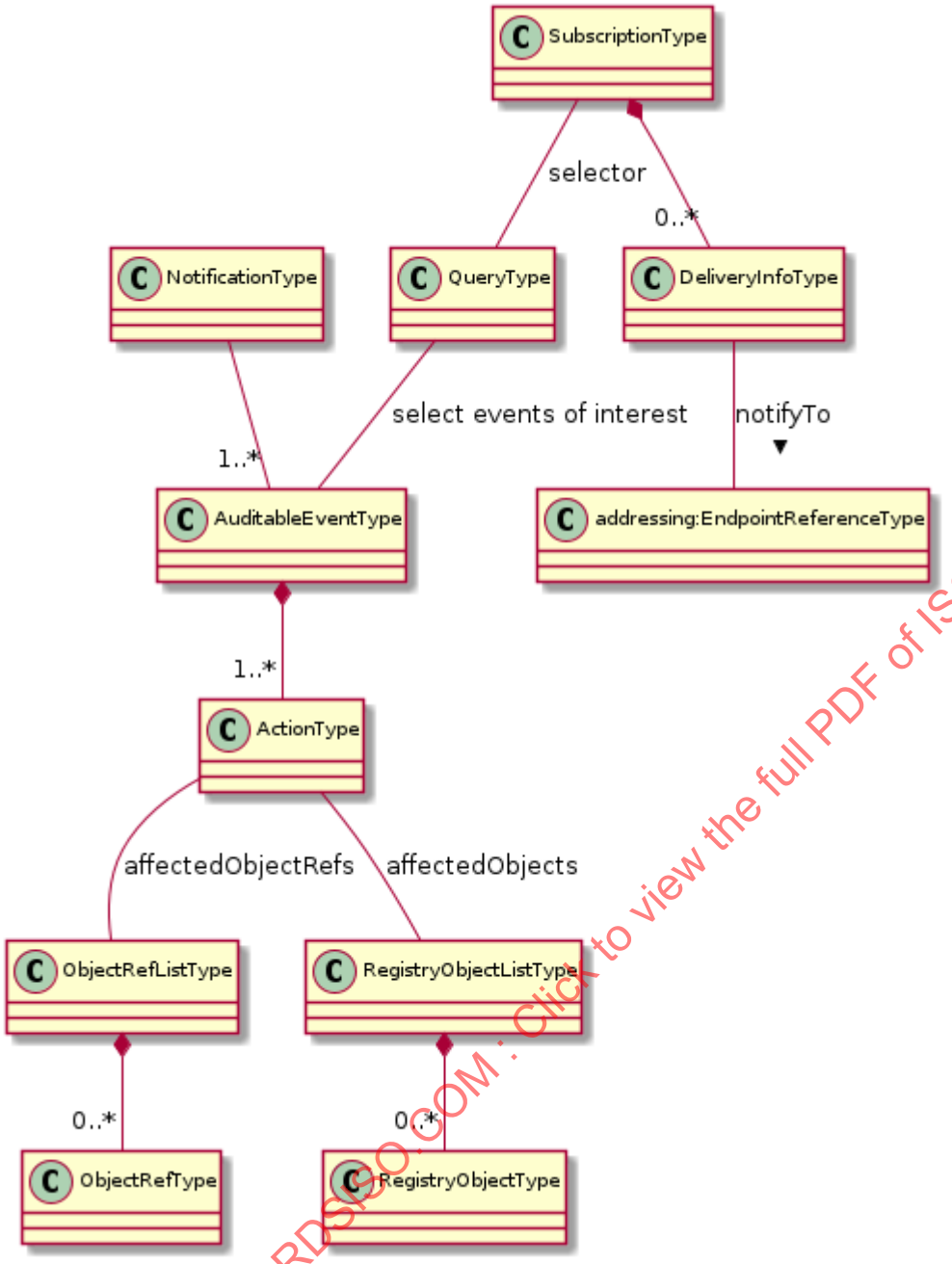

Figure 9 Event information model

4.8.2 AuditableEventType

4.8.2.1 Overview

Base Type: RegistryObjectType

This type represents a server event. AuditableEventType instances provide a long-term record of events that effected changes in the state of a RegistryObjectType instance. AuditableEventType instances shall be generated by the server and shall not be submitted by clients.

AuditableEventType instances represent a change in the state of a RegistryObjectType instance. For example a client request could Create, Update, Deprecate or Delete a RegistryObjectType instance. An AuditableEventType instance is created when a request creates or alters the state of a

RegistryObjectType instance. Read-only requests typically do not generate an AuditableEventType instance.

4.8.2.2 XML schema type definition

```
<complexType name="AuditableEventType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="Action" type="tns:ActionType"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="timestamp" type="dateTime" use="required"/>
      <attribute name="user" type="string" use="required"/>
      <attribute name="requestId"
        type="string" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: an AuditableEventType instance that logs the creation of an object within the context of a client request.

```
<rim:RegistryObject xsi:type="rim:AuditableEventType"
  requestid="urn:uuid:24cee176-9098-4931-894f-fea5dab1732a"
  timestamp="2008-01-10T19:20:30+01:00" user="farid"
  ...>
  <rim:Action eventType="urn:oasis:names:tc:ebxml-regrep:EventType:Created">
    <rim:AffectedObjectRefs>
      <rim:ObjectRef id="urn:acme:person:Danyal" />
    </rim:AffectedObjectRefs>
  </rim:Action>
</AuditableEvent>
```

4.8.2.3 Description

Table 36 shows the description of AuditableEventType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Action	ActionType	1..*		Registry	No
requestId	xs:string	1		Registry	No
timestamp	xs:dateTime	1		Registry	No
user	xs:string	1		Registry	No

Table 36 AuditableEventType

a) Element Action – Represents an action taken by the server within the context of an AuditableEventType instance. An AuditableEventType instance shall have one or more Action instances.
b) Attribute requestId – Specifies the id of the request that generated the AuditableEventType instance.
c) Attribute timestamp – Specifies the timestamp that represents the date and time the event occurred.
d) Attribute user – Specifies the id of the registered user associated with the client that made the request to the server that generated the AuditableEventType instance. Note that the inherited attribute owner should be set by a server to an internal system user since it is the server and not the user associated with the request that creates an AuditableEventType instance

4.8.3 ActionType

4.8.3.1 Overview

Base Type: ExtensibleObjectType

This type represents an action taken by the server within the context of an AuditableEventType instance.

4.8.3.2 XML schema type definition

```
<complexType name="ActionType">
    <complexContent>
      <extension base="tns:ExtensibleObjectType">
        <sequence>
          <!-- List of all objects affected by this event-->
          <element name="AffectedObjects" type="tns:RegistryObjectListType"
                 minOccurs="0" maxOccurs="1"/>
          <element name="AffectedObjectRefs" type="tns:ObjectRefListType"
                 minOccurs="0" maxOccurs="1"/>
        </sequence>
        <attribute name="eventType" type="tns:objectReferenceType" use="required"/>
      </extension>
    </complexContent>
  </complexType>
```

4.8.3.3 Description

Table 37 shows the description of ActionType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
AffectedObjects	RegistryObjectListType	0..1		Registry	No
AffectedObjectRefs	ObjectRefListType	0..1		Registry	No
eventType	objectReferenceType	1		Registry	No

Table 37 ActionType

a) Element AffectedObject – Identifies the RegistryObjectType instances that were affected by the event. The AffectedObject element contains any number of elements of type RegistryObjectType, each of which is a RegistryObjectType instance affected by the event. If this element is present then AffectedObjectRefs element shall not be present.

b) Element AffectedObjectRefs – Identifies the RegistryObjectType instances that were affected by the event. The AffectedObject element contains any number of ObjectRef elements each of which reference a RegistryObjectType instance that was affected by the event. If this element is present then AffectedObjects element shall not be present.

c) Attribute eventType – Specifies the type of event associated with the Action within an AuditableEventType instance.

   The value of the eventType attribute shall be a reference to a ClassificationNode in the canonical EventType ClassificationScheme.

   1) A Registry shall support the event types as defined by the EventType ClassificationScheme.

   2) The canonical EventType ClassificationScheme may easily be extended by adding additional ClassificationNodes to it

4.8.4 SubscriptionType

4.8.4.1 Overview

Base Type: RegistryObjectType

This type represents a subscription on behalf of a client to receive notifications by the server of events that are of interest to the client.

4.8.4.2 XML schema type definition

```
<complexType name="SubscriptionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="DeliveryInfo"
          type="tns:DeliveryInfoType" minOccurs="0" maxOccurs="unbounded" />
        <element name="Selector"
          type="tns:QueryType" minOccurs="1" maxOccurs="1" />
      </sequence>
      <attribute name="startTime" type="dateTime" use="optional"/>
      <attribute name="endTime" type="dateTime" use="optional"/>
      <attribute name="notificationInterval"
        type="duration" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: subscription to receive notification of changes to the object whose id value matches "urn:acme:person:Danyal". The DeliveryInfo specifies the SOAP endpoint where the server should deliver the Notification.

```
<rim:RegistryObject xsi:type="rim:SubscriptionType"
  id="urn:acme:Subscription:subscribeToDanyal"
  startTime="2008-01-10T19:20:30+01:00" endTime="2009-01-10T19:20:30+01:00"
  ...>
  <DeliveryInfo>
    <NotifyTo>
      <wsa:Address rim:endpointType="urn:oasis:names:tc:ebxml-
regrep:endPointType:soap">http://www.acme.com/notificationListener</wsa:Address>
    </NotifyTo>
  </DeliveryInfo>
  <Selector queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
    <Slot name="id">
      <SlotValue xsi:type="rim:StringValueType">
        <Value>urn:acme:person:Danyal</Value>
      </SlotValue>
    </Slot>
  </Selector>
</rim:RegistryObject>
```

4.8.4.3 Description

Table 38 shows the description of SubscriptionType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
DeliverInfo	DeliveryInfoType	0..*		Client	Yes
endTime	xs:dateTime	0..1		Client	Yes
notificationInterval	xs:duration	0..1		Client	Yes

Selector	QueryType	1		Client	Yes
startTime	xs:dateTime	0..1	Time of submission	Client	Yes

Table 38 SubscriptionType

a) Attribute startTime, endTime – Define the time window within which the subscription is valid.
   1) A server shall use the current time at the time of submission of Subscription as value for the startTime attribute if it is unspecified.
   2) The Subscription validity window shall be inclusive of the startTime and endTime.
   3) If endTime is unspecified then a server shall assume the Subscription is valid at any time any time since startTime inclusively.

b) Element DeliveryInfo – Specifies the information needed by the server to deliver notifications for the subscription. It includes the reference to the endpoint where notifications should be delivered.
   1) A server shall deliver notifications that match the Selector query for a valid SubscriptionType instance to the endpoint specified by each DeliveryInfo element of the SubscriptionType instance.
   2) If no DeliveryInfo element is present then client shall use the canonical query GetNotification via the Query protocol to "pull" the pending notification if any at a time of their choosing as defined in ebRS.

c) Attribute notificationInterval – Specifies the duration that a server shall wait between delivering successive notifications to the client. The client specifies this attribute in order to control the frequency of notification communication between server and client.
   1) A server shall deliver any pending notifications within the interval specified by this attribute.
   2) A server shall not deliver the same event more than once for the same subscription.

d) Element Selector – Specifies the query that the server shall invoke to determine whether an event matches a subscription or not. If the result of the query contains an object that is affected by an event that has not yet been delivered to the subscriber then the event matches the subscription.

4.8.5 DeliveryInfoType

4.8.5.1 Overview

Base Type: ExtensibleObjectType

This type provides the information needed by the server to *deliver* notifications for the subscription. It includes the reference to the endpoint where notifications should be delivered. The endpoint reference is typically one of the following types:

a) SOAP service endpoint
b) REST service endpoint
c) E-mail address endpoint
d) Software plugin endpoint that is configured within the same process as the registry server

4.8.5.2 XML schema type definition

```
<complexType name="DeliveryInfoType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="NotifyTo"
          type="wsa:EndpointReferenceType" minOccurs="1" maxOccurs="1" />
      </sequence>
      <attribute name="notificationOption" type="tns:objectReferenceType"
default="urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs"/>
    </extension>
  </complexContent>
</complexType>
```

4.8.5.3 Description

Table 39 shows the description of ObjectRefType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
notificationOption	objectReferenceType	0..1		Client	Yes
NotifyTo	wsa:EndpointReferenceType	1		Client	Yes

Table 39 DeliveryInfoType

a) Attribute notificationOption – Specifies the modality of how notifications are to be delivered to the subscriber. Its value shall reference a ClassificationNode in the canonical NotificationOptionType ClassificationScheme.

   1) urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:Objects – Indicates that the server shall provide complete RegistryObjectType instances in notifications delivered to the subscriber when this mode is specified.

   2) urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs – Indicates that the server shall provide ObjectRefType instances rather than complete RegistryObjectType instances in notifications delivered to the subscriber when this mode is specified. A client may pull the complete RegistryObjectType instances using Query protocol after receiving the notification.

b) Element NotifyTo – Specifies the endpoint reference for the endpoint where the server should deliver notifications for the Subscription.

   1) The type of this element is wsa:EndpointReferenceType as defined by the Web Services Addressing 1.0 – Core recommendation.

   2) The NotifyTo element has a <wsa:Address> sub-element

   3) The content of the <wsa:Address> element is a string representing the endpoint address which should be a URI

   4) The type of endpoint (SOAP, REST, email, ...) is indicated by an extension attribute rim:endpointType defined on the <wsa:Address> element as follows:

      a) If endpoint is a SOAP web service then the rim:endpointType attribute value shall be “urn:oasis:names:tc:ebxml-regrep:endPointType:soap”

      b) If endpoint is a REST web service then the rim:endpointType attribute value shall be “urn:oasis:names:tc:ebxml-regrep:endPointType:rest”

      c) If endpoint is an email address then the rim:endpointType attribute value shall be “urn:oasis:names:tc:ebxml-regrep:endPointType:mail”

d) If endpoint is a software plugin then the rim:endpointType attribute value shall be "urn:oasis:names:tc:ebxml-regrep:endPointType:plugin"

4.8.6 NotificationType

4.8.6.1 Overview

Base Type: RegistryObjectType

This type represents a notification that is sent by the server to a client to notify it of server events that are of interest to the client.

4.8.6.2 XML schema type definition

```
<complexType name="NotificationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
        <sequence>
          <element name="Event" type="tns:AuditableEventType"
            minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
      <attribute name="subscription"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
<element name="Notification" type="tns:NotificationType"/>
```

EXAMPLE: a Notification sent by the server for the subscription in earlier example. It notifies the subscriber that the object with id "urn:acme:person:Danyal" has changed.

```
<Notification subscription="urn:acme:Subscription:subscribeToDanyal" ...>
  <Event user="123456" timestamp="2008-10-17T15:44:29.637" ...>
    <Action eventType="urn:oasis:names:tc:ebxml-regrep:EventType:Created">
      <AffectedObjectRefs>
        <ObjectRef id="urn:acme:person:Danyal"/>
      </AffectedObjectRefs>
    </Action>
  </Event>
</Notification>
```

4.8.6.3 Description

Table 40 shows the description of NotificationType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
Event	AuditableEventType	1..*		Server	No
subscription	objectReferenceType	1		Server	No

Table 40 NotificationType

a) Element Event – Represents an Event that is of interest to the subscriber.

1) Unlike an AuditableEvent element that contains all objects affected by it, the Event element shall only contain objects that match the selector query of the SubscriptionType instance. It has only a subset of affected objects compared to the actual AuditableEvent it represents. The subset of affected objects shall be those that match the selector query for the subscription.

2) The Action elements within the Event element shall contain a RegistryObjectList element if subscription's notificationOption is “Push”.

3) The Action elements within the Event element shall contain a RegistryObjectRefList element if subscription's notificationOption is “Pull”.

b) Attribute subscription – References the SubscriptionType instance for which this is a Notification.

4.9 Federation information model

4.9.1 Overview

This clause describes the information model that supports the definition of registry federations. A registry federation is a set of ebXML RegRep servers that have voluntarily agreed to form a loosely coupled union. Such a federation may be based on common business interests or membership in a community-of-interest. Registry federations enabled clients to query the content of their member servers using federated queries as if they are a single logical server.

4.9.2 Federation configuration

A federation is created by the creation of a FederationType instance. A federation may have any number of registries as well as other federations as its members.

Membership of a registry or federation within a parent federation is established by creating an Association between the RegistryType or FederationType instance representing the registry or federation seeking membership, and the FederationType instance representing the parent federation as follows:

a) The Association shall have its associationType be the id of the canonical ClassificationNode “HasFederationMember”
b) The Association shall have as its sourceObject the FederationType instance representing the parent federation
c) The Association shall have as its targetObject the RegistryType or FederationType instance that is seeking membership within the parent federation as shown in Figure 10.

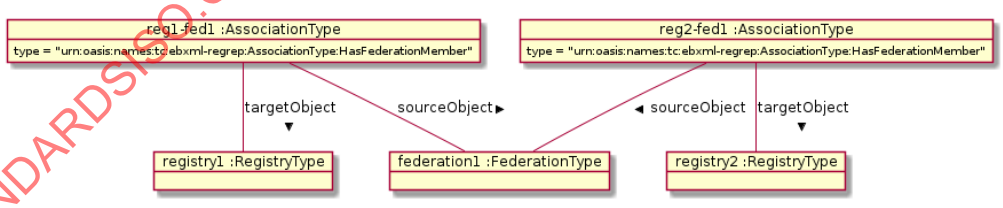

Figure 10 Federation information model

Thus a Federation is defined by a tree where a FederationType instances are root and intermediate n, RegistryType instances are leaf nodes and HasFederationMember AssociationType instances are the edges between the nodes. This tree is referred to as the federation membership tree.

4.9.3 RegistryType

4.9.3.1 Overview

Base Type: RegistryObjectType

RegistryType instances are used to represent an ebXML RegRep server. RegistryType instances are also used by a server to advertise the capabilities it supports. A client may read the RegistryType instance for a server to determine whether it is compatible with a server or not. Profiles of this specification may define canonical slots to represents support for the profile as well as optional features defined by the profile.

4.9.3.2 XML schema type definition

```
<complexType name="RegistryType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="operator"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="specificationVersion"
        type="string" use="required"/>
      <attribute default="P1D" name="replicationSyncLatency"
        type="duration" use="optional"/>
      <attribute default="PT0S" name="catalogingLatency"
        type="duration" use="optional"/>
      <attribute name="conformanceProfile"
        use="optional" default="RegistryLite">
        <simpleType>
          <restriction base="NCName">
          <enumeration value="RegistryFull"/>
            <enumeration value="RegistryLite"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: an ebXML RegRep server operated by organization with id “urn:acme:Organization:acme-inc”, that implements the “RegistryFull” conformance level of version 4.0 of this specification. The server performs replication synchronization once a day (P1D) and performs cataloging of submitted content immediately when content is submitted.

```
<rim:RegistryObject xsi:type="rim:RegistryType"
  id="urn:acme:Registry:serviceRegistry"
  operator="urn:acme:Organization:acme-inc"
  specificationVersion="4.0"
  conformanceProfile="RegistryFull"
  replicationSyncLatency="P1D"
  catalogingLatency="PT0S"
  ...>
  ...
</rim:RegistryObject>
```

4.9.3.3 Description

Table 41 shows the description of RegistryType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
catalogingLatency	xs:duration	0..1	P1D (once a day)	Server	Yes
conformanceProfile	xs:string	0..1	RegistryLite	Server	Yes
operator	objectReferenceType	1		Server	Yes

Node	Type	Cardinality	Default Value	Specified By	Mutable
replicationSyncLatency	xs:duration	0..1	PT0S (immediately)	Server	Yes
specificationversion	objectReferenceType	1		Server	Yes

Table 41 RegistryType

a) Attribute catalogingLatency - A RegistryType instance may have an attribute named *catalogingLatency* that specifies the maximum latency between the time a submission is made to the server and the time it gets cataloged by any cataloging services defined for the objects within the submission. The default value of PT0S indicates a duration of 0 seconds which implies that cataloging happens immediately when request is submitted.
b) Attribute conformanceProfile - A RegistryType instance may have an attribute named *conformanceProfile* that declares the conformance profile that the server supports. The conformance profiles choices are "RegistryLite" and "RegistryFull" as defined by the registry services (see clause 5).
c) Attribute operator - A RegistryType instance shall have an attribute named *operator* that is a reference to the Organization instance representing the organization for the server's operator. Since the same Organization may operate multiple registries, it is possible that the home registry for the Organization referenced by operator may not be the local registry.
d) Attribute replicationSyncLatency - A RegistryType instance may have an attribute named *replicationSyncLatency* that specifies the maximum latency between the time when an original object changes and the time when its replica object within the local server gets updated to synchronize with the new state of the original object. The default value of P1D indicates a duration of once a day.
e) Attribute specificationVersion - A RegistryType instance shall have an attribute named *specificationVersion* that is the version of this specification it implements.

4.9.4 FederationType

4.9.4.1 Overview

Base Type: RegistryObjectType

Federation instances are used to represent a registry federation. A FederationType instance has a set of RegistryType instances as its members. The membership of a RegistryType instance in a federationType instance is represented by an AssociationType instance whose type is HasFederationMember.

4.9.4.2 XML schema type definition

```
<complexType name="FederationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="replicationSyncLatency"
        type="duration" use="optional" default="P1D" />
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE: a Federation with two independently-operated ebXML RegRep servers as members.

```
<rim:RegistryObject xsi:type="rim:FederationType"
  id="urn:acme:Federation:supplierFederation"
  replicationSyncLatency="P1D" ...>
  ...
```

```
</rim:RegistryObject>

<rim:RegistryObject xsi:type="rim:AssociationType"
  sourceObject="urn:acme:Federation:supplierFederation"
  targetObject="urn:widgetInc:Registry:widget-inc"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>

<rim:RegistryObject xsi:type="rim:AssociationType"
  sourceObject="urn:acme:Federation:supplierFederation"
  targetObject="urn:supplierInc:Registry:supplier-inc"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>
```

4.9.4.3 Description

Table 42 shows the description of FederationType.

Node	Type	Cardinality	Default Value	Specified By	Mutable
replicationSyncLatency	xs:duration	0..1	P1D (1 day)	Client	Yes

Table 42 FederationType

Attribute replicationSyncLatency - A FederationType instance may specify a *replicationSyncLatency* attribute that describes the time duration that is the amount of time within which a member of this Federation shall synchronize itself with the current state of the Federation. Members of the Federation may use this parameter to periodically synchronize the federation metadata they shall cache locally about the state of the Federation and its members. Such synchronization may be based upon the registry event notification capability.

4.10 Access control information model

4.10.1 Overview

This clause defines the Information Model used to control access to RegistryObjects and RepositoryItems managed by it. It also defines a normative profile of the OASISeXtensible Access Control Markup Language (XACML) for ebXML RegRep.

It is assumed that the reader is already familiar with XACML. This specification does not provide any introduction to XACML.

A server shall support the roles of both Enforcement Point (PEP) and a Policy Decision Point (PDP) as defined in the OASISeXtensible Access Control Markup Language standard.

The Access Control Model attempts to reuse terms defined by the OASISeXtensible Access Control Markup Language wherever possible. The definitions of some key terms are duplicated in Table 43 from the OASISeXtensible Access Control Markup Language for convenience of the reader.

Term	Description
Access	Performing an action. EXAMPLE 1: a user performing a delete action on a RegistryObject.
Access Control	Controlling access in accordance with a policy. EXAMPLE 2: preventing a user from performing a delete action on a RegistryObject that is not owned by that user.

Action	An operation on a resource. EXAMPLE 3: the delete action on a RegistryObject.
Attribute	Characteristic of a subject, resource, action. Some examples are: a) id attribute of a subject b) role attribute of a subject c) group attribute of a subject id attribute of a RegistryObject resource
Policy	A set of rules. May be a component of a policy set
PolicySet	A set of policies, other policy sets. May be a component of another policy set
Resource	Data, service or system component. Examples are: a) A RegistryObject resource b) A RepositoryItem resource
Subject	An actor whose attributes may be referenced by within a Policy definition. Examples of subject include: a) The registered user associated with a client request b) An ebXML registry server c) A software service or agent

Table 43 Access control information model terminology

4.10.2 Defining an access control policy

A RegistryObjectType instance is associated with exactly one Access Control Policy that governs "who" is authorized to perform "what" action on that RegistryObject. This Access Control Policy is expressed as an XACML document which is the repositoryItem for an ExtrinsicObjectType instance. The Access Control Policy is published to the server as an ExtrinsicObject and repositoryItem pair using the Submit protocol defined by the registry services (see clause 5).

The objectType attribute of this ExtrinsicObject shall reference a descendent of the "XACML" ClassificationNode (e.g. "Policy" or PolicySet") in the canonical ObjectType ClassificationScheme.

4.10.3 Assigning access control policy to a registry object

4.10.3.1 Overview

An Access Control Policy may be assigned to a RegistryObjectType instance using the canonical slot "urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy". The value slot references the ExtrinsicObject representing the Access Control Policy and contains the id of that ExtrinsicObject.

If a RegistryObjectType instance does not have an Access Control Policy explicitly associated with it via the canonical slot with name "urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy", then it is implicitly associated with the default Access Control Policy defined for the server.

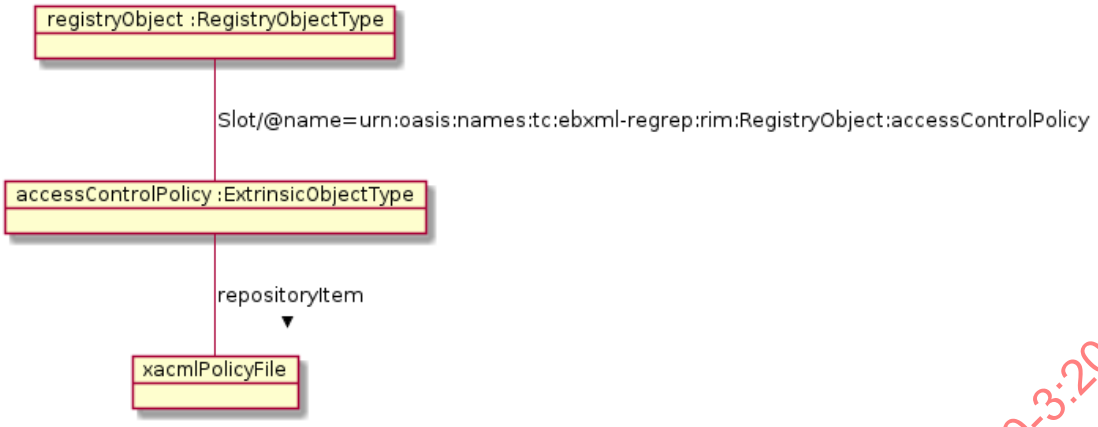

Figure 11 Assigning access control policy to a registry object

Figure 11 shows a UML instance diagram where an Organization instance org references an ExtrinsicObject instance accessControlPolicy as its Access Control Policy object using the canonical accessControlPolicy slot.

4.10.3.2 **Default Access Control Policy for a RegistryObject**

A server shall support a default Access Control Policy. A server may implement any default access control policy. The default Access Control Policy applies to all RegistryObjectType instances that do not explicitly have an Access Control Policy assigned.

This following specify the semantics of a suggested default Access Control Policy that a server should implement:

a) An unauthenticated client is permitted to perform read actions (that do not modify the state of resources) on any resource
b) An authenticated client with authentication credentials of a registered user is permitted all actions on RegistryObjects submitted by the user
c) A authenticated client with authentication credentials that are assigned the canonical subject role of "urn:oasis:names:tc:ebxml-regrep:SubjectRole:RegistryAdministrator" is permitted to perform any action on any object

4.10.3.3 **Access control policy inheritance**

4.10.3.3.1 Overview

A RegistryObjectType instance that does not explicitly define an Access Control Policy may inherit an Access Control Policy from its nearest RegistryPackageType ancestor if it is in the membership hierarchy of a RegistryPackageType instance.

An Access Control Policy for members of a RegistryPackageType instance may be assigned to the RegistryPackageType instance using the canonical slot “urn:oasis:names:tc:ebxml-regrep:rim:RegistryPackage:memberAccessControlPolicy”. The value slot references the ExtrinsicObject representing the Access Control Policy and contains the id of that ExtrinsicObject. The member Access Control Policy is implicitly inherited as the applicable Access Control Policy for any member RegistryObjectType instance that does not have an explicit Access Control Policy assigned to it.

In the event that a RegistryObjectType instance has a member Access Control Policy defined from two RegistryPackageType ancestors at the same ancestry level a server may choose any mechanism to select one of the two member Access Control Policies.

4.10.3.3.2 Algorithm for getting applicable access control policy

A server shall implement the following algorithm for determining the applicable Access Control Policy for a RegistryObjectType instance:

a) If an Access Control Policy is explicitly assigned to the object then use it
b) If no Access Control Policy is explicitly assigned to the object then get the member Access Control Policy from a nearest RegistryPackageType ancestor of the object
c) If no Access Control Policy is explicitly assigned to the object or inherited from a RegistryPackageType ancestor of the object then use the system-wide default Access Control Policy

4.10.3.4 Performance implications

Excessive use of custom Access Control Policies may result in slower processing of registry requests in some registry implementations. It is therefor suggested that, whenever possible, a submitter should reuse an existing Access Control Policy. Submitters should use good judgment on when to reuse or extend an existing Access Control Policy and when to create a new one.

4.10.4 Defining a contextual role

4.10.4.1 RoleType

A contextual role may be defined by a RoleType instance within a server as defined next.

Base Type: RegistryObjectType

4.10.4.2 XML schema type definition

```
<complexType name="RoleType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="type" type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE" a RoleType instances representing a contextual role of "ProjectLead" within the organizational context of an Organization TestSubmittingOrg1.

```
<rim:RegistryObject xsi:type="rim:RoleType"
  type="urn:oasis:names:tc:ebxml-regrep:SubjectRole:ProjectLead" ...>
  <rim:Slot name="urn:oasis:names:tc:ebxml-regrep:RoleAssociation:organizationContext">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>urn:test:Organization:TestSubmittingOrg1</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
</rim:RegistryObject>
```

4.10.4.3 Description

Table 44 shows the description of RoleType.

Node	Type	Cardinality	Default Value	Specified By	Mutable

type	objectReferenceType	1		Client	Yes

Table 44 RoleType

a) Attribute type - Each RoleType instance shall have a t*ype* attribute that identifies the type of role.

   1) The value of the type attribute should be a reference to a ClassificationNode within the canonical SubjectRole ClassificationScheme.

   2) A server shall support the canonical subject role types as defined by the canonical SubjectRole ClassificationScheme. Deployments and profiles may extend the canonical SubjectRole ClassificationScheme by adding additional ClassificationNodes to it.

b) The RoleType instance may have any number of Slots that provide context for the role as a name/value pair. The name attribute of each such context slot provides the context key while the value of the Slot (typically a string) provides the context value

4.10.5 Assigning a contextual role to a subject

A subject such as a registered user may be assigned a contextual role by associating the id attribute value of the RoleType instance representing the contextual role with the id of the subject. This specification does not define how such an association is made. Implementations may provide this association in an implementation specific manner.

EXAMPLE: in an LDAP based identity management system this may be done by making the node representing the subject to be a member of a groupOfName node as illustrated below:

```
#person ProjectLead1 definition
dn: uid=ProjectLead1,...
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Project Lead 1
sn: ProjectLead1
uid: ProjectLead1

#Role TestSubmittingOrg1ProjectLead definition
dn: cn=urn:test:Role:TestSubmittingOrg1ProjectLead,...
objectclass: top
objectclass: groupOfNames
cn: urn:test:Role:TestSubmittingOrg1ProjectLead
member: uid=ProjectLead1,ou=people,...
```

4.10.6 Action matching

4.10.6.1 Overview

An XACML Access Control Policy may use an action identifier associated with the action as an action attribute within <xacml:ActionMatch> elements to match the action that is authorized for a subject on a resource.

The following requirements are defined for a server for action matching:

a) A server shall specify the action identifier in the request context for an XACML decision request using a <xacmlc:Request>/<xacmlc:Action>/<xacmlc:Attribute> element

   1) The <xacmlc:Attribute> element shall have an AttributeId attribute with content "urn:oasis:names:tc:xacml:1.0:action:action-id"

2) The <xacmlc:Attribute> element shall have a DataType attribute with value "http://www.w3.org/2001/XMLSchema#string"

3) The <xacmlc:Attribute> shall have a <xacmlc:AttributeValue> element whose content shall be the id attribute of the ClassificationNodeType instance within the canonical ActionTypeScheme that represent the requested action

b) The following requirements are defined for an Access Control Policy for action matching:

b) The policy may match an action using an <xacmlp:ActionMatch> element

c) The <xacmlp:ActionMatch> element shall have a <xacmlp:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content shall be the id attribute of a ClassificationNodeType instance within the canonical ActionTypeScheme

d) The <xacmlp:ActionMatch> element shall have a <xacmlp:ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string"/> element

EXAMPLE: the following example shows an Action that matches the "Read" action.

```
<Target>
  <Actions>
    <Action>
      <ActionMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
          urn:oasis:names:tc:ebxml-regrep:ActionType:read
        </AttributeValue>
        <ActionAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </ActionMatch>
    </Action>
  </Actions>
</Target>
```

4.10.6.2 **ActionAttribute: *reference-source***

This attribute is only relevant to the "Reference" action. This attribute may be used to specify the object from which the reference is being made to the resource being protected. The AttributeId of this attribute shall be "urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source". The value of this attribute shall be the value of the id attribute for the object that is the source of the reference. A server shall specify this attribute for a reference action.

4.10.6.3 **ActionAttribute: reference-source-attribute**

This attribute is only relevant to the "Reference" action. This attribute may be used to specify the attribute name within the RegistryObjectype that the reference-source object is an instance of. A server shall specify this attribute for a reference action. The AttributeId of this attribute shall be "urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source-attribute". The value of this attribute shall be the name of an attribute within the RIM type that is the type for the reference source object.

EXAMPLE: if the reference source object is an Association instance then the reference-source-attribute may be used to specify the values "sourceObject" or "targetObject" to restrict the references to be allowed from only specific attributes of the source object. This enables, for example, a policy to only allow reference to objects under its protection only from the sourceObject attribute of an Association instance.

4.10.7 Subject matching

4.10.7.1 Overview

An XACML Access Control Policy may use the identity and roles associated with the subject as subject attributes within XACML SubjectMatch elements to match the subject that is authorized for an action on a resource.

The following requirements are defined for a server for subject matching:

a) A server shall specify the subject identifier in the request context for an XACML decision request using a `<xacmlc:Request>/<xacmlc:Subject>/<xacmlc:Attribute>` element

   1) The Attribute element shall have an AttributeId attribute with value "urn:oasis:names:tc:xacml:1.0:subject:subject-id"

   2) The Attribute element shall have a DataType attribute with value “http://www.w3.org/2001/XMLSchema#string”

   3) The Attribute shall have an AttributeValue element whose content shall be the unique id associated with the requestor

b) A server shall specify any subject roles in the request context for an XACML decision request using a `<xacmlc:Request>/<xacmlc:Subject>/<xacmlc:Attribute>` element. This specification does not define how roles are assigned to a subject. Implementations should provide that functionality in an implementation-specific manner.

   1) The Attribute element shall have an AttributeId attribute with value "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:role"

   2) The Attribute element shall have a DataType attribute with value “http://www.w3.org/2001/XMLSchema#string”

   3) The Attribute shall have an AttributeValue element whose content shall be the id attribute value of a RoleType instance associated with the requestor

4.10.7.2 Matching subjects by id

The following requirements are defined for an Access Control Policy for subject matching by the subject id:

The policy may match a subject by id using an XACML `<xacmlp:SubjectMatch>` element

   1) The SubjectMatch element shall have an AttributeValue DataType element whose content shall be the unique id of the subject

   2) The SubjectMatch element shall have an `<xacmlp:SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema#string"/>` element

EXAMPLE: the following example shows a Subject that matches a registered user with id “urn:acme:person:Danyal”:

```
<Target>
  <Subjects>
    <Subject>
      <SubjectMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
```



```
          urn:acme:person:Danyal
        </AttributeValue>
        <SubjectAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
    </Subject>
  </Subjects>
</Target>
```

4.10.7.3 **Matching subject by role**

The following requirements are defined for an Access Control Policy for subject matching by a subject role:

The policy may match a subject by a contextual role using an `<xacmlp:Condition>` element

The <xacmlp:Condition> element shall use an `<xacmlp:Apply/@FunctionId='urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role'>` element to invoke the “matches-role” XACML function as defined by this specification

EXAMPLE: the following example shows a Subject that matches a subject role “ProjectLead” within the organizational context of Organization TestSubmittingOrg1 and register context of Register TestRegister1:

```
<Rule Effect="Permit" RuleId="urn:test:customACP1:rule:restricted-delete">
  <Target/>

  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role">
      <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:subject:role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:SubjectRole:ProjectLead</AttributeValue>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:RoleAssociation:organizationContext</AttributeValue>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Organization:TestSubmittingOrg
1</AttributeValue>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:RoleAssociation:registerContext</AttributeValue>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Register:TestRegister1</Attrib
uteValue>
    </Apply>
  </Condition>

</Rule>
```

4.10.8 Resource matching

4.10.8.1 **Overview**

An XACML Access Control Policy may use the id associated with the resource as resource attributes within <xacml:ResourceMatch> elements to match a resource within an authorization decision for an action on the resource.

The following requirements are defined for a server for resource matching:

a) A server shall specify the resource identifier in the request context for an XACML decision request using a <xacmlc:Request>/<xacmlc:Resource>/<xacmlc:Attribute> elementThe <xacmlc:Attribute> element shall have an AttributeId attribute with value "urn:oasis:names:tc:xacml:1.0:resource:resource-id"

   1) The <xacmlc:Attribute> element shall have a DataType attribute with value “http://www.w3.org/2001/XMLSchema#string”

   2) The <xacmlc:Attribute> shall have a <xacmlc:AttributeValue> element whose content shall be the unique id associated with the resource

b) A server shall specify the owner attribute value of the RegistryObjectType resource in the request context for an XACML decision request using a <xacmlc:Request>/<xacmlc:Resource>/<xacmlc:Attribute> element

   1) The <xacmlc:Attribute> element shall have an AttributeId attribute with value "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:owner"

   2) The <xacmlc:Attribute> element shall have a DataType attribute with value “http://www.w3.org/2001/XMLSchema#string”

   3) The <xacmlc:Attribute> shall have a <xacmlc:AttributeValue> element whose content shall be the owner attribute value of the RegistryObjectType resource

4.10.8.2 **Matching a resource by id**

The following requirements are defined for an Access Control Policy for resource matching by the resource's id:

The policy may match a resource by id using an <xacmlp:ResourceMatch> element

   1) The <xacmlp:ResourceMatch> element shall have a <xacmlp:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content shall be the unique id of the resource

   2) The <xacmlp:ResourceMatch> element shall have a <xacmlp:ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#string"/> element

EXAMPLE: the following example shows a ResourceMatch that matches a resource with id “urn:acme:person:Danyal”:

```
<Target>
  <Resources>
    <Resource>
      <ResourceMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
          urn:acme:person:Danyal
        </AttributeValue>
        <ResourceAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </ResourceMatch>
```



```
    </Resource>
  </Resources>
</Target>
```

4.10.8.3 Matching a Resource Using XPATH Expression

An XACML Access Control Policy may use any node in the XML document representing a RegistryObjectType instance within an <xacml:ResourceMatch> element. In this case, the <xacml:ResourceMatch> element should use an XPATH expression to match any part of the XML element representing the RegistryObjectType instance.

EXAMPLE: the following example uses XPATH expression to match resource if it has a Slot with name "someSlotName".

```
<Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
  <ResourceMatch
    MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      //<rim:Slot>/@name="someSlotName"
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
</Resource>
```

4.10.9 Canonical XACML functions

4.10.9.1 Overview

Clause A.3 of the OASIS eXtensible Access Control Markup Language standard defines a set of standard functions. This clause defines addition XACML functions that shall be supported by an ebXML RegRep server that supports XACML based custom access control policies. XACML specifies the following functions. If an argument of one of these functions were to evaluate to "Indeterminate", then the function shall be set to "Indeterminate".

4.10.9.2 Function AssociationExists

Function ID: urn:oasis:names:tc:ebxml-regrep:rim:acp:function:AssociationExists

Table 45 shows the description of AssociationExists.

Parameter / Return	Name	Description	Data Type
Parameter 1	sourceObject	Specifies a value for the sourceObject attribute of AssociationType. may use '%' and '_' as wildcard to match multiple or single characters.	http://www.w3.org/2001/XMLSchema#string

Paremeter 2	targetObject	Specifies a value for the targetObject attribute of AssociationType. may use '%' and '_' as wildcard to match multiple or single characters.	http://www.w3.org/2001/XMLSchema#string
Parameter 3	type	Specifies the path attribute value for a ClassificationNode in the AssociationType ClassificationScheme. may use '%' and '_' as wildcard to match multiple or single characters. This attribute is used to match the type attribute of AssociationType. The type parameter shall also match ClassificationNodes that are descendants of ClassificationNode specified by the type parameter. This parameter is optional and may be omitted.	http://www.w3.org/2001/XMLSchema#string
Returns		shall return “True” if and only if an AssociationType instance exists that matches the specified sourceObjectId, targetObjectId and type. shall return “False” otherwise.	http://www.w3.org/2001/XMLSchema#boolean

Table 45 Function AssociationExists

4.10.9.3 **Function ClassificationNodeCompare**

Function ID: urn:oasis:names:tc:ebxml-regrep:rim:acp:function:ClassificationNodeCompare

A client may use this XACML function to test whether a resource's objectType attribute matches a specific objectType or its sub-types.

Table 46 shows the description of ClassificationNodeCompare.

Parameter / Return	Name	Description	Data Type
Parameter 1	node1	Specifies the id of a ClassificationNode.	http://www.w3.org/2001/XMLSchema#string
Paremeter 2	node2	Specifies the id of a ClassificationNode.	http://www.w3.org/2001/XMLSchema#string
Returns		shall return “True” if and only if ClassificationNode with id matching node2 value is same	http://www.w3.org/2001/XMLSchema#boolean

		as or descendent of if ClassificationNode with id matching node1. shall return "False" otherwise.	

Table 46 Function ClassificationNodeCompare

4.10.9.4 **Function matches-role**

Function ID: urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role

Table 47 shows the description of matches-role.

Parameter / Return	Name	Description	Data Type
Parameter 1	roles	Specifies a bag containing ids of RoleType instances representing the contextual roles that a subject is expected to have	Bag of attributes of type http://www.w3.org/2001/XMLSchema#string
Paremeter 2	roleType	Specifies the id of a ClassificationNode within the canonical SubjectRole ClassificationScheme	http://www.w3.org/2001/XMLSchema#string
Subsequent parameters shall come in pairs where each pair specifies a context key/value pair. Any number of such pairs shall be supported by server implementing this function			
Parameter 3+N	contextKey	Specifies a context identifier	http://www.w3.org/2001/XMLSchema#string
4+N	contextValue	Specifies a context value associated with the context identifier specified by previous parameter	http://www.w3.org/2001/XMLSchema#string
Returns		shall return "True" if and only if at least one RoleType instance assigned to the subject meets the following conditions: a) If roleType is specified, then the type attribute of the RoleType instance shall match the role type ClassificationNode (or a descendant of it) specified by the roleType parameter b) If any context key/value pairs are specified then the RoleType instance shall have a Slot whose name matches the context key and whose value matches the context	http://www.w3.org/2001/XMLSchema#boolean

		value shall return “False” otherwise.	

Table 47 Function matches-role

4.10.10 Constraints on XACML binding

This specification normatively defines the following constraints on the binding of the Access Control Model to the OASISeXtensible Access Control Markup Language. These constraints may be relaxed in future versions of this specification.

All Policy and PolicySet definitions shall reside within an ebXML Registry as RepositoryItems.

4.10.11 Resolving policy references

An XACML PolicySet may reference XACML Policy objects defined outside the repository item containing the XACML PolicySet. A server implementation shall be able to resolve such references. To resolve such references efficiently a server should be able to find the repository item containing the referenced Policy without having to load and search all Access Control Policies in the repository. This clause describes the normative behavior that enables a server to resolve policy references efficiently.

A server should define a Content Cataloging Service for the canonical XACML PolicySet objectType. The PolicySet cataloging service shall automatically catalog every PolicySet upon submission to contain a special Slot with name ComposedPolicies. The value of this Slot shall be a Set where each element in the Set is the id for a Policy object that is composed within the PolicySet.

Thus a server is able to use an ad hoc query to find the repositoryItem representing an XACML PolicySet that contains the Policy that is being referenced by another PolicySet.

5 Registry services

5.1 Overview

This clause specifies the ebXML registry service interfaces (ebRS) and the protocols they support.

5.2 Abstract protocol

5.2.1 Overview

This clause describes the types RegistryRequestType, RegistryResponseType and RegistryExceptionType defined within rs.xsd that are the abstract types used by most protocols defined by this specification in subsequent clauses. A typical registry protocol is initiated by a request message that extends RegistryRequestType. In response the registry server sends a response that extends RegistryResponseType. If an error is encountered by the server during the processing of a request, the server returns a fault message that extends the RegistryExceptionType.

5.2.2 RegistryRequestType

5.2.2.1 Overview

Base Type: ExtensibleObjectType

The RegistryRequestType is the abstract base type for most requests sent by client to the server.

5.2.2.2 XML schema type definition

```
<complexType name="RegistryRequestType">
  <complexContent>
    <extension base="rim:ExtensibleObjectType">
      <attribute name="id" type="string" use="required"/>
      <attribute name="comment" type="string" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

5.2.2.3 Description

a) Attribute comment – The comment attribute if specified contains a String that describes the request. A server may save this comment within a CommentType instance and associate it with the AuditableEvent(s) for that request as described by the registry information model (see clause 4).
b) Attribute id – The id attribute must be specified by the client to uniquely identify a request. Its value should be a UUID URN like “urn:uuid:a2345678-1234-1234-123456789012”.

5.2.3 RegistryResponseType

5.2.3.1 Overview

Base Type: ExtensibleObjectType

The RegistryResponseType is the base type for most responses sent by the server to the client in response to a client request. A global RegistryResponse element is defined using this type which is used by several requests defined within this specification.

5.2.3.2 XML schema type definition

```
<complexType name="RegistryResponseType">
  <complexContent>
    <extension base="rim:ExtensibleObjectType">
      <sequence>
        <element name="Exception" type="tns:RegistryExceptionType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element ref="rim:RegistryObjectList" minOccurs="0" maxOccurs="1"/>
        <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1"/>
      </sequence>
      <attribute name="status" type="rim:objectReferenceType" use="required"/>
      <attribute name="requestId" type="anyURI" use="optional"/>
    </extension>
  </complexContent>
</complexType>
<element name="RegistryResponse" type="tns:RegistryResponseType"/>
```

5.2.3.3 Description

a) Element RegistryObjectList– Contains a sequence of zero or more RegistryObject elements. It is used by responses that return a list of RegistryObject instances.
b) Element ObjectRefList – Contains a sequence of zero or more ObjectRef elements. It is used by responses that return a list of references to RegistryObject instances
c) Attribute requestId – This attribute contains the id of the request that returned this QueryResponse.
d) Attribute status – This attribute contains the status of the response. Its value shall be a reference to a ClassificationNode within the canonical ResponseStatusType ClassificationScheme. A server shall support the status types as defined by the canonical ResponseStatusType ClassificationScheme. The canonical ResponseStatusType ClassificationScheme may be extended by adding additional ClassificationNodes to it.

The following canonical values are defined for the ResponseStatusType ClassificationScheme:

a) Failure - This status specifies that the request encountered a failure.
b) PartialSuccess - This status specifies that the request was partially successful. Certain requests such as federated queries allow this status to be returned.
c) Success - This status specifies that the request was successful.
d) Unavailable – This status specifies that the response is not yet available. This may be the case if this RegistryResponseType represents an immediate response to an asynchronous request where the actual response is not yet available.

5.2.4 RegistryExceptionType

5.2.4.1 Overview

Base Type: ExtensibleObjectType

The RegistryExceptionType is the abstract base type for all exception or fault messages sent by the server to the client in response to a client request. A list of all protocol exceptions is available in the Protocol Exceptions appendix.

5.2.4.2 XML schema type definition

```
<complexType name="RegistryExceptionType">
    <annotation>
        <documentation>Base for all registry exceptions. Based upon SOAPFault:
http://www.w3schools.com/soap/soap_fault.asp</documentation>
    </annotation>
    <complexContent>
      <extension base="rim:ExtensibleObjectType">
        <attribute name="code" type="string" use="optional"/>
        <attribute name="detail" type="string" use="optional"/>
        <attribute name="message" type="string"/>
        <attribute name="severity" type="rim:objectReferenceType"
default="urn:oasis:names:tc:ebxml-regrep:ErrorSeverityType:Error"/>
      </extension>
    </complexContent>
</complexType>
```

5.2.4.3 Description

In addition to the attributes and elements inherited from ExtensibleObjectType this type defines the following attributes and elements:

a) Attribute code – The code attribute value may be used by a server to provide an error code or identifier for an Exception.
b) Attribute detail – The detail attribute value may be used by a server to provide any detailed information such as a stack trace for an Exception.
c) Attribute message – The message attribute value shall be used by a server to provide a brief message summarizing an Exception.
d) Attribute severity – The severity attribute value provides a severity level for the exception. Its value should reference a ClassificationNode within the canonical ErrorSeverityType ClassificationScheme.

5.2.5 Server Plugins

Deployments of a server may extend the core functionality of the server by using function-specific software modules called plugins. A plugin extends the server by adding additional functionality to it. A plugin shall conform to standard interfaces as defined by this specification. These standard interfaces are referred to as Service Provider Interfaces (SPI).

Subsequent clauses will specifies various Service Provider Interfaces (SPI) that defines the standard interface for various types of server plugins. These interfaces are described in form of the Web Services Description Language (WSDL) Version 1.1 specification.

A server may implement these interfaces as external web services invoked by the server using as defined in the SOAP Version 1.2 Part1 Messaging Framework and the SOAP Version 1.2 Part2 – Adjuncts specifications or as plugin modules that share the same process as the server and are invoked by local function calls.

Examples of types of server plugins include, but are not limited to query plugin, validator plugin and cataloger plugin.

This specification does not define how a plugin is implemented or how it is configured within a server. Nor does it define whether or how, plugin configuration functionality is made discoverable to clients.

5.3 QueryManager interface

5.3.1 Overview

The QueryManager interface allows a client to invoke queries on the server.

5.3.2 Parameterized queries

5.3.2.1 Overview

A server may support any number of pre-configured queries known as *Parameterized Queries*, that may be invoked by clients. Parameterized queries are similar in concept to stored procedures in SQL.

This specification defines a number of canonical queries that are standard queries that shall be supported by a server. Profiles, implementations and deployments may define additional parameterized queries beyond the canonical queries defined by this specification.

A client invokes a parameterized query supported by the server by specifying its unique id as well as values for any parameters supported by the query.

A parameterized query may be stored in the server as a specialized RegistryObject called QueryDefinition object which is defined by the registry information model (see clause 4). The definition of a QueryDefinition may contain any number of Parameters supported by the query.

5.3.2.2 Invoking adhoc queries

A client may invoke a client-specific ad hoc query using a special canonical parameterized query called the AdhocQuery query defined by this specification. Due to the risks associated with un-controlled ad hoc queries, a deployment may choose to restrict the invocation of the AdhocQuery query to specific roles. This specification does not define a standard query expression syntax for ad hoc queries. A server may support any number of query expression syntaxes for ad hoc queries.

5.3.3 Query protocol

5.3.3.1 Overview

A client invokes a parameterized query using the *Query* protocol defined by the executeQuery operation of the QueryManager interface.

A client initiates the Query protocol by sending a QueryRequest message to the QueryManager endpoint.

The QueryManager sends a QueryResponse back to the client as response. The QueryResponse contains a set of objects that match the query. This is shown in Figure 12.

Figure 12 Query protocol

5.3.3.2 QueryRequest

5.3.3.2.1 Overview

Base Type: RegistryRequestType

The QueryRequest message is sent by the client to the QueryManager interface to invoke a query.

5.3.3.2.2 XML schema element definition

```
<element name="QueryRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <element name="ResponseOption" type="tns:ResponseOptionType"
            minOccurs="1" maxOccurs="1"/>
          <element name="Query" type="rim:QueryType"
            minOccurs="1" maxOccurs="1" />
        </sequence>
        <attribute name="federated" type="boolean"
          use="optional" default="false"/>
        <attribute name="federation" type="anyURI" use="optional"/>
        <attribute name="format" type="string"
          use="optional" default="application/ebrim+xml"/>
        <attribute ref="xml:lang" use="optional"/>
        <attribute name="startIndex" type="integer" default="0"/>
        <attribute name="maxResults" type="integer" default="-1"/>
        <attribute name="depth" type="integer" default="0"/>
        <attribute name="matchOlderVersions" type="boolean"
          use="optional" default="false"/>
      </extension>
    </complexContent>
  </complexType>
</element>
```

EXAMPLE: the following example shows a QueryRequest which gets an object by its id using the canonical GetObjectById query.

```
<query:QueryRequest maxResults="-1" startIndex="0" ...>
  <rs:ResponseOption returnComposedObjects="true"
returnType="LeafClassWithRepositoryItem"/>
  <query:Query queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
    <rim:Slot name="id">
      <rim:SlotValue xsi:type="StringValueType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <rim:Value>%danyal%</rim:Value>
      </rim:SlotValue>
    </rim:Slot>
  </query:Query>
```



```
</query:QueryRequest>
```

5.3.3.2.3 Description

a) Element ResponseOption - This required element allows the client to control the content of the QueryResponse generated by the server in response to this request as described in clause 5.3.3.3.
b) Element Query - This element identifies a parameterized query and supplies values for its parameters as described in clause 4.7.7.
c) Attribute depth - This optional attribute specifies the pre-fetch depth of the response desired by the client. A depth of 0 (default) indicates that the server shall return only those objects that match the query. A depth of N where N is greater that 0 indicates that the server shall also return objects that are reachable by N levels of references via attributes that reference other objects. A depth of -1 indicates that the server shall return all objects within the transitive closure of all references from objects that matches the query.
d) Attribute federated – This optional attribute specifies that the server must process this query as a federated query. By default its value is *false*. This value shall be false when a server routes a federated query to another server. This is to avoid an infinite loop in federated query processing.
e) Attribute federation - This optional attribute specifies the id of the target Federation for a federated query in case the server is a member of multiple federations. In the absence of this attribute a server must route the federated query to all registries that are a member of all federations configured within the local server. This value shall be unspecified when a server routes a federated query to another server. This is to avoid an infinite loop in federated query processing.
f) Attribute format - This optional attribute specifies the format of the response desired by the client. The default value is "application/x-ebrs+xml" which returns the response in ebRS QueryResponse format.
g) Attribute lang - This optional attribute specifies the natural language of the response desired by the client. The default value is to return the response with all available natural languages.
h) Attribute matchOlderVersions – This optional attribute specifies the behavior when multiple versions of the same object are matched by a query. When the value of this attribute is specified as *false* (the default) then a server shall only return the latest matched version for any object and shall not return older versions of such objects even though they may match the query. When the value of this attribute is specified as *true* then a server shall return all matched versions of all objects.
i) Attribute maxResults - This optional attribute specifies a limit on the maximum number of results the client wishes the query to return. If unspecified, the server should return either all the results, or in case the result set size exceeds a server specific limit, the server should return a sub-set of results that are within the bounds of the server specific limit. This attribute is described further in the Iterative Queries clause.
j) Attribute startIndex - This optional integer value is used to indicate which result must be returned as the first result when iterating over a large result set. The default value is 0, which returns the result set starting with index 0 (first result). This attribute is described further in the Iterative Queries clause.

5.3.3.2.4 Response

This request returns QueryResponse as response.

5.3.3.2.5 Exceptions

In addition to common exceptions, the following exceptions may be returned:

QueryException: signifies that the query syntax or semantics was invalid. Client must fix the query syntax or semantic error and re-submit the query

5.3.3.3 Element ResponseOption

5.3.3.3.1 Overview

A client specifies a ResponseOption structure within a QueryRequest to control the type and structure of results within the corresponding QueryResponse.

5.3.3.3.2 XML schema type definition

```
<complexType name="ResponseOptionType">
  <attribute name="returnType" default="LeafClassWithRepositoryItem">
    <simpleType>
      <restriction base="NCName">
        <enumeration value="ObjectRef"/>
        <enumeration value="RegistryObject"/>
        <enumeration value="LeafClass"/>
        <enumeration value="LeafClassWithRepositoryItem"/>
      </restriction>
    </simpleType>
  </attribute>
  <attribute name="returnComposedObjects"
    type="boolean" use="optional" default="false"/>
</complexType>
 <element name="ResponseOption" type="tns:ResponseOptionType"/>
```

5.3.3.3.3 Description:

a) Attribute returnComposedObjects - This optional attribute specifies whether the RegistryObjects returned should include composed objects as defined by **错误!未找到引用源。** in the registry information model (see clause 4). The default is to return all composed objects.
b) Attribute returnType - This optional attribute specifies the type of RegistryObject to return within the response. Values for returnType are as follows:
    1) *ObjectRef* - This option specifies that the QueryResponse shall contain a `<rim:ObjectRefList>` element. The purpose of this option is to return references to objects rather than the actual objects.
    2) *RegistryObject* - This option specifies that the QueryResponse shall contain a `<rim:RegistryObjectList>` element containing `<rim:RegistryObject>` elements with xsi:type="rim:RegistryObjectType".
    3) *LeafClass* - This option specifies that the QueryResponse shall contain a collection of `<rim:RegistryObjectList>` element containing `<rim:RegistryObject>` elements that have an `xsi:type` attribute that corresponds to leaf classes as defined in https://standards.iso.org/iso/15000/-3/ed-1/en/xsd/rim.xsd. No RepositoryItems should be included for any `rim:ExtrinsicObjectType` instance in the `<rim:RegistryObjectList>` element.
    4) *LeafClassWithRepositoryItem* - This option is the same as the LeafClass option with the additional requirement that the response include the RepositoryItems, if any, for every rim:ExtrinsicObjectType instance in the `<rim:RegistryObjectList>` element.

If "*returnType*" specified does not match a result returned by the query, then the server shall use the closest matching semantically valid returnType that matches the result.

EXAMPLE: consider a case where a Query that matches *rim:OrganizationType* instances is asked to return *LeafClassWithRepositoryItem*. As this is not possible, QueryManager will assume the LeafClass option instead.

5.3.3.4 QueryResponse

5.3.3.4.1 Overview

The QueryResponse message is sent by the QueryManager in response to a QueryRequest when the format requested by the client is the default ebrs format.

5.3.3.4.2 XML schema element definition

```
<element name="QueryResponse">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryResponseType">
        <attribute name="startIndex" type="integer" default="0"/>
        <attribute name="totalResultCount" type="integer" use="optional"/>
      </extension>
    </complexContent>
  </complexType>
</element>
```

EXAMPLE: the following shows a sample response for the example QueryRequest presented earlier.

```
<query:QueryResponse totalResultCount="1" startIndex="0" status="urn:oasis:names:tc:ebxml-
regrep:ResponseStatusType:Success">
  <rim:RegistryObjectList>
    <RegistryObject xsi:type="PersonType"
      status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
      objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Person"
      lid="urn:acme:Person:Danyal" id="urn:acme:Person:Danyal">
      <Name>
        <LocalizedString value="Danyal Najmi" xml:lang="en-US"/>
      </Name>
      <VersionInfo versionName="1"/>
      <PersonName lastName="Najmi" middleName="Idris" firstName="Danyal"/>
    </RegistryObject>
  </rim:RegistryObjectList>
</query:QueryResponse>
```

5.3.3.4.3 Description

a) Element RegistryObjectList (inherited) - This is the element that contains the RegistryObject instances that matched the specified query. A server shall provide this element in a QueryResponse even if it contains no RegistryObject instances.

b) Attribute startIndex - This optional integer value is used to indicate the index for the first result in the result set returned by the query, within the complete result set matching the query. By default, this value is 0. This attribute is described further in the Iterative Queries clause.

c) Attribute totalResultCount - This optional parameter specifies the size of the complete result set matching the query within the server. When this value is unspecified, the client should assume it is the size of the result set contained within the result. When this value is -1, the client should assume that the number of total results is unknown. In this case the client should keep iterating through the remaining result set for the query until no more results are returned. This attribute is described further in the Iterative Queries clause.

5.3.3.5 Iterative Queries

The QueryRequest and QueryResponse support the ability to iterate over a large result set matching a query by allowing multiple QueryRequest requests to be submitted in succession such that each query requests a different subset of results within the result set. This feature enables the server to handle queries that match a very large result set, in a scalable manner. The iterative query feature is accessed

via the startIndex and maxResults parameters of the QueryRequest and the startIndex and totalResultCount parameters of the QueryResponse as described earlier.

A server shall return a result set whose size is less than or equal to the maxResults parameter depending upon whether enough results are available starting at startIndex.

The iterative queries feature is not a true Cursor capability as found in databases. A server is not required to maintain transactional consistency or state between iterations of a query. Thus it is possible for new objects to be added or existing objects to be removed from the complete result set in between iterations. As a consequence it is possible to have a result set element be skipped or duplicated between iterations. However, a server shall return the same result in a deterministic manner for the same QueryRequest if no changes have been made in between the request to the server (or servers in case of federated queries).

Note that while it is not required, a server may implement a transactionally consistent iterative query feature.

5.3.4 Parameterized query definition

A parameterized query is defined by submitting a rim:QueryDefinitionType instance to the server using the submitObjects protocol. A detailed specification of the rim:QueryDefinitionType is defined in ebRIM. The definition of a parameterized query includes detailed specification of each supported parameter including its name, description, data type, cardinality and domain.

5.3.5 Canonical Query: AdhocQuery

5.3.5.1 Overview

The canonical query AdhocQuery allows clients to invoke a client-specified ad hoc query in a client-specified query expression syntax that is supported by the server. This specification does not require a server to support any specific query expression syntax. It is likely that servers may support one or more common syntaxes such as SQL-92, XQuery, XPath, SPARQL, Search-WS, OGC Filter etc.

5.3.5.2 Parameter summary

Table 48 shows the parameter summary of AdhocQuery.

Parameter	Description	Data Type	Default Value	Cardinality
queryExpression	Value is a query expression string in the language specified by the queryLanguage parameter	string		1
queryLanguage	Value is the id of a ClassificationNode within the canonical QueryLanguageScheme ClassificationScheme.	taxonomyElement		1

Table 48 AdhocQuery

5.3.5.3 Query semantics

a) The queryExpression may specify any number of named parameters
b) The server shall use rim:Slot child elements of the rim:Query as named parameters to the query queryExpression
c) The server shall return a QueryException fault message if the queryLanguage used by the queryExpression is not supported by the server
d) The server should return an AuthorizationException fault message if the client is not authorized to invoke this query
e) The server shall return the objects matching the query if the query is processed without any exceptions

5.3.6 Canonical query: BasicQuery

5.3.6.1 Overview

The canonical query BasicQuery allows clients to query for RegistryObjects by their name, description, type, status and classifications.

5.3.6.2 Parameter summary

Table 49 shows the parameter summary of BasicQuery.

Parameter	Description	Data Type	Default Value	Cardinality
classifications	Set whose elements are path attribute values to ClassificationNodes. Matches RegistryObjects that have a classification whose classificationNode attribute value matches the id of the ClassificationNode where `rim:RegistryObject[@xsi:type="rim:ClassificationNodeType"]/@path` matches specified value When multiple values are specified it implies a logical AND operation.	string		0..*
description	Matches `rim:RegistryObject/rim:Description/rim:LocalizedString/@value`	string		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
name	Matches `rim:RegistryObject/rim:Name/rim:LocalizedString/@value`	string		0..1
objectType	Matches RegistryObjects whose objectType attribute matches the id of the ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1
owner	Matches `rim:RegistryObject/@owner`. Note that a parameter value of	string		0..1

	"#@'@#rs:currentUserId()#@'@#" may be used to specify the id of the user associated with the current request			
status	Matches RegistryObjects whose status attribute matches the id of the ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1

Table 49 BasicQuery

5.3.6.3 Query Semantics

a) This query has several optional parameters
b) Each parameter implies a predicate within the underlying query
c) 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
d) If an optional parameter is not supplied then its corresponding predicate shall not be included in the underlying query

5.3.7 Canonical query: ClassificationSchemeSelector

5.3.7.1 Overview

The canonical query ClassificationSchemeSelector allows clients to create a Subscription to a remote server to replicate a remote ClassificationScheme. This query may be used as Selector query in the subscription as defined in the object replication feature.

5.3.7.2 Parameter summary

Table 50 shows the parameter summary of ClassificationSchemeSelector.

Parameter	Description	Data Type	Default Value	Cardinality
classificationSchemeId	Matches `rim:RegistryObject[@xsi:type="rim:ClassificationSchemeType"]/@id`. Does not allow wildcards.	string		1

Table 50 ClassificationSchemeSelector

5.3.7.3 Query semantics

a) The server shall return the specified ClassificationScheme and all ClassificationNodes that are descendants of that ClassificationScheme.
b) The ClassificationNodes shall not be returned as nested elements inside their parent Taxonomy element. Instead they shall be returned as sibling elements with the RegistryObjectList element of the QueryResponse.

5.3.8 Canonical query: FindAssociations

5.3.8.1 Overview

The canonical query FindAssociations query allows clients to find Associations that match the specified criteria.

5.3.8.2 Parameter summary

Table 51 shows the parameter summary of FindAssociations.

Parameter	Description	Data Type	Default Value	Cardinality
associationType	Matches Associations whose type attribute references a ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
sourceObjectId	Matches `rim:RegistryObject[@xsi:type="rim:AssociationType"]/@sourceObject`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		0..1
sourceObjectType	Matches Associations whose sourceObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	taxonomyElement		0..1
targetObjectId	Matches `rim:RegistryObject[@xsi:type="rim:AssociationType"]/@targetObject`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		0..1
targetObjectType	Matches Associations whose targetObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1

Table 51 FindAssociations

5.3.8.3 Query semantics

a) All parameters are optional
b) The server shall return the objects matching the query if the query is processed without any exceptions
c) Predicates for each supplied parameter are combined using 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

5.3.9 Canonical query: FindAssociatedObjects

5.3.9.1 Overview

The canonical query FindAssociatedObjects allows clients to find RegistryObjects that are associated with the specified RegistryObject and match the specified criteria.

5.3.9.2 Parameter summary

Table 52 shows the parameter summary of FindAssociatedObjects.

Parameter	Description	Data Type	Default Value	Cardinality
associationType	Matches associated RegistryObjects of Association's whose type attribute references a ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
sourceObjectId	Matches target RegistryObjects of Associations where the source RegistryObject's id matches `rim:RegistryObject[@xsi:type="rim:AssociationType"]/@sourceObject`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		0..1
sourceObjectType	Matches target RegistryObjects of Associations whose sourceObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1
targetObjectId	Matches source RegistryObjects of Associations where the target RegistryObject's id matches `rim:RegistryObject[@xsi:type="rim:AssociationType"]/@targetObject`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single	string		0..1

	character.			
targetObjectType	Matches source RegistryObjects of Associations whose targetObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where `rim:ClassificationNode/@path` matches specified value	taxonomyElement		0..1

Table 52 FindAssociatedObjects

5.3.9.3 Query semantics

a) All parameters are optional
b) The server shall return the objects matching the query if the query is processed without any exceptions
c) Either sourceObjectId or targetObjectId shall be specified. If neither are specified then QueryException fault shall be returned
d) Both sourceObjectId and targetObjectId shall not be specified. If both are specified then QueryException fault shall be returned
e) Predicates for each supplied parameter are combined using 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

5.3.10 Canonical query: GarbageCollector

5.3.10.1 Overview

The canonical query GarbageCollector allows clients to find RegistryObjects that are deemed as garbage by the server.

5.3.10.2 Parameter summary

This query specifies no parameters.

5.3.10.3 Query semantics

a) The server may return any objects it considers as garbage or no longer relevant or needed
b) The definition of what objects are garbage may be implementation, profile or deployment specific
c) The server shall return the following type of objects:

Dangling Associations - AssociationType instances that have an unresolvable or null sourceObject or targetObject attribute

5.3.11 Canonical query: GetAuditTrailById

5.3.11.1 Overview

The canonical query GetAuditTrailById allows clients to get the change history or audit trail for a RegistryObject whose id attribute value is the same as the value of the id parameter.

5.3.11.2 Parameter Summary

Table 53 shows the parameter summary of GetAuditTrailById.

Parameter	Description	Data Type	Default Value	Cardinality
endTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime		0..1
Id	Matches `rim:RegistryObject/@id`.	string		1
startTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime		0..1

Table 53 GetAuditTrailById

5.3.11.3 **Query semantics**

a) The server shall return a set of AuditableEvents that affected the object with id matching the specified id parameter value. The set is sorted by the timestamp attribute value in descending order (latest first)
b) If startTime is specified the server shall only include AuditableEvents whose timestamp is >= startTime parameter value
c) If endTime is specified the server shall only include AuditableEvents whose timestamp is <= endTime parameter value

5.3.12 Canonical query: GetAuditTrailByLid

5.3.12.1 **Overview**

The canonical query GetAuditTrailByLid allows clients to get the change history or audit trail for all RegistryObjects whose lid attribute value is the same as the value of the lid parameter.

5.3.12.2 **Parameter summary**

Table 54 shows the parameter summary of GetAuditTrailByLid.

Parameter	Description	Data Type	Default Value	Cardinality
endTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime		0..1
Lid	Matches `rim:RegistryObject/@lid`.	string		1
startTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime		0..1

Table 54 GetAuditTrailByLid

5.3.12.3 Query semantics

a) The server shall return a set of AuditableEvents that affected objects with lid matching the specified lid parameter value. The set is sorted by the timestamp attribute value in descending order (latest first)
b) If startTime is specified the server shall only include AuditableEvents whose timestamp is >= startTime parameter value
c) If endTime is specified the server shall only include AuditableEvents whose timestamp is <= endTime parameter value

5.3.13 Canonical query: GetAuditTrailByTimeInterval

5.3.13.1 Overview

The canonical query GetAuditTrailByTimeInterval allows clients to get *all* changes to *all* objects in the server within a specified time interval. This query may be used to keep a client periodically synchronized with changes in the server.

5.3.13.2 Parameter summary

Table 55 shows the parameter summary of GetAuditTrailByTimeInterval.

Parameter	Description	Data Type	Default Value	Cardinality
endTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime	5 minutes before current time	0..1
startTime	Specifies the end of the time interval (inclusive) for `rim:RegistryObject[@xsi:type="rim:AuditableEventType"]/@timestamp` value	dateTime	Current time	0..1

Table 55 GetAuditTrailByTimeInterval

5.3.13.3 Query semantics

a) The server shall return a set of AuditableEvents whose timestamp attribute is within the time interval specified by startTime and endTime parameters. The set is sorted by the timestamp attribute value in descending order (latest first)
b) The server shall only include AuditableEvents whose timestamp is >= startTime parameter value
c) The server shall only include AuditableEvents whose timestamp is <= endTime parameter value

5.3.14 Canonical query: GetChildrenByParentId

5.3.14.1 Overview

The canonical query GetChildrenByParentId allows clients to get the children of a RegistryObject whose Id attribute value is the same as the value specified for the parentId parameter. This query is used to query objects hierarchies with parent-child relationships such as the following:

a) ClassificationScheme – Child ClassificationNodes

b) Organization – Child Organizations
c) RegistryPackage – RegistryPackage Members

5.3.14.2 **Parameter summary**

Table 56 shows the parameter summary of GetChildrenByParentId.

Parameter	Description	Data Type	Default Value	Cardinality
Depth	Specifies how many levels of descendants to fetch: a) depth > 0 implies get descendants upto "depth" levels b) depth <= 0 implies get all descendants	integer	1	0..1
exclusiveChildrenOnly	Specifies how to handle children that may have multiple parents: a) True value specifies that only children that are not children of any other parent should be returned b) false value specifies that children that have other parents should also be matched	boolean	false	0..1
objectType	Specifies the type of object hierarchy for the query	string		0..1
parentId	Specifies the id of the parent object	string		0..1

Table 56 GetChildrenByParentId

5.3.14.3 **Query semantics**

a) If objectType and parentId are both unspecified the server shall return all RegistryObjects that are not members of a RegistryPackage (root level objects)
b) If parentId parameter is unspecified and objectType parameter is specified the server shall return all root level objects for the object hierarchy identified by the objectType as follows:
   1) If objectType parameter value contains the string "ClassificationScheme" the server shall return all ClassificationSchemes
   2) If objectType parameter value contains the string "Organization" the server shall return all Organizations that are not a member of another Organization (root level Organizations)
   3) If objectType parameter value contains the string "RegistryPackage" the server shall return all RegistryPackages that are not a member of another RegistryPackage (root level RegistryPackages)
c) If parentId parameter is specified then the behavior is as follows:
   1) If objectType parameter value is unspecified or if its value contains the string "RegistryPackage" the server shall return all RegistryObjects that are member of a RegistryPackage whose id is the same as the value of the parentId attribute
   2) If objectType parameter is specified and its value contains the string "ClassificationScheme" the server shall return all ClassificationNodes that are children of a TaxonomyElementType instance whose id is the same as the value of the parentId attribute
   3) If objectType parameter is specified and its value contains the string "Organization" the server shall return all Organizations that are members of an Organization whose id is the same as the value of the parentId attribute

d) If depth parameter is specified then the server shall also return all descendants upto the specified depth as described by the definition of the depth parameter above
e) If exclusiveChildrenOnly is specified with a true value then the server shall not return any descendants that have multiple parents

5.3.15 Canonical query: GetClassificationSchemesById

5.3.15.1 Overview

The canonical query GetClassificationSchemesById allows clients to fetch specified ClassificationSchemes.

5.3.15.2 Parameter summary

Table 57 shows the parameter summary of GetClassificationSchemesById.

Parameter	Description	Data Type	Default Value	Cardinality
id	Matches `rim:RegistryObject[@xsi:type="rim:ClassificationSchemeType"]/@id`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		0..1

Table 57 GetClassificationSchemesById

5.3.15.3 Query Semantics

a) The server shall return the objects matching the query if the query is processed without any exceptions
b) The depth parameter of the QueryRequest may be used to pre-fetch the ClassificationNodes of matches ClassificationSchemes

5.3.16 Canonical query: GetRegistryPackagesByMemberId

5.3.16.1 Overview

The canonical query GetRegistryPackagesByMemberId allows clients to get the RegistryPackages that a specified RegistryObject is a member of.

5.3.16.2 Parameter summary

Table 58 shows the parameter summary of GetRegistryPackagesByMemberId.

Parameter	Description	Data Type	Default Value	Cardinality
memberId	Matches RegistryPackages that have a RegistryObject as an immediate member where the RegistryObject's id rim:RegistryObject/@id matches the specified value.	string		0..1

	Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.			

Table 58 GetRegistryPackagesByMemberId

5.3.16.3 **Query semantics**

The server shall return the objects matching the query if the query is processed without any exceptions

5.3.17 Canonical query: GetNotification

5.3.17.1 **Overview**

The canonical query GetNotification allows clients to "pull" any pending Notification for a Subscription at a time of their choosing. This is defined in detail under clause titled "Pulling Notification on Demand".

5.3.17.2 **Parameter summary**

Table 59 shows the parameter summary of GetNotification.

Parameter	Description	Data Type	Default Value	Cardinality
subscriptionId	Matches `rim:RegistryObject[@xsi:type="rim:SubscriptionType"]/@id`. Wildcards are not allowed.	string		1
startTime	The time since which events should be included in the Notification	xs:dateTime		0..1

Table 59 GetNotification

5.3.17.3 **Query semantics**

a) The server shall return a Notification with events that affected objects matching the query selector query for the Subscription.
b) The server shall return only those events that have a timestamp later than startTime.

5.3.18 Canonical query: GetObjectById

5.3.18.1 **Overview**

The canonical query GetObjectById allows clients to find RegistryObjects based upon the value of their id attribute.

5.3.18.2 **Parameter summary**

Table 60 shows the parameter summary of GetObjectById.

Parameter	**Description**	**Data**	**Default**	**Cardinality**

		Type	Value	
id	Matches `rim:RegistryObject/@id`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		1

Table 60 GetObjectById

5.3.18.3 Query semantics

The server shall return the RegistryObjects whose id attribute value matches the specified value of the id parameter.

5.3.19 Canonical query: GetObjectsByLid

5.3.19.1 Overview

The canonical query GetObjectByLid allows clients to find RegistryObjects based upon the value of their lid attribute. It is used to fetch all versions of a logical object without any specific order or relationship among them.

5.3.19.2 Parameter summary

Table 61 shows the parameter summary of GetObjectsByLid.

Parameter	Description	Data Type	Default Value	Cardinality
lid	Matches `rim:RegistryObject/@lid`. Allows use of "%" wildcard character to match multiple characters. Allows use of "?" wildcard character to match a single character.	string		1

Table 61 GetObjectsByLid

5.3.19.3 Query semantics

The server shall return all RegistryObjects whose lid attribute value matches the specified value of the lid parameter.

5.3.20 Canonical query: GetReferencedObject

5.3.20.1 Overview

The canonical query GetReferencedObject allows clients to get a RegistryObject that is the target of an rim:objectReferenceType attribute value.

5.3.20.2 **Parameter summary**

Table 62 shows the parameter summary of GetReferencedObject.

Parameter	Description	Data Type	Default Value	Cardinality
objectReference	Contains the value for a rim:objectReferenceType attribute	string		0..1

Table 62 GetReferencedObject

5.3.20.3 **Query semantics**

The server shall return the RegistryObjectType instance that is being referenced by the specified value for the objectReference parameter.

a) If the objectReference contains the id of a local object that is not a DynamicObjectRef instance then the server shall return that object.
b) If the objectReference contains the id of a local DynamicObjectRef instance then the server shall invoke the Query within the DynamicObjectRef instance and resolve the reference to the singleton result of the Query and return the matching object.
c) If the objectReference contains the canonical URL for a remote object then the server shall invoke the GetReferencedObject query against the remote server using the id of the remote object as the value of the objectReference parameter and return the matching object. The id of the remote object is accessible from its canonical URL as the value of the id parameter within the URL.

5.3.21 Canonical query: KeywordSearch

5.3.21.1 **Overview**

The canonical query KeyWordSearch allows clients to find RegistryObjects and RepositoryItems that contain text that matches keywords identified by specified search patterns.

5.3.21.2 **Canonical indexes**

This query defines a set of canonical index names as defined by Table 63. Each index name is associated with a particular type of information that it indexes. A server shall index all information that is defined by the canonical indexes below. A server may define additional indexes to index information not specified by this clause.

Index Name	Description
name.localizedString.value	Indexes the value of all localized string in all Name elements of all RegistryObjects
description.localizedString.value	Indexes the value of all localized string in all Description elements of all RegistryObjects
slot.name	Indexes the name of all slots on all RegistryObjects
slot.value	Indexes the value of all slots on all RegistryObjects
repositoryItem	Indexes the text of all text based repository items associated with ExtrinsicObjects

personName.firstName	Indexes the firstName attribute of PersonName elements in all Person objects
personName.middleName	Indexes the middleName attribute of PersonName elements in all Person objects
personName.lastName	Indexes the lastName attribute of PersonName elements in all Person objects
emailAddress.address	Indexes the address attribute of all EmailAddress objects
postalAddress.city	Indexes the city attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.country	Indexes the country attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.postalCode	Indexes the postalCode attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.stateOrProvince	Indexes the stateOrProvince attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.street	Indexes the street attribute of all PostalAddress elements contained within any RegistryObject

Table 63 Canonical indexes

5.3.21.3 **Parameter summary**

Parameter	Description	Data Type	Default Value	Cardinality
keywords	A space separated list of keywords to search for	string		1

Table 64 KeywordSearch

5.3.21.4 **Query semantics**

The value of the keywords parameter may consist of multiple terms where each term is separated by one or more spaces

EXAMPLE 1: ebxml regrep

Semantics: Matches objects containing either “ebxml” or “regrep”.

A term may be enclosed in double-quotes to include white space characters as a literal value.

EXAMPLE 2: “ebxml regrep”

Semantics: Matches objects containing “ebxml regrep”

Terms may be specified using wildcard characters where '*' matches one or more characters and “?” matches a single character.

Terms may be combined using boolean operators “AND”, “OR” and “not”. Absence of a boolean operator between terms implies an implicit OR operator between them.

EXAMPLE 3: ebxml AND regrep

Semantics: Matches objects containing “ebxml” and “regrep”

EXAMPLE 4: ebxml not regrep

Semantics: Matches objects containing “ebxml” and not containg “regrep”

EXAMPLE 5: ebxml OR regrep

Semantics: Matches objects containing “ebxml” or “regrep”

EXAMPLE 6: ebxml regrep

Semantics: Matches objects containing “ebxml” or “regrep”

Terms may be grouped together using “(“ at the beginning and “)” at the end of the group. Grouping allowing boolean operators to be applied to a group of terms as a whole and enables more flexible searches.

EXAMPLE 7: ebxml AND (registry OR regrep)

Semantics: Matches objects containing both “ebxml” and either “registry” or “regrep”

The server shall return all RegistryObjects that contain indexed data matching the semantics of the keywords parameter.

The server shall return all ExtrinsicObjects that have a repository item that contains indexed data matching the semantics of the keywords parameter.

5.3.22 Canonical query: RegistryPackageSelector

5.3.22.1 Overview

The canonical query RegistryPackageSelector allows clients to create a Subscription to a remote server to replicate a remote RegistryPackage as well as all its member objects and the AssociationType instances that relate the members of the RegistryPackage to it. This query may be used as Selector query within the Subscription for the replication as defined in the object replication feature.

5.3.22.2 Parameter summary

Table 65 shows the parameter summary of RegistryPackageSelector.

Parameter	Description	Data Type	Default Value	Cardinality
registryPackageIds	A set of IDs of rim:RegistryPackageType instances. Does not allow wildcards.	string		1..*

Table 65 RegistryPackageSelector

5.3.22.3 Query semantics

a) The server shall return the specified RegistryPackageType instance, all RegistryObjectType instances that are members of the specified RegistryPackage as well as all "HasMember" AssociationType instances between the RegistryPackageType instance and its members, that are descendants of that ClassificationScheme.
b) The member RegistryObjectType instances shall not be returned as nested elements inside the RegistryPackage. Instead they shall be returned as sibling elements with the RegistryPackage and Associations within the RegistryObjectList element of the QueryResponse.

5.3.23 Query functions

5.3.23.1 Overview

A server may support any number of functions known as *Query Functions*, that may be used within a query expression or query parameter. Query functions are similar in concept to functions in SQL. Query functions may be used within the query expression of a parameterized query as well as within its invocation parameter values. Query functions enable parameterized queries to use specialized search algorithms to augment their capabilities.

This specification defines a number of canonical functions that are standard functions that shall be supported by a server. Profiles, implementations and deployments may define additional query functions beyond the canonical functions defined by this specification.

5.3.23.2 Using functions in query expressions

A parameterized query stored as a rim:QueryDefinition instance may have a rim:QueryExpression which defines a query expression within its sub-nodes. A client may submit a rim:QueryDefinition such that its query expression may use any number of query functions supported by the server any where within the query expression where it is syntactically correct to use the value returned by the function.

If a query expression contains one or more function invocations then the query expression shall delimit the parts of the query expression that are not a function invocation with the leading characters "#@" and trailing characters "@#". This is similar in syntax to a Java multi-line comment syntax where a comment is delimited by leading characters "/*" and trailing characters "*/". The delimiters serve the following purposes:

a) Allows a parser to recognize the non-function parts of the query expression that shall be preserved *as is*
b) Allows implementations to be optimized to skip function parsing and evaluation if the special delimiter characters are not present in query expression

EXAMPLE: a SQL query expression which uses the getClassificationNodes function to match all RegistryObjects that are targets of Association with specified sourceObject and type that is a subnode of AffiliatedWith node upto a depth of 2 levels in the descendant hierarchy. The delimiter characters are in bold font while the function invocations is in bold and italic font below:

```
--example of a query expression with query functions
#@SELECT targetObject.* FROM
RegistryObjectType targetObject, AssociationType a WHERE

  a.sourceObject = :sourceObject AND
  a.type IN (@# getClassificationNodes("urn:oasis:names:tc:ebxml-
regrep:AssociationType:AffiliatedWith", 0, 2, "false", ",", "${id}") #@) AND
  targetObject.id = a.targetObject@#
```


5.3.23.3 Using functions in query parameters

A client may use query functions supported by a server within parameter values specified when invoking a parameterized query. A client may invoke a parameterized query using the Query protocol such that its query parameter values may use any number of query functions supported by the server any where within the query parameter where it is syntactically correct to use the value returned by the function.

If a query parameter value contains one or more function invocations then the query expression shall delimit the parts of the query parameter that are not a function invocation with the leading characters "#@" and trailing characters "@#". If a query parameter value only has function invocations and contains no non-function parts then it must include at least one leading or trailing "#@@#" delimiter token pair to allow optimized parsing and evaluation of query functions only when needed.

EXAMPLE 1: the following is an example of a query expression that has no query functions. Its two parameters are shown in bold font:

```
--Following is the query expression within the server
--This time it has no query functions as they are in the query parameters
SELECT targetObject.* FROM
RegistryObjectType targetObject, AssociationType a WHERE

  a.sourceObject = :sourceObject AND
  a.type IN ( :types ) AND
  targetObject.id = a.targetObject
```

EXAMPLE 2: the following is an example of invocation of a parameterized query that uses the above query expression and uses the getClassificationNodes function from previous example within the value of the *types* parameter. Note the trailing "#@@#" delimiter tokens are present as required.

```
<query:QueryRequest maxResults="-1" startIndex="0" ...>
  <rs:ResponseOption returnComposedObjects="true"
returnType="LeafClassWithRepositoryItem"/>
  <query:Query queryDefinition="urn:acme:ExampleQuery">
    <rim:Slot name="sourceObject">
      <rim:SlotValue xsi:type="StringValueType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <rim:Value>urn:test:Person:Danyal</rim:Value>
      </rim:SlotValue>
    <rim:Slot name="types">
      <rim:SlotValue xsi:type="StringValueType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <rim:Value>getClassificationNodes("urn:oasis:names:tc:ebxml-
regrep:AssociationType:AffiliatedWith", 0, 2, "false", ",", "${id}")#@@#</rim:Value>
      </rim:SlotValue>
    </rim:Slot>
  </query:Query>
</query:QueryRequest>
```

5.3.23.4 Function processing model

A server shall meet the following function processing requirements during the processing of a QueryRequest:

a) When processing a query expression elements (rim:QueryDefinition/rim:QueryExpression) the server should not perform function processing if the special delimiter sequences of "#@" and "@#" are not found in the query expression
b) When processing query invocation parameter elements (query:QueryRequest/query:Query/rim:Slot/rim:SlotValue) the server should not perform

function processing if the special delimiter sequences of “#@” and “@#” are not found in the query expression

c) When processing a query expression element if the special delimiter sequences of “#@” and “@#” are found then the server shall process query expression elements to replace all function invocations with the value returned when the function is invoked with specified parameters

d) When processing query invocation parameter elements if the special delimiter sequences of “#@” and “@#” are found then the server shall process each query parameter element to replace all function invocations with the value returned when the function is invoked with specified parameters

e) When invoking a function that has another function invocation as its parameter the inner most functions shall be invoked first so that the outer function can be invoked with the value returned by the inner function invocation

f) When processing a query expression or query parameter the special delimiter characters “#@” and “@#” shall be removed and the value contained within them shall be preserved without any change

5.3.23.5 **Function processor BNF**

The following BNF grammar normatively describes the grammar for query expressions and query invocation parameters with embedded function invocations. The ***start*** production describes the grammar for query expressions and query invocation parameters with embedded function invocations.

```
<DEFAULT> SKIP : {
" "
| "\t"
| "\r"
| "\n"
}
```

```
<DEFAULT> TOKEN : {
<FLOAT: <INTEGER> "." <INTEGER> | "." <INTEGER> | <INTEGER> ".">
| <INTEGER: (<DIGIT>)+>
| <DIGIT: ["0"-"9"]>
| <BOOLEAN: "true" | "false">
}
```

```
<DEFAULT> TOKEN : {
<S_IDENTIFIER: (<LETTER>)+ (<DIGIT> | <LETTER> | <SPECIAL_CHARS>)*>
| <#LETTER: ["a"-"z","A"-"Z"]>
| <#SPECIAL_CHARS: "_">
| <S_CHAR_LITERAL: "\'" (~["\'"])* "\'" ("\'" (~["\'"])* "\'")*>
| <S_QUOTED_IDENTIFIER: "\"" (~["\n","\r","\""])* "\"">
| <OPENPAREN: "(">
| <CLOSEPAREN: ")">
| <COMMA: ",">
| <COLON: ":">
| <DELIMITED_TEXT: "#@" (~["@"])* "@#">
}

start   ::=   ( textOrFunctionCall  )+ <EOF>
text  ::=   ( ( <DELIMITED_TEXT> ) )
textOrFunctionCall    ::=   ( text | FunctionCall )
FunctionCall  ::=  FunctionReference <OPENPAREN> ( FunctionArgumentList )* <CLOSEPAREN>
FunctionReference  ::=  <S_IDENTIFIER> <COLON> <S_IDENTIFIER>
FunctionArgumentList  ::=  FunctionArgument ( <COMMA> FunctionArgument )*
FunctionArgument   ::=   ( FunctionCall | <S_CHAR_LITERAL> | <S_QUOTED_IDENTIFIER> |
<FLOAT> | <INTEGER> | <BOOLEAN> )
```


5.3.24 Common patterns in query functions

5.3.24.1 Overview

This clause defines some commonly occurring patterns in query functions and defines some common solutions to addressing these patterns. Profiles should conform to the solutions defined in this clause whenever possible.

5.3.24.2 Specifying a null value for string param or return value

A function that accepts a string parameter should treat a value of “rs:null” as a null string. A null string is a string whose value is unspecified.

When a function returns a “string” type it should return a null value string as the canonical value “rs:null”.

5.3.25 Canonical functions

5.3.25.1 Overview

This clause defines a set of standard canonical functions that shall be supported by all servers. A client may use these functions within a query expression or within the value of a parameter to a parameterized query. A server shall process the functions according to their behavior as specified in this clause. The function processing model is specified in Function Processing Model.

A client shall use the “rs:” namespace prefix when using a canonical function defined by this profile. Profiles of this specification may define their own canonical functions as well as a standard namespace prefix to be used with these functions.

A client shall specify the parameters of a function in the same order as specified in the table for the function specification.

Table 66 summarizes the canonical functions defined by this specification.

Function Name	Semantics
currentTime	Returns the current time in ISO 8601 format
currentUserId	Returns the id of the user associated with the current RegistryRequest
relativeTime	Returns a time in the future or past, relative to the current time where the offset period is determined by specified parameter
getClassificationNodes	Returns all ClassificationNode's that are descendants and / or ancestor of the specified reference ClassificationNode and within the specified number of levels as indicated by the ancestorLevels and descendantLevels parameters.

Table 66 Canonical functions defined by this specification

5.3.25.2 Canonical function: currentTime

This canonical function takes no parameters and returns the current time associated with the server.

5.3.25.2.1 Function semantics

The server shall return a string if the query is processed without any exceptions

The value of the string shall be current time in ISO 8601 format using the UTC time zone.

EXAMPLE: an example of value returned is “2010-02-25T15:22:14.534Z”.

5.3.25.3 Canonical function: currentUserId

This canonical function takes no parameters and returns a string whose value is the id of the user associated with the current RegistryRequest. This specification does not define how user's are managed within the server nor does it define how an id is assigned to a user.

5.3.25.3.1 Function semantics

The server shall return a string if the query is processed without any exceptions

The value of the string shall be “rs:null” if no current user is associated with the RegistryRequest

5.3.25.4 Canonical function: relativeTime

This canonical function takes a string parameter in the format specified by xs:duration that specify a time offset period and returns a time in the future or past relative to the current time by the specified period.

5.3.25.4.1 Parameter summary

Table 67 shows the parameter summary of relativeTime.

Parameter	Description	Data Type
duration	A duration of time in the format as specified by the duration type defined by XML Schema duration type. The duration format supports negative or positive durations so this function may be used to return a time relative to current in the future or the past.	duration

Table 67 relativeTime

5.3.25.4.2 Function semantics

a) The server shall return a string if the query is processed without any exceptions
b) The format of the duration parameter shall conform to the format as specified by the duration type defined by XML Schema duration type otherwise the server shall return InvalidRequestException
c) The value of the string shall be a time in ISO 8601 format that is offset by the specified period in the future relative to the current time.

EXAMPLE: an example of value returned is “2010-02-25T15:22:14.534Z”

5.3.25.5 Canonical function: getClassificationNodes

This canonical function takes a reference ClassificationNode's id as parameter and returns all ClassificationNode's that are descendants and/or ancestors of the specified reference ClassificationNode and within the specified number of levels as indicated by the ancestorLevels and descendantLevels parameters.

5.3.25.5.1 Parameter summary

Table 68 shows the parameter summary of getClassificationNodes.

Parameter	Description	Data Type
nodeId	Specifies the id of the reference ClassificationNodeType instance	string
ancestorLevels	Specifies how many levels to match ancestors of reference node	integer
descendantLevels	Specifies how many levels to match descendants of reference node	integer
includeSelf	Specifies whether to include the reference ClassificationNodeType instance or not	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string
template	The value of this parameter specifies a template to contain each id returned by the function. The template may contain one or more occurrences of template parameter string "${id}" as placeholder for the id of a matched ClassificationNode	string

Table 68 getClassificationNodes

5.3.25.5.2 Function semantics

a) The server shall return a string if the query is processed without any exceptions
b) The string shall be "rs:null" if no ClassificationNode is found that matches the function parameters
c) The string shall consist of a set of substrings separated by the appropriate delimiter character when any ClassificationNode's are found that match the function parameters:
   1) There shall be a substring for each ClassificationNode matched by the function
   2) Each substring shall conform to the specified template such that all occurrences of ${id} are replaced by the id of a ClassificationNode matched by the function
d) The id of the reference ClassificationNode shall be included if and only if the includeSelf parameter value is true
e) A ancestorLevels value of N where N > 0 matches all ClassificationNodes upto the Nth level ancestors of the reference ClassificationNode. A value of 1 matches the immediate parents of the reference ClassificationNode while a value of 2 matches the parents and grandparents of the

reference ClassificationNode. A value of -1 matches all ancestors of the reference ClassificationNode

f) A descendantsLevels value of N where N > 0 matches all ClassificationNodes upto the Nth level descendants of the reference ClassificationNode. A value of 1 matches the immediate children of the reference ClassificationNode while a value of 2 matches the children and grandchildren of the reference ClassificationNode. A value of -1 matches all descendants of the reference ClassificationNode

g) A template value of "rs:null" is implicitly equivalent to a template value of "${id}"

5.3.26 Query plugins

5.3.26.1 Overview

Query plugins allow a server to use specialized extension modules to implement support for a parameterized query. Since query plugins are software modules, they are able to handle highly specialized query semantics that may not be expressed in most query languages. A specific instance of a query plugin is designed and configured to handle a specific parameterized query.

5.3.26.2 Query plugin interface

A Query plugin implements the QueryManager interface. A QueryManager endpoint shall delegate an executeQuery operation to a Query plugin if a Query plugin has been configured for the requested parameterized query. A Query plugin shall process the query and return a QueryResponse or fault message to the QueryManager. The QueryManager shall then deliver that response to the client.

5.4 LifecycleManager interface

5.4.1 Overview

The LifecycleManager interface allows a client to perform various lifecycle management operations on RegistryObjects. These operations include submitting RegistryObjects to the server, updating RegistryObjects in the server, creating new versions of RegistryObjects in the server and removing RegistryObjects from the server.

A server shall implement the LifecycleManager interface as an endpoint.

5.4.2 SubmitObjects protocol

5.4.2.1 Overview

The SubmitObjects protocol allows a client to submit RegistryObjects to the server. It also allows a client to completely replace existing RegistryObjects in the server.

A client initiates the SubmitObjects protocol by sending a SubmitObjectsRequest message to the LifecycleManager endpoint.

The LifecycleManager sends a RegistryResponse back to the client as response. This is shown in Figure 13.

Figure 13 SubmitObjects protocol

5.4.2.2 SubmitObjectsRequest

The SubmitObjectsRequest message is sent by a client to submit RegistryObjects to the server.

5.4.2.2.1 XML schema type and element definition

```
<simpleType name="mode">
  <restriction base="NCName">
    <enumeration value="CreateOrReplace"/>
    <enumeration value="CreateOrVersion"/>
    <enumeration value="CreateOnly"/>
  </restriction>
</simpleType>

<element name="SubmitObjectsRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <element ref="rim:RegistryObjectList" minOccurs="0" maxOccurs="1"/>
        </sequence>
        <attribute name="checkReferences" type="boolean" use="optional"
          default="false"/>
        <attribute name="mode" type="tns:mode" use="optional"
          default="CreateOrReplace"/>
      </extension>
    </complexContent>
  </complexType>
</element>
```

5.4.2.2.2 Description

a) Element RegistryObjectList - Specifies a set of RegistryObject instances that are being submitted to the server. The RegistryObjects in the list may be new objects being submitted to the server or they may be current objects already existing in the server.

a) Attribute checkReferences – Specifies the reference checking behavior expected of the server

   1) true - Specifies that a server shall check submitted objects and make sure that all references via reference attributes and slots to other RegistryObjects are resolvable. If a reference does not resolve then the server shall return UnresolvedReferenceException

   2) false (default) – Specifies that a server shall not check submitted objects to make sure that all references via reference attributes and slots to other RegistryObjects are resolvable. If a reference does not resolve then the server shall not return UnresolvedReferenceException

b) Attribute mode – Specifies the semantics for how the server should handle RegistryObjects being submitted when they already exist in the server:

   1) CreateOrReplace (default) - If an object does not exist, server shall create it as a new object. If an object already exists, server shall replace the existing object with the submitted object

2) CreateOrVersion - If an object does not exist, server shall create it as a new object. If an object already exists, server shall not alter the existing object and instead it shall create a new version of the existing object using the state of the submitted object

3) CreateOnly - If an object does not exist, server shall create it as a new object. If an object already exists, the server shall return an ObjectExistsException fault message

5.4.2.2.3 id and lid requirements

Table 69 defines the requirements for id and lid attribute values for RegistryObjectType instances that are submitted via the SubmitObjects protocol.

Mode / Requirements	ID Requirements	LID Requirements
CreateOrReplace	a) shall be specified by client or else server shall return InvalidRequestException b) If id does not exists, server shall create new object using that id (create) If id exists, server shall replace existing object matching that id (update)	shall be specified by client or else server shall return InvalidRequestException
CreateOrVersion	a) shall be specified by client or else server shall return InvalidRequestException b) If id does not exists and lid does not exist, server shall create new object using that id (create) c) If id does not exists and lid exists, server shall throw InvalidRequestException (otherwise multiple root level versions would become possible) d) If id exists, server shall create a new version of existing object matching that id (version)	shall be specified by client or else server shall return InvalidRequestException
CreateOnly	a) may be specified by client b) If unspecified Server shall generate UUID URN c) If id does not exists, server shall create new object using that id (create) d) If id exists, server shall return ObjectExistsException	a) shall be specified by client or else server shall return InvalidRequestException b) shall not exist or else server shall return ObjectExistsException

Table 69 Requirements for id and lid during SubmitObjects protocol

5.4.2.2.4 Returns

This request returns a RegistryResponse.

5.4.2.2.5 Exceptions

A server shall return an UnsupportedCapabilityException fault message if the request contains a type that is an extension of types defined by ebRIM and if the server cannot support such extension.

5.4.2.3 Audit trail requirements

a) The server shall create a single AuditableEvent object as follows:
   1) If RegistryObjects were created by the request, it contain a single Action sub-element with eventType *Created* for all the RegistryObjects created during processing of the request
   2) If RegistryObjects were updated by the request, it contain a single Action sub-element with eventType *Updated* for all the RegistryObjects updated during processing of the request
b) The server should create AuditableEvents *after* successfully processing the request in a separate transaction from the request

5.4.2.4 Sample SubmitObjectsRequest

EXAMPLE: the following simplified example shows a SubmitObjectsRequest that submits a single Organization object to the server.

```
<lcm:SubmitObjectsRequest>
  <rim:RegistryObjectList>
    <rim:RegistryObject xsi:type="rim:OrganizationType" lid="${LOGICAL_ID}"
        id="${ID}" ...>

     ...

    </rim:RegistryObject>
  </rim:RegistryObjectList>
</SubmitObjectsRequest>
```

5.4.3 UpdateObjects protocol

5.4.3.1 Overview

The UpdateObjectsRequest protocol allows a client to make partial updates to one or more RegistryObjects that already exist in the server. This protocol enables *partial* update of RegistryObjects rather than a *complete replacement*. A client should use the SubmitObjects protocol for complete replacement of RegistryObjects.

A server shall return InvalidRequestException fault message if the client attempts to update the id, lid or objectType attribute of a RegistryObject. This is shown in Figure 14.

Figure 14 UpdateObjects protocol

5.4.3.2 UpdateObjectsRequest

The UpdateObjectsRequest message is sent by a client to partially update existing RegistryObjects in the server. An UpdateObjectsRequest identifies a set of RegistryObjects as target objects to be updated by the request. It also specifies the update action that modifies each target object. Update actions may insert a node within a target object, delete an existing node from a target object or update an existing node within the target object. A node in the context of the UpdateObjects protocol is defined to be an XML DOM node (typically an element or an attribute).

5.4.3.2.1 XML schema element definition

```
<element name="UpdateObjectsRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <!-- Query and ObjectRefList select objects to update -->
          <element name="Query" type="rim:QueryType" minOccurs="0" maxOccurs="1" />
          <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />

          <!-- Specifies how to update selected objects -->
          <element name="UpdateAction" type="tns:UpdateActionType"
           minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="checkReferences" type="boolean" use="optional"
          default="false"/>
        <attribute name="mode" type="tns:mode" use="optional"
          default="CreateOrReplace"/>
      </extension>
    </complexContent>
  </complexType>
</element>
```

5.4.3.2.2 Description

a) Element Query - Specifies a query to be invoked. A server shall use all objects that match the specified query in addition to any other objects identified by the ObjectRefList element as targets of the update action.
b) Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances in the server. A server shall use all objects that are referenced by this element in addition to any other objects identified by the Query element as targets of the update action.
c) Element UpdateAction – Specifies the details of how to update the target objects
d) Attribute checkReferences – Specifies the reference checking behavior expected of the server
    1) true - Specifies that a server shall check updated objects and make sure that all references via reference attributes and slots to other RegistryObjects are resolvable. If a reference does not resolve then the server shall return UnresolvedReferenceException
    2) false (default) – Specifies that a server shall not check updated objects to make sure that all references via reference attributes and slots to other RegistryObjects are resolvable. If a reference does not resolve then the server shall not return UnresolvedReferenceException

e) Attribute mode – Specifies the semantics for how the server should handle RegistryObjects being updated in the server:
   1) CreateOrReplace (default) - If an object does not exist, server shall return ObjectNotFoundException. If an object already exists, server shall update the existing object without creating a new version
   2) CreateOrVersion - If an object does not exist, server shall return ObjectNotFoundException. If an object already exists, server shall create a new version of the existing object before applying the requested update action
   3) CreateOnly – This mode does not apply to UpdateObjectsRequest. If specified, server shall return an InvalidRequestException

5.4.3.2.3 Returns

This request returns a RegistryResponse.

5.4.3.2.4 Exceptions

A server shall return an UnsupportedCapabilityException fault message if the request contains a type that is an extension of types defined by ebRIM and if the server cannot support such extension.

5.4.3.3 UpdateAction

An UpdateRequest contains one or more UpdateActions. Each UpdateObjectsRequest defines a specific update action to be performed on each target object.

5.4.3.3.1 XML schema type definition

```
<complexType name="UpdateActionType">
  <annotation>
    <documentation xml:lang="en">
    </documentation>
  </annotation>
  <sequence>
    <!-- Value for attribute or element -->
    <element name="ValueHolder" type="rim:ValueType"
      minOccurs="0" maxOccurs="1"/>
    <!--
    Value of selector is an XPATH expression that uniquely identifies
    an attribute or an element within target documents.
    -->
    <element name="Selector" type="rim:QueryExpressionType"
      minOccurs="1" maxOccurs="1"/>
  </sequence>

  <!--
  Specifies whether to insert, update or delete a node from
  target document.
  -->
  <attribute name="mode" use="required">
    <simpleType>
      <restriction base="NCName">
        <enumeration value="Insert"/>
        <enumeration value="Update"/>
        <enumeration value="Delete"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>
```


5.4.3.3.2 Description

a) Element Selector – Is a QueryExpressionType that contains the expression that identifies a node of the resource representation to be updated.

The value of this element shall conform to the queryLanguage specified in the queryLanguage attribute of the Selector. A resource shall generate an QueryException fault if the expression is invalid. If the expression syntax is not valid with respect to the queryLanguage then a resource should specify a fault detail of "InvalidExpressionSyntaxException". If the expression value is not valid for the resource type then the resource should specify a fault detail of "InvalidExpressionValueException".

A server shall minimally support XPATH 1.0 as the queryLanguage for Selector element. The scope of the XML document that is processed by the XPATH expression is the RegistryObjectType instance. A server shall implicitly support the standard namespace prefixes used by RegRep schemas (rim:, query:, rs:, lcm:, spi:) as a notational convenience. These standard namespace prefixes should map to the latest version of the specification supported by the server.

An XPATH selector expression shall be specified using the RegistryObject being updated as the context node.

An XPATH selector expression may select an attribute or an element relative to the RegistryObject context node. If it selects an attribute then the ValueHolder element should use a ValueType subtype for a primitive type (instead of AnyValueType) that corresponds to the primitive type for the attribute (e.g. StringValueType). The ValueHolder/Value element's content shall contain the attribute value.

b) Element ValueHolder - This element contains the value to be written to the target object. If the mode attribute is "Insert" or "Update" then this element shall be present. If the mode is "Delete" then this element shall not be present.

c) Attribute mode – This attribute specifies the semantics for how the server should update target objects:
   1) Insert - Indicates that the value provided by ValueHolder shall be added to the target object. If the selector targets a repeated element (maxOccurs > 1), the node shall be added at the end. If the selector targets a non-repeated element (maxOccurs = 1) that already exists, the resource shall generate an InvalidRequestException with a fault detail of NodeAlreadyExistsException. If the selector targets an existing item of a repeated element, the value provided by ValueHolder shall be added before the existing item.
   2) Update – Indicates that the node identified by selector shall be replaced by value by the ValueHolder in its place. If the selector resolves to nothing then there should be no change to the target object.
   3) Delete - indicates that the node identified by selector shall be deleted from the target object if it is present.

5.4.3.4 Audit Trail Requirements

a) The server shall create a single AuditableEvent object as follows:

   If RegistryObjects were updated by the request, it contain a single Action sub-element with eventType Updated for all the RegistryObjects updated during processing of the request

b) The server should create AuditableEvents *after* successfully processing the request in a separate transaction from the request

5.4.3.5 Sample UpdateObjectsRequest

EXAMPLE: the following example shows an UpdateObjectsRequest which updates the Name element within a PersonType instance with the Name element specified by the Value element within UpdateAction. The Selector element uses an XPATH expression to select the Name element node within the Person objects identified as target of update in the ObjectRefList. The context node of the XPATH expression is the RegistryObject element for the PersonType instance. The target objects could also have been chosen by a Query element.

```
<UpdateObjectsRequest ...>
  <rim:ObjectRefList>
    <rim:ObjectRef id="urn:acme:person:Danyal"/>
  </rim:ObjectRefList>
  <UpdateAction mode="Update">
    <Value xsi:type="rim:AnyValueType">
      <rim:Name>
        <rim:LocalizedString xml:lang="en-US" value="Danny"/>
      </rim:Name>
    </Value>
    <Selector xsi:type="rim:StringQueryExpressionType"
      queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:XPath">

      <rim:Value>./rim:Name</rim:Value>
    </Selector>
  </UpdateAction>
</UpdateObjectsRequest>
```

5.4.4 RemoveObjects Protocol

5.4.4.1 Overview

The Remove Objects protocol allows a client to remove or delete one or more RegistryObject instances from the server.

A client initiates the RemoveObjects protocol by sending a RemoveObjectsRequest message to the LifecycleManager endpoint.

The LifecycleManager sends a RegistryResponse back to the client as response. This is shown in Figure 15.

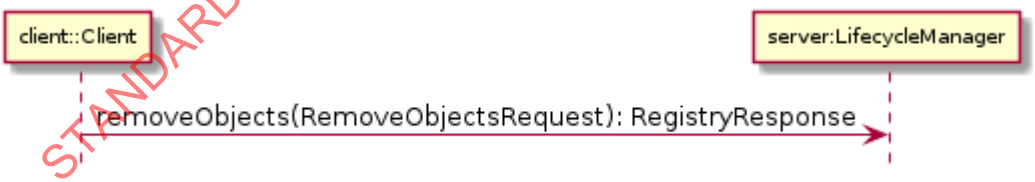

Figure 15 RemoveObjects protocol

5.4.4.2 RemoveObjectsRequest

The RemoveObjectsRequest message is sent by a client to remove one or more existing RegistryObjects from the server.

5.4.4.2.1 XML schema element definition

```
<element name="RemoveObjectsRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <element name="Query" type="rim:QueryType"
            minOccurs="0" maxOccurs="1" />
          <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attribute name="checkReferences" type="boolean" use="optional"
          default="false"/>
        <attribute name="deleteChildren" type="boolean" use="optional"
          default="false"/>
        <attribute name="deletionScope" type="rim:objectReferenceType"
          use="optional" default="urn:oasis:names:tc:ebxml-
regrep:DeletionScopeType:DeleteAll"/>
      </extension>
    </complexContent>
  </complexType>
</element>
```

5.4.4.2.2 Description

a) Attribute checkReferences – Specifies the reference checking behavior expected of the server
   1) true - Specifies that a server shall check objects being removed and make sure that there are no references to them from other objects via reference attributes and slots. If a reference exists then the server shall return ReferencesExistsException
   2) false (default) – Specifies that a server shall not check objects being removed to make sure that there are no references to them from other objects via reference attributes and slots. If a reference exists then the server shall not return ReferencesExistsException

b) Attribute deleteChildren – This attribute specifies whether or not to delete children of the objects being deleted according to the following behavior:
   1) false – Specifies the server shall not delete the children of objects that are specified to be deleted
   2) true – Specifies the server shall delete children of objects being deleted if and only if those children are not children of any other parent objects

c) Attribute deletionScope - This attribute specifies the scope of impact of the RemoveObjectsRequest. The value of the deletionScope attribute shall be a reference to a ClassificationNode within the canonical DeletionScopeType ClassificationScheme as described in ebRIM. A server shall support the deletionScope types as defined by the canonical DeletionScopeType ClassificationScheme. The canonical DeletionScopeType ClassificationScheme may be extended by adding additional ClassificationNodes to it.

   The following canonical ClassificationNodes are defined for the DeletionScopeType ClassificationScheme:

   1) DeleteRepositoryItemOnly - Specifies that the server shall delete the RepositoryItem for the specified ExtrinsicObjects but shall not delete the specified ExtrinsicObjects

   2) DeleteAll (default) - Specifies that the request shall delete both the RegistryObject and the RepositoryItem (if any) for the specified objects

d) Element Query - Specifies a query to be invoked. A server shall remove all objects that match the specified query in addition to any other objects identified by the ObjectRefList element.

e) Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances in the server. A server shall remove all objects that are referenced by this element in addition to any other objects identified by the Query element.

5.4.4.2.3 Returns:

This request returns a RegistryResponse.

5.4.4.2.4 Exceptions:

In addition to the exceptions common to all requests, the following exceptions may be returned:

a) UnresolvedReferenceException - Indicates that the requestor referenced an object within the request that was not resolved during the processing of the request.
b) ReferencesExistException - Indicates that the requestor attempted to remove a RegistryObject while references to it still exist. Note that it is valid to remove a RegistryObject and all RegistryObjects that refer to it within the same request. In such cases the ReferencesExistException shall not be thrown.

5.4.4.3 Audit trail requirements

a) The server shall create a single AuditableEvent object as follows:

   If RegistryObjects were removed by the request, it contain a single Action sub-element with eventType Deleted for all the RegistryObjects removed during processing of the request

b) The server should create AuditableEvents *after* successfully processing the request in a separate transaction from the request

5.4.4.4 Sample RemoveObjectsRequest

The following is a sample RemoveObjectsRequest to remove an Object by its id.

```
<lcm:RemoveObjectsRequest ...>
    <rim:ObjectRefList>
        <rim:ObjectRef id="urn:acme:Person:Danyal"/>
    </rim:ObjectRefList>
</lcm:RemoveObjectsRequest>
```

5.5 Version control

5.5.1 Overview

This clause describes the version control features of the ebXML RegRep.

Versioning of a RegistryObjectType instance is the process of updating the object in such a way that the original instance remains unchanged while a new instance is created as a new version of the original instance. Any specific version of an object may itself be versioned. Thus in general the versions of an object form a tree structure referred to as the Version Tree for that object.

A *Version Tree* for an object is defined to be a tree structure where:

a) There is a single root node for the tree
b) The root is the original version
c) Each non-root node in the tree is a version of the object
d) Each version is created from a parent version and is represented in the version tree as a child node of the node representing the parent version node for that version

EXAMPLE:

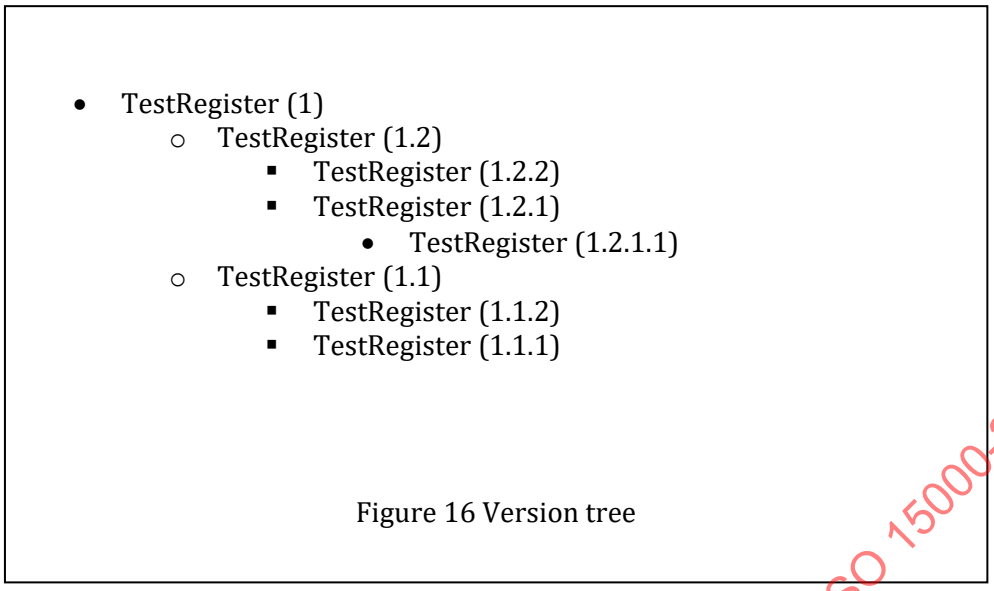

Figure 16 Version tree

Figure 16 visualizes the version tree concept. In this non-normative example the object TestRegister has 8 versions. Each node's version is identified by the parenthesized string suffix like "(1.2.2)". Version 1 is the original version. Version 1 was versioned twice to create versions 1.1 and 1.2. Version 1.1 was versioned twice to create versions 1.1.1 and 1.1.2. Version 1.2 was versioned twice to create versions 1.2.1 and 1.2.2. Version 1.2.1 was versioned once to create version 1.2.1.1. Note that this example uses a version naming convention for ease of understanding only. This specification does not prescribe a specific version naming convention for server to use when assigning version names.

The terms "logical object" or "logical RegistryObject" are used to refer to all version of a RegistryObject in a version independent manner. The terms "object version" or "RegistryObject version" are used to refer to a specific version of the logical object. The terms "RegistryObject instance" and "RegistryObjectType instance" imply a specific object version.

5.5.2 Version controlled resources

Version controlled resources are resources that support versioning capability.

All repository items in an ebXML RegRep are implicitly version-controlled resources as defined by clause 2.2.1 of RFC 3253. No explicit action is required to make them a version-controlled resource.

Instances of RegistryObjectType types are also implicitly version-controlled resources. The only exceptions are those sub-types of RegistryObjectType that are composed types and their instances do not have independent lifecycles that are separate from the lifecycle of their parent objects.

Composed object types are identified in class diagrams in the registry information model (see clause 4) as classes with composition or "solid diamond" relationship with a RegistryObject type.

EXAMPLE: Some examples of such composed types are:

a) ClassificationType
b) ExternalIdentifierType
c) ExternalLinkType
d) ServiceEndpointType

A server may further limit specific non-composed types from being version-controlled resources based upon server specific policies.

5.5.3 Versioning and id attribute

Each object version of a logical RegistryObject is a unique object and as such has its own unique value for its id attribute as defined by the registry information model (see clause 4).

5.5.4 Versioning and lid attribute

A RegistryObject instance shall have a *Logical ID (LID)* defined by its "lid" attribute to identify the logical RegistryObject of which it is a version. All versions of a logical RegistryObject have the same "lid" attribute value. Note that this is in contrast with the "id" attribute that shall be unique for each version of the same logical RegistryObject. A client may refer to the logical RegistryObject in a version independent manner using its LID.

5.5.5 Version identification for RegistryObjectType

A RegistryObjectType instance shall have a VersionInfo element whose type is the VersionInfoType type defined by ebRIM. The VersionInfo element identifies the version information for that RegistryObjectType instance. The versionName attribute of the VersionInfo element identifies the version name for a specific version of a logical object. A server shall not allow two versions of the same logical object to have the same versionName attribute value within its VersionInfo element.

5.5.6 Version identification for RepositoryItem

5.5.6.1 Overview

When a RegistryObject is an ExtrinsicObject with an associated repository item, the version identification for the repository item is distinct from the version identification for the ExtrinsicObject.

An ExtrinsicObject that has an associated repository item shall have a contentVersionInfo element whose type is VersionInfoType defined by ebRIM. The contentVersionInfo attributes identifies the version information for that repository item instance.

5.5.6.2 Versioning of RegistryObjectType

This clause describes the versioning of all RegistryObjectType types with the exception of ExtrinsicObjectType which is defined in a separate clause.

The following rules apply to versioning of all RegistryObjectType instances that are not instances of ExtrinsicObjectType type. It assumes that versioning is enabled for such RegistryObjectType types:

a) A server shall create a new version of a version-controlled, non-composed RegistryObjectType instance in the following cases:
   1) An existing object is replaced using the submitObjects protocol with mode of CreateOrVersion
   2) An existing object is updated using the updateObjects protocol with mode of CreateOrVersion
b) A server shall not create a new version of a composed RegistryObjectType instance when it is updated.
c) When creating a new version for a non-composed RegistryObjectType instance, a server shall create new logical objects for any composed logical objects within the new version of the composed object. Any such new logical object for composed objects shall have a new server generated universally unique id and lid attribute.

5.5.6.3 Versioning of ExtrinsicObjectType

The ExtrinsicObjectType type requires special consideration for versioning because it may have an associated RepositoryItem which is versioned independently from the ExtrinsicObjectType instance.

The following rules apply to versioning of ExtrinsicObjectType instances assuming that a server has versioning enabled for the ExtrinsicObjectType type:

a) A server shall create a new version of an existing ExtrinsicObjectType instance and assign it a new unique versionName within its VersionInfo element when either the ExtrinsicObjectType instance or its RepositoryItem are updated using the submitObjects or updateObjects protocol and the mode is CreateOrVersion
b) A server shall create a new version of an ExtrinsicObjectType instance and assign it a new unique versionName within its VersionInfo element when the previous version had a RepositoryItem and the new version does not have one (RepositoryItem was deleted).
c) A server shall create a new version of an ExtrinsicObjectType instance and assign it a new unique versionName within its VersionInfo element when the previous version did not have RepositoryItem and the new version has one (RepositoryItem was added). In such cases the server shall also create a new version of the RepositoryItem and assign it a new unique versionName within the ContentVersionInfo element.
d) A server shall create a new version of the RepositoryItem for an existing ExtrinsicObjectType instance and assign it a new unique versionName within the ContentVersionInfo element when the RepositoryItem is updated using the submitObjects or updateObjects protocol and the mode is CreateOrVersion

5.5.7 Versioning and references

An object reference from a RegistryObjectType instance references a specific version of the referenced RegistryObjectType instance. When a server creates a new version of a referenced RegistryObjectType instance it shall not move references from other objects from the previous version to the new version of the referenced object. Clients that wish to always reference the latest versions of an object may use the "dynamic reference" defined in ebRIM feature to always reference the latest version.

A special case is when a SubmitObjectsRequest contains an object that is being versioned by the server and the request contains other objects that reference the object being versioned. In such case, the server shall update all references within the submitted objects to the object being versioned such that those objects now reference the new version of the object being created by the request.

5.5.8 Versioning of RegistryPackages

When a server creates a new version of a RegistryPackageType instance, it shall implicitly make all members of the old version also be members of the new version. This requires that the server shall make a copy of all HasMember Associations in which the old version of the RegistryPackage is the sourceObject as follows:

a) The copied Associations shall be new versions of their original Association (shall have the same lid)
b) The sourceObject of the copied Associations shall reference the new version of the RegistryPackage rather than the older version

5.5.9 Versioning and RegistryPackage membership

A RegistryPackage shall not contain more than version of the same logical object as its member.

A server shall return an InvalidRequestException fault message if a client attempts to publish more than one version of the same logical object as member of the same RegistryPackage instance

5.5.10 Inter-version association

Each RegistryObject node in the version tree of a logical object except for the root version shall be linked to the RegistryObject node in the version tree that was its immediate predecessor (previous version).

A server shall automatically link each new version in the version tree for a RegistryObject to its predecessor using an Association between the two versions

a) The type attribute value of the Association shall reference the canonical AssociationType "Supersedes"
b) The sourceObject attribute value of the Association shall reference the new version
c) The targetObject attribute value of the Association shall reference the old version

Note that this clause is functionally equivalent to the predecessor-set successor-set elements of the Version Properties as defined by RFC 3253.

5.5.11 Version removal

Specific versions of a logical object may be deleted using the RemoveObjects protocol by specifying the version by its unique id.

a) A server may allow authorized clients to remove specified versions of a RegistryObject
b) A server may prune older versions of RegistryObjects based upon server specific administrative policies in order to manage storage resources
c) When a non-leaf version within a version tree is deleted, a server shall implicitly delete the entire version sub-tree under that non-leaf version such that no versions created directly or indirectly from the specified remain in the registry

5.5.12 Locking and concurrent modifications

This specification does not define explicit checkin and checkout capabilities as defined by RFC 3253. A server may support such features in an implementation specific manner.

This specification does not prescribe a locking model. An implementation may choose to support a locking model in an implementation specific manner. A future specification may address these capabilities.

5.5.13 Version creation

The server manages creation of new version of a version-controlled resource automatically. A server that supports versioning shall implicitly create a new version for the resource if an existing version of the resource is updated via a SubmitObjectsRequest or UpdateObjectsRequest when the mode attribute value is CreateOrVersion. A server shall update the existing version of a resource without creating a new version when the mode attribute is set to CreateOrReplace.

5.6 Validator interface

5.6.1 Overview

The Validator interface allows the validation of objects published to the server. The interface may be used by clients to validate objects already published to the server or may be used by the server to validate objects during the processing of the submitObjects or updateObjects protocol

A server shall implement the Validator interface as an endpoint. The Validator interface validates objects using Validator Plugins specific to the type of object being validated.

5.6.2 ValidateObjects protocol

5.6.2.1 Overview

The ValidateObjects protocol is initiated by sending an ValidateObjectsRequest message to the Validator endpoint.

Figure 17 ValidateObjects protocol

The Validator endpoint sends an ValidateObjectsResponse back as response. The ValidateObjectsResponse contains information on whether the objects were valid and if invalid objects were found it includes any validation errors that were encountered.

5.6.2.2 ValidateObjectsRequest

The ValidateObjectsRequest message initiates the validateObjects protocol and specifies the objects that need to be validated.

5.6.2.2.1 XML schema element definition

```
<element name="ValidateObjectsRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <element name="Query" type="rim:QueryType"
            minOccurs="0" maxOccurs="1" />
          <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
          <element name="OriginalObjects" type="rim:RegistryObjectListType"
            minOccurs="1" maxOccurs="1"/>
          <element name="InvocationControlFile"
            type="rim:ExtrinsicObjectType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

EXAMPLE: the following example shows a client request to validate a specified WSDL file. It assumes that the server will be configured with a Validator plugin for WSDL files. It also assumes that the server

will specify OriginalObjects and InvocationControlFile elements when it relays the request to the appropriate Validator plugin.

```
<spi:ValidateObjectsRequest ...>
  <rim:ObjectRefList>
    <rim:ObjectRef id="urn:acme:wsdl:purchaseOrder.wsdl"/>
  </rim:ObjectRefList>
</ValidateObjectsRequest>
```

5.6.2.2.2 Description

a) Element InvocationControlFile – Specifies an ExtrinsicObject that is used to control the validation process in a type specific manner. See Canonical XML Validator plugin for an example. This element may be specified by server when sending the request to the Validator plugin if the Validator plugin requires an invocation control file. It should not be specified by the client.
   1) Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances in the server. A server shall validate all objects that are referenced by this element. This element is typically used when a client initiates the validateObjects protocol.
   2) Element OriginalObjects - Specifies a collection of RegistryObject instances. A server shall validate all objects that are contained in this element. This element is typically used when a server initiates the validateObjects protocol during the processing of a submitObjects or updateObjects protocol request or when it is delegating a client initiated validateObjects protocol request to a Validator plugin.

b) Element Query - Specifies a query to be invoked. A server shall validate all objects that match the specified query. This element is typically used when a client initiates the validateObjects protocol.

5.6.2.2.3 Response

This request returns ValidateObjectsResponse as response.

5.6.2.2.4 Exceptions

In addition to the common exceptions, the following exceptions may be returned:

ValidationException: signifies that an exception was encountered during the validateObjects operation

5.6.2.3 ValidateObjectsResponse

Currently ValidateObjectsResponse is a simple extension to RegistryResponseType and does not define additional attributes or elements.

5.6.3 Validator plugins

5.6.3.1 Overview

Validator plugins allow a server to use specialized extension modules to validate specific types of objects during the processing of a SubmitObjectsRequest, UpdateObjectsRequest or a ValidateObjectsRequest.

A specific instance of a Validator plugin is designed and configured to validate a specific type of object.

EXAMPLE: the canonical XML Validator plugin is designed and configured to validate XML Objects using Schematron documents as InvocationControlFile.

5.6.3.2 Validator plugin interface

A Validator plugin implements the Validator interface. The server's Validator endpoint should delegate a validateObjects operation to any number of Validator plugins using the following algorithm:

a) The server selects the RegistryObjects that are the target of the validateObjects operations using the <spi:Query> and <rim:ObjectRefList> elements. Any objects specified by the OriginalObjects element shall be ignored by the server.
b) The server partitions the set of target objects into multiple sets based upon the objectType attribute value for the target objects
c) The server determines whether there is a Validator plugin configured for each objectType for which there is a set of target objects
d) For each set of target objects that share a common objectType and for which there is a configured Validator plugin, the server shall invoke the Validator plugin. The Validator plugin invocation shall specify the target objects for that set using the OriginalObjects element. The server shall not specify <spi:Query> and <rim:ObjectRefList> elements when invoking validateObjects operation on a Validator plugin
e) Each Validator plugin shall process the ValidateObjectsRquest and return a ValidateObjectsResponse or fault message to the server's Validator endpoint.
f) The server's Validator endpoint shall then combine the results of the individual ValidateObjectsRequest to Validator plugins into a single unified ValidateObjectsResponse and return it to the client.

5.6.3.3 Canonical XML validator plugin

The canonical XML Validator plugin is a validator plugin that validates XML content using a Schematron file as InvocationControlFile. The Schematron file specifies validation rules using the ISO 19757-3 Schematron language to validate XML content. The server may configure the canonical XML Validator plugin such that it is invoked with an appropriate schematron file as InvocationControlFile based upon the objectType of the object being validated.

5.7 Cataloger interface

5.7.1 Overview

The Cataloger interface allows a client to catalog or index objects already in the server. The interface may be used by clients to catalog objects already published to the server or may be used by the server to catalog objects during the processing of the submitObjects or updateObjects protocol .

A server shall implement the Cataloger interface as an endpoint. The Cataloger interface catalogs objects using Cataloger Plugins specific to the type of object being cataloged.

5.7.2 CatalogObjects protocol

5.7.2.1 Overview

A client catalogs RegistryObjects residing in the server using the *CatalogObjects* protocol supported by the catalogObjects operation of the Cataloger interface.

The CatalogObjects protocol is initiated by sending an CatalogObjectsRequest message to the Cataloger endpoint.

Figure 18 CatalogObjects protocol

The Cataloger endpoint sends a CatalogObjectsResponse back to the client as response.

5.7.2.2 CatalogObjectsRequest

The CatalogObjectsRequest message initiates the catalogObjects protocol and specifies the objects that need to be cataloged.

5.7.2.2.1 XML schema element definition

```
<element name="CatalogObjectsRequest">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryRequestType">
        <sequence>
          <element name="Query" type="rim:QueryType"
            minOccurs="0" maxOccurs="1" />
          <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
          <element name="OriginalObjects" type="rim:RegistryObjectListType"
            minOccurs="0" maxOccurs="1"/>
          <element name="InvocationControlFile"
            type="rim:ExtrinsicObjectType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
```

EXAMPLE: the following example shows a client request to catalog a specified WSDL file. It assumes that the server will be configured with a Cataloger plugin for WSDL files. It also assumes that the server will specify OriginalObjects and InvocationControlFile elements when it relays the request to the appropriate Cataloger plugin.

```
<spi:CatalogObjectsRequest ...>
  <rim:ObjectRefList>
    <rim:ObjectRef id="urn:acme:wsdl:purchaseOrder.wsdl"/>
  </rim:ObjectRefList>
</CatalogObjectsRequest>
```

5.7.2.2.2 Description

a) Element InvocationControlFile – Specifies an ExtrinsicObject that is used to control the cataloging process in a type specific manner. See Canonical XML Catalogor plugin for an example. This element may be specified by server when sending the request to the Cataloger plugin if the Cataloger plugin requires an an invocation control file. It should not be specified by the client.
b) Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances in the server. A server shall catalog all objects that are referenced by this element. This element is typically used when a client initiates the catalogObjects protocol.
c) Element OriginalObjects - Specifies a collection of RegistryObject instances. A server shall catalog all objects that are contained in this element. This element is typically used when a

server initiates the catalogObjects protocol during the processing of a submitObjects or updateObjects protocol request or when it is delegating a client initiated catalogObjects protocol request to a Cataloger plugin.

d) Element Query - Specifies a query to be invoked. A server shall catalog all objects that match the specified query. This element is typically used when a client initiates the catalogObjects protocol.

5.7.2.2.3 Response

This request returns CatalogObjectsResponse as response.

5.7.2.2.4 Exceptions

In addition to common exceptions, the following exceptions may be returned:

CatalogingException: signifies that an exception was encountered during the catalogObjects operation

5.7.2.3 CatalogObjectsResponse

The CatalogObjectsResponse message is sent by the Cataloger endpoint in response to a CatalogObjectsRequest.

5.7.2.3.1 XML schema element definition

```
<element name="CatalogObjectsResponse">
  <complexType>
    <complexContent>
      <extension base="rs:RegistryResponseType">
      </extension>
    </complexContent>
  </complexType>
</element>
```

EXAMPLE: the following example shows a CatalogObjectsResponse sent by a server to the client in response to a CatalogedObjectRequest. It shows that the Cataloger augmented the Original object with a new Slot that catalogs the target namespace used by the WSDL file.

```
<CatalogObjectsResponse status="urn:oasis:names:tc:ebxml-
regrep:ResponseStatusType:Success">
  <rim:RegistryObjectList>
    <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType"
      mimeType="text/xml"
      status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
      objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:WSDL"
      lid="urn:acme:wsdl:purchaseOrder.wsdl"
      id="urn:acme:wsdl:purchaseOrder.wsdl">
      <rim:Slot
        name="urn:oasis:names:tc:ebxml-regrep:profile:wsdl:slot:targetNamespace">
        <rim:SlotValue xsi:type="rim:StringValueType">
          <rim:Value>urn:acme:Service:PurchaseOrder</rim:Value>
        </rim:SlotValue>
      </rim:Slot>
      <rim:RepositoryItem>...binary encoded content...</rim:RepositoryItem>
    </rim:RegistryObject>
  </rim:RegistryObjectList>
</CatalogObjectsResponse>
```

5.7.2.3.2 Description

In addition to elements and attributes defined by RegistryResponseType the following are defined:

a) Element RegistryObjectList (Inherited) – Contains the RegistryObjects that are produced as output of the catalogObjects operation. Typically this list contains the objects that were input to the catalogObjects operation, as well as new objects that were the output of the catalogObjects operation. The input objects may be modified by the cataloger as a result of the catalogObjects operation.
b) A cataloger shall create AssociationType instance between the source object for the catalogObjects operation (specified by OriginalObjects element in CatalogRequest) and each of the cataloged RegistryObjectType instances generated by the cataloger. Each such AssociationType instance
    1) shall have its type attribute reference the canonical AssociationType "urn:oasis:names:tc:ebxml-regrep:AssociationType:HasCatalogedMetadata"
    2) shall have its sourceObject attribute reference the source object for the catalogObjects operation
    3) shall have its targetObject attribute reference a cataloged RegistryObjectType instance generated by the cataloger
c) A cataloger should assign the same accessControlPolicy to cataloged objects as their source object. A cataloger may use a different strategy for assigning access control policy to cataloged objects.
d) A server shall delete all cataloged metadata generated by a cataloger when the source object is deleted.
e) A server shall update all cataloged metadata generated by a cataloger when the source object is updated without creating a new version.

5.7.3 Cataloger plugins

5.7.3.1 Overview

Cataloger plugins allow a server to use specialized extension modules to catalog specific types of objects during the processing of a SubmitObjectsRequest, UpdateObjectsRequest or a CatalogObjectsRequest.

A specific instance of a Cataloger plugin is designed and configured to catalog a specific type of object.

EXAMPLE: the canonical XML Cataloger plugin is designed and configured to catalog XML Objects using XSLT documents as InvocationControlFile.

5.7.3.2 Cataloger plugin interface

A Cataloger plugin implements the Cataloger interface. The server's Cataloger endpoint should delegate a catalogObjects operation to any number of Cataloger plugins using the following algorithm:

a) The server selects the RegistryObjects that are the target of the catalogObjects operations using the <spi:Query> and <rim:ObjectRefList> elements. Any objects specified by the OriginalObjects element shall be ignored by the server.
b) The server partitions the set of target objects into multiple sets based upon the objectType attribute value for the target objects
c) The server determines whether there is a Cataloger plugin configured for each objectType for which there is a set of target objects
d) For each set of target objects that share a common objectType and for which there is a configured Cataloger plugin, the server shall invoke the Cataloger plugin. The Cataloger plugin invocation shall specify the target objects for that set using the OriginalObjects element. The server shall not specify <spi:Query> and <rim:ObjectRefList> elements when invoking catalogObjects operation on a Cataloger plugin
e) Each Cataloger plugin shall process the CatalogObjectsRquest and return a CatalogObjectsResponse or fault message to the server's Cataloger endpoint.

f) The server's Cataloger endpoint shall then combine the results of the individual CatalogObjectsRequest to Cataloger plugins and commit these objects as part of the transaction associated with the request. It shall then combine the individual CatalogObjectsResponse messages into a single unified CatalogObjectsResponse and return it to the client.

5.7.3.3 Canonical XML cataloger plugin

The canonical XML Cataloger plugin is a Cataloger plugin that catalogs XML content using an XSLT file as InvocationControlFile. The XSLT file specifies transformations rules using the XSLT language to catalog XML content. The server may configure the canonical XML Cataloger plugin such that it is invoked with an appropriate XSLT file as InvocationControlFile based upon the objectType of the object being cataloged.

An XSLT file used as InvocationControlFile with the Canonical XML Cataloger shall meet the following constraints:

a) Support an ExtrinsicObject as primary input
b) Support an XML RepositoryItem for the ExtrinsicObject object as a secondary input
c) The secondary input is specified using an <xsl:param> with name “repositoryItem” and with value that is the id of the ExtrinsicObject for which it is a RepositoryItem

A server shall implement the Canonical XML Cataloger with the following constraints:

a) Uses an XSLT processor with the XSLT file specified as InvocationControlFile
b) Specifies the ExtrinsicObject being cataloged as the primary input to the XSLT processor
c) Specifies the RepositoryItem for the ExtrinsicObject object being cataloged by setting the parameter named “repositoryItem” with a value that is the id of the ExtrinsicObject for which it is a RepositoryItem
d) Resolves references to the RepositoryItem via the $repositoryItem parameter value within the XSLT file specified as InvocationControlFile

5.8 Subscription and notification

5.8.1 Overview

A client may subscribe to events that transpire in the server by creating a Subscription. A server supporting Subscription and Notification feature shall deliver a Notification to the subscriber when an event transpires that matches the event selection criteria specified by the client.

5.8.2 Server events

5.8.2.1 Overview

Activities within the server result in events. The registry information model (see clause 4) defines the AuditableEvent element, instances of which represent server events. A server creates AuditableEvent instances during the processing of client requests.

5.8.2.2 Pruning of events

A server may periodically prune AuditableEvents in order to manage its resources. It is up to the server when such pruning occurs. A server should perform such pruning by removing the older AuditableEvents first.

5.8.3 Notifications

A Notification message is used by the server to notify clients of events they have subscribed to. A Notification contains the RegistryObjects, or references to the RegistryObjects, that are affected by the event for which the Notification is being sent, based upon the notificationOption within the DeliveryInfo for the subscription.

Details for the Notification element are defined in the registry information model (see clause 4).

5.8.4 Creating a subscription

5.8.4.1 Overview

A client may create a subscription within a server if it wishes the server to send it a Notification when a specific type of event transpires. A client creates a subscription by submitting a rim:SubscriptionType instance to the server using the standard SubmitObjects protocol.

Details for the rim:SubscriptionType are defined in the registry information model (see clause 4).

5.8.4.2 Subscription authorization

A deployment may use custom Access Control Policies to decide which users are authorized to create a subscription and to what events. A server shall return an AuthorizationException in the event that an unauthorized user submits a Subscription to a server.

5.8.4.3 Subscription quotas

A server may use server specific policies to decide an upper limit on the number of Subscriptions a user is allowed to create. A server should return a QuotaExceededException in the event that an authorized user submits more Subscriptions than allowed by their server-specific quota.

5.8.4.4 Subscription expiration

Each subscription may define a startTime and endTime attribute which determines the period within which a Subscription is valid. If startTime is unspecified then a server shall set it to the time of submission of the subscription. If endTime is unspecified then the server shall choose a default value based on its policies.

Outside the bounds of the valid period, a Subscription may exist in an expired state within the server. A server may remove an expired Subscription at any time.

A server shall not deliver notifications for an event to an expired Subscription. An expired Subscription may be renewed by updating the startTime and / or endTime for the Subscription using the UpdateObjects protocol.

5.8.4.5 Event selection

A client shall specify a Selector element within the Subscription to specify its criteria for selecting events of interest. The Selector element is of type rim:QueryType and specifies an parameterized query to be invoked with specified query parameters.

A server shall process AuditableEvents and determine which Subscriptions match the event using the algorithm illustrated by the following pseudo-code fragment:


```
//Get objects that match selector query
List<RegistryObjectType> objectsOfInterest =
  getObjectsMatchingSelectorQuery(selectorQuery);

if (objectsOfInterest.size() > 0) {

  //Now get AuditableEvents that affected objectsOfInterest
  //shall not include AuditableEvents that have already been delivered
  //to this subscriber
  List<RegistryObjectType> eventsOfInterest =
    getEventsOfInterest(objectsOfInterest);

  if (eventsOfInterest.size() > 0) {
    //Now create Notification on objectsOfInterest.
    //Notification will include eventsOfInterest that only include objects
    //that are affected by the event and are also in objectsOfInterest
    NotificationType notification = createNotification(
      objectsOfInterest, eventsOfInterest);

    //Now send notification using info in DeliveryInfo
    sendNotification(notification);
  }
}
```

a) Objects of interest shall be those objects that match the selector query for the subscription
b) Events of interest shall have affected at least one object of interest
c) Events of interest shall contain all objects of interest (or references to them) that were affected by the event
d) Events of interest shall not contain an object or reference to an object that is not an object of interest

5.8.5 Event delivery

5.8.5.1 Overview

A client may specify zero or more DeliveryInfo elements within the Subscription to specify how the server should deliver events matching the subscription to the client. The DeliveryInfo element shall include a NotifyTo element which specifies an EndPoint Reference (EPR) as defined by the Web Services Addressing 1.0 – Core recommendation. The NotifyTo element contains a <wsa:Address> element which contains a URI to the endpoint.

Details for the DeliveryInfo element are defined in the registry information model (see clause 4).

5.8.5.2 Notification option

A client may specify a notificationOption attribute in DeliveryInfo element of a Subscription. The notificationOption attribute specifies how the client wishes to be notified of events. This attribute controls whether the Event within a Notification contains complete RegistryObjectType instances or only ObjectRefType instances. It is defined in detail in ebRIM.

5.8.5.3 Delivery to NotificationListener web service

If the <wsa:Address> element has a rim:endpointType attribute value of “urn:oasis:names:tc:ebxml-regrep:endPointType:soap”, then the server shall use the specified address as the web service endpoint URL to deliver the Notification to. The target web service in this case shall implement the NotificationListener interface.

5.8.5.4 Delivery to email address

If the <wsa:Address> element has a rim:endpointType attribute value of "urn:oasis:names:tc:ebxml-regrep:endPointType:rest", then the server shall use the specified address as the email address to deliver the Notification via email. This specification does not define how a server is configured to send Notifications via email.

5.8.5.5 Delivery to a NotificationListener plugin

If the <wsa:Address> element has a rim:endpointType attribute value of "urn:oasis:names:tc:ebxml-regrep:endPointType:plugin", then the server shall use the specified address as a Notification plugin identifier and deliver the Notification via local call to the plugin. This specification does not define how a server is configured for Notification plugins.

5.8.5.5.1 Processing email notification via XSLT

A client may specify an XSLT style sheet within a DeliveryInfo element to process a Notification prior to it being delivered to an email address. The XSLT style sheet may be specified using a Slot in DeliveryInfo element where the Slot's name is "urn:oasis:names:tc:ebxml-regrep:rim:DeliveryInfo:emailNotificationFormatter" and the Slots value is the id of an ExtrinsicObject whose repository item is the XSLT. The ExtrinsicObject and repository item shall be submitted prior to or at the same time as the Subscription.

5.8.6 NotificationListener interface

The NotificationListener interface allows a client to receive Notifications from the server for their Subscriptions. A client shall implement the NotificationListener interface as an endpoint if they wish to receive Notifications via SOAP or REST. A server shall implement a NotificationListener interface as an endpoint if it supports the object replication feature as this endpoint will be used by remote servers to deliver Notification of changes to replicated objects.

5.8.7 Notification protocol

5.8.7.1 Overview

A server sends a Notification to an endpoint using the *Notification* protocol supported by the onNotification operation of the NotificationListener interface.

A server initiates the Notification protocol by sending a Notification message to the NotificationListener endpoint registered within the Subscription for which the Notification is being delivered. This is shown in Figure 19.

Figure 19 Notification protocol

The onNotification operation does not send a response back to the server.

5.8.7.2 Notification

The Notification message is sent by the server to a NotificationListener interface implemented by the client and delivers an event notification for a subscription. It is a one-way request pattern and produces no response. The syntax and semantics of the Notification message is described in detail in ebRIM.

5.8.8 Pulling notification on demand

A client may "pull" Notifications for a Subscription by invoking the GetNotification canonical query. A client may specify a startTime since which it wishes to include events within the pulled Notification. If client does not specify a startTime then all events since the last "push" delivery to that client's NotifyTo endpoint shall be included in the Notification. If Subscription does not define any "push" delivery for that client's NotifyTo endpoint then a client shall use startTime parameter to avoid getting the same events within the Notification returned by the GetNotification query.

Pulling a Notification leaves the Notification intact on the server for any potential pushing of the Notification to endpoints defined in DeliveryInfo elements of the Subscription.

5.8.9 Deleting a subscription

A client may terminate a Subscription with a server if it no longer wishes to be notified of events related to that Subscription. A client terminates a Subscription by deleting the corresponding Subscription object using the standard RemoveObjects protocol.

5.9 Multi-server features

5.9.1 Overview

This clause describes features of ebXML RegRep that involve more than one ebXML RegRep server instances. These features include:

a) Remote object reference – allows references between objects residing in different servers
b) Object replication – allows replication of objects residing in a remote server to a local server
c) Federated queries – allows queries that execute against, and return results from multiple servers

5.9.2 RemoteObjects reference

A RegistryObject in one ebXML RegRep server may contain a reference to a RegistryObject in *any* other ebXML RegRep server that is compatible with this specification of a compatible version number as the source server. Remote object reference feature does not require the local and remote servers to be part of the same federation. Remote object references are described in detail in the registry information model (see clause 4). This is shown in Figure 20.

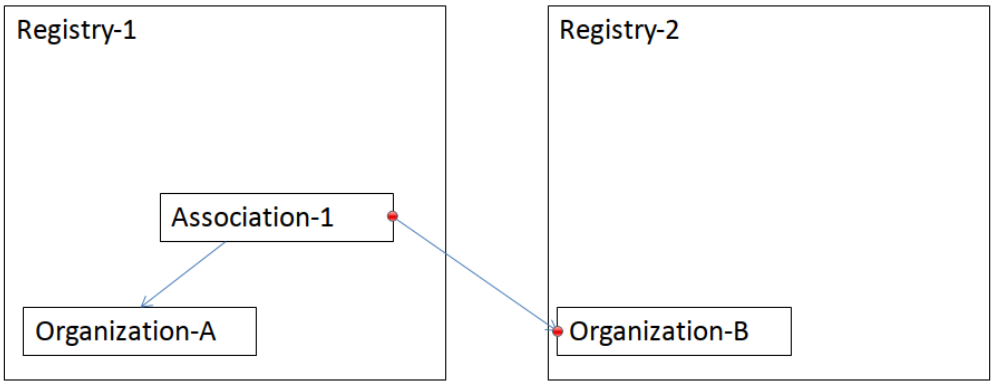

Figure 20 Remote object reference

5.9.3 Local replication of remote objects

5.9.3.1 Overview

RegistryObjects within a server may be replicated in another server. A replicated copy of a remote object is referred to as its replica. The remote object may be an original object or it may be a replica. A replica from an original is referred to as a first-generation replica. A replica of a replica is referred to as a second-generation replica (and so on).

A server that replicates a remote object locally is referred to as the local server for the replication. The server that contains the remote object being replicated is referred to as the remote server for the replication. This is shown in Figure 21.

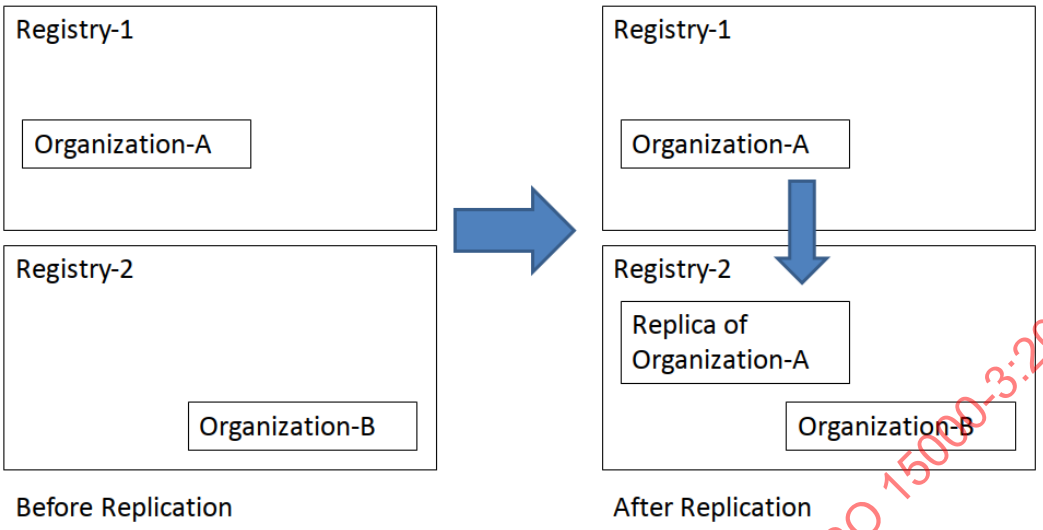

Figure 21 local replication of remote objects

The following rules govern replication of remote objects:

a) A server shall match local replicas of remote objects in the same manner as local objects within the Query protocol.
b) A client shall not perform update operations via SubmitObjects and UpdateObjects operations on a local replica of a remote object.
c) A server shall return an InvalidRequestException fault message if a client attempts to update a replica via SubmitObjects and UpdateObjects operations.
d) A server shall delete a replica if a client uses RemoveObjects operation to remove the replica.
e) Objects may be replicated from any server to any other server without any requirement that the registries belong to the same federation.

5.9.3.2 Creating local replica and keeping it synchronized

Replication feature relies upon the Subscription and Notification feature to keep replicas synchronized with changes to the remote object. A local replica of a remote objects is created as follows:

a) A client submits a Subscription to the remote server on behalf of the local server.
   1) The subscription is published like any other RegistryObjectType instance using the Submit Objects protocol with the LifecycleManager endpoint of the remote server.
   2) This typically requires that the client is registered with the remote server and can authenticate with it.
b) The Subscription defines a Selector query that matches one or more objects that need to be replicated from remote server to local server.

   Selector query may match any number of objects using any selection criteria supported by the query.

c) The Subscription specifies the address of a NotificationListener endpoint implemented by the local server where the remote server may send Notifications regarding the objects that need to be replicated.

