OASIS ebXML RegRep Registry
Information Model Version 4.0

Draft

October 15, 2008

Specification URIs:
This Version:
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.html
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.odt
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.pdf

Previous Version:
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.html
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.odt
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf

Latest Version:
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.html
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.odt
http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rim-4.0-draft-1.pdf

Latest Approved Version:
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.html
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.odt
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf

Technical Committee:
OASIS ebXML RegRep TC

Chair(s):
Kathryn Breininger, Boeing

Editor(s):
Farrukh Najmi, Wellfleet Software
Nikola Stojanovic, RosettaNet

Contributors:
Kathryn Breininger, Boeing
Carl Mattocks, MetLife
Farrukh Najmi, Wellfleet Software
Related Work:
This specification replaces or supercedes:
  • [specifications replaced by this standard - OASIS as well as other standards organizations]
  • [specifications replaced by this standard - OASIS as well as other standards organizations]
This specification is related to:
  • [specifications related to this standard - OASIS as well as other standards organizations]
  • [specifications related to this standard - OASIS as well as other standards organizations]

Declared XML Namespace(s):
This following table lists the namespace prefixes defined and / or referenced by this specification.
<table>
<thead>
<tr>
<th>Namespace Prefix</th>
<th>Namespace URI</th>
<th>Defining Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcm</td>
<td>urn:oasis:names:tc:ebxml-regrep:xsd:lcm:4.0</td>
<td>ebXML RegRep Services and Protocols 4.0 (ebRS)</td>
</tr>
<tr>
<td>query</td>
<td>urn:oasis:names:tc:ebxml-regrep:xsd:query:4.0</td>
<td>ebXML RegRep Services and Protocols 4.0 (ebRS)</td>
</tr>
<tr>
<td>rim</td>
<td>urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0</td>
<td>ebXML RegRep Registry Information Model 4.0 (ebRIM)</td>
</tr>
<tr>
<td>rs</td>
<td>urn:oasis:names:tc:ebxml-regrep:xsd:rs:4.0</td>
<td>ebXML RegRep Services and Protocols 4.0 (ebRS)</td>
</tr>
<tr>
<td>wsdl</td>
<td><a href="http://schemas.xmlsoap.org/wsd/wdsl/">http://schemas.xmlsoap.org/wsd/wdsl/</a></td>
<td>WSDL 1.1 namespace defined by WSDL 1.1 specification.</td>
</tr>
</tbody>
</table>

| Table 1: Namespaces Used |

**Abstract:**
This document defines the types of metadata and content that can be stored in an ebXML RegRep.

A separate document, OASIS ebXML RegRep: Service and Protocols [ebRS], defines the services and protocols for an ebXML RegRep.
**Status:**

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

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at [http://www.oasis-open.org/committees/regrep/](http://www.oasis-open.org/committees/regrep/).

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

The non-normative errata page for this specification is located at [http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/errata.pdf](http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/errata.pdf)
9 Event Information Model

9.1 AuditableEventType

9.1.1 Syntax

9.1.2 Example

9.1.3 Description

9.2 ActionType

9.2.1 Syntax

9.2.2 Description

9.3 SubscriptionType

9.3.1 Syntax

9.3.2 Example

9.3.3 Description

9.4 DeliveryInfoType

9.4.1 Syntax

9.4.2 Description

9.5 NotificationType

9.5.1 Syntax

9.5.2 Example

9.5.3 Description

10 Federation Information Model

10.1 Federation Configuration

10.2 RegistryType

10.2.1 Syntax

10.2.2 Example

10.2.3 Description

10.3 FederationType

10.3.1 Syntax

10.3.2 Example

10.3.3 Description

11 Access Control Information Model

11.1 Defining an Access Control Policy

11.2 Assigning Access Control Policy to a RegistryObject

11.3 Default Access Control Policy

11.4 Root Access Control Policy

11.5 Performance Implications

11.6 Action Matching

11.6.1 Action Attribute: reference-source

11.6.2 Action Attribute: reference-source-attribute
Illustration Index

Illustration 1: Information Model - Relationships View..........................................................12
Illustration 2: Information Model Inheritance View..................................................................13
Illustration 3: Classification Example.........................................................................................35
Illustration 4: Classification Information Model...........................................................................35
Illustration 5: Organization to RegistryObject Association..........................................................48
Illustration 6: Service Information Model Update Figure??..........................................................49
Illustration 7: Federation Information Model..................................................................................66
Illustration 8: Assigning Access Control Policy to a RegistryObject..............................................71

Index of Tables

Table 1: Namespaces Used..............................................................................................................3
1 Introduction

All text is normative unless otherwise indicated.

1.1 Terminology

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

1.2 Normative References


1.3 Non-normative References

[Reference] [reference citation]
2 Overview

The ebXML Registry Information Model is defined as an XML Schema in rim.xsd file. It defines the metadata types and their relationships within ebXML RegRep specifications.

2.0.1 Class Relationships View

Illustration 1 provides a high level overview of the metadata types defined by the model and their “Has-A” relationships as a UML Class Diagram. It does not show “Is-A” or Inheritance relationships nor does it show Class attributes. Further, it only shows a subset of classes in the model rather than all the classes in the model. The relationship links in the figure are either UML association or composition relationships (solid diamonds). In case of UML composition, instances of a class on the far side of the solid diamond are referred to as composed objects in the [ebRIM] and [ebRS] specifications.

2.0.2 Class Inheritance View

Illustration 2 shows the inheritance or “Is-A” relationships between the types in the information model. Note that it does not show the other types of relationships, such as “Has-A” relationships, since they have already been shown in Illustration 1. Class attributes are also not shown to conserve page space. Detailed description of attributes of each class will be displayed in tabular form within the detailed description of each class.
The types in the information model are presented in related groups as follows:

- **Core Information Model**: Defines core metadata types in the model including the common base types.
- **Association Information Model**: Defines types that enable objects to be associated with each other.
- **Classification Information Model**: Defines types that enable objects to be classified.
- **Provenance Information Model**: Defines types that enable the description of provenance or source information about an object.
- **Service Information Model**: Defines types that enable service description.
- **Query Information Model**: Defines types that enable definition and invocation of queries.
- **Event Information Model**: Defines types that enable the event subscription and notification feature defined in [ebRS].
● Federation Information Model: Defines types that enable the federated registries feature defined in [ebRS].
● Access Control Information Model: Defines types that enable access control and authorization for ebXML RegRep.

The remainder of this document will describe each of the above related group of classes in a dedicated chapter named accordingly.
3 Core Information Model

3.1 InternationalStringType

The InternationalStringType type is used throughout the schema whenever a textual value needs to be represented in multiple local languages. Two global elements Name and Description are defined to use InternationalStringType as their type.

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

3.1.1 Syntax

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

<element name="Name" type="tns:InternationalStringType"/>
<element name="Description" type="tns:InternationalStringType"/>
```

3.1.2 Example

```xml
<rim:Name>
  <rim:LocalizedString
    xml:lang="en-US" charset="UTF-8" value="freebXMLRegistry"/>
</rim:Name>
```

3.1.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalizedString</td>
<td>LocalizedStringType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

3.2 LocalizedStringType

This type allows the definition of a string value using the specified local language and character set. It is used within the InternationalStringType as the type of the LocalizedString sub-element.

3.2.1 Syntax

```xml
<complexType name="LocalizedStringType">
  <attribute ref="xml:lang" default="en-US" use="optional"/>
  <attribute default="UTF-8" name="charset" use="optional"/>
  <attribute name="value" type="tns:FreeFormText" use="required"/>
</complexType>
```
### 3.2.2 Example

```xml
<rim:LocalizedString
    xml:lang="en-US" charset="UTF-8" value="freebXMLRegistry"/>
```

### 3.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>lang</td>
<td>language</td>
<td>0..1</td>
<td>en-US</td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>charset</td>
<td>String</td>
<td>0..1</td>
<td>UTF-8</td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute `lang` - Each `LocalizedStringType` instance MAY have a `lang` attribute that specifies the language used by that `LocalizedStringType` instance.
- Attribute `charset` - Each `LocalizedStringType` instance MAY have a `charset` attribute that specifies the name of the character set used by that `LocalizedStringType` instance. The value of this attribute SHOULD be registered with IANA at: [http://www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets)
- Attribute `value` - Each `LocalizedStringType` instance MUST have a `value` attribute that specifies the string value used by that `LocalizedStringType` instance.

### 3.3 SlotType

This type is a container or wrapper that is capable of containing any type of information that may be represented in an XML document. It is an important extensibility mechanism with ebRIM.

A SlotType instance contains a `ValueList` element which contains one or more `ValueListItems`. It is the `valueListItems` that represent the values associated with the SlotType instance.

#### 3.3.1 Syntax

```xml
<complexType name="SlotType">
    <sequence>
        <element ref="tns:ValueList" minOccurs="1" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="tns:LongName" use="required"/>
    <attribute name="dataType" type="tns:LongName" use="optional"/>
    <attribute name="collectionType" type="tns:referenceURI" use="optional"/>
</complexType>
```

#### 3.3.2 Example

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

```xml
<rim:Slot
    name="spatialSlot1"
    dataType="urn:ogc:def:dataType:ISO-19107:GM_Envelope">
    <rim:ValueListItem xsi:type="rim:AnyValueType">
        <gml:Envelope srsName="urn:ogc:def:crs:OGC:2:WGS84">
            <gml:lowerCorner>-122.35 19.31</gml:lowerCorner>
            <gml:upperCorner>-61.80 48.93</gml:upperCorner>
        </gml:Envelope>
    </rim:ValueListItem>
</rim:Slot>
```
### 3.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectionType</td>
<td>ObjectRef</td>
<td>0..1</td>
<td>Client</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>dataType</td>
<td>LongName??</td>
<td>0..1</td>
<td>Client</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>LongName</td>
<td>1</td>
<td>Client</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ValueList</td>
<td>ValueListType</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

- **Attribute collectionType** – Defines the type of collection for the ValueList collection. Must be an ObjectRef that references a ClassificationNode in the canonical ClassificationScheme CollectionTypeScheme. A server MUST enforce the following semantics associated with the following canonical collection types:
  - List – Server MUST maintain the order of the values in the collection
  - Set – Server MUST NOT allow duplicate values in the collection
  - Sorted Set – Server MUST NOT allow duplicate values in the collection and MUST maintain a sort order according to the alphanumeric ordering of its elements according to the default local associated with the server
  - Bag – Server MUST allow duplicate values and MAY not maintain order of values
- **Attribute dataType** – A string that specifies the datatype for the values in the ValueList
- **Attribute name** – The name of this SlotType instance
- **Element ValueList** – This element is the container for the actual values within a SlotType instance.

### 3.4 ValueListType

This type is a container for ValueListItem instances that represent the values associated with a SlotType instance.

#### 3.4.1 Syntax

```xml
<complexType name="ValueListType">
  <sequence>
    <element name="ValueListItem" type="tns:ValueType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

#### 3.4.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueListItem</td>
<td>ValueType</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Element ValueListItem – This element represents a value within the collection of values in a SlotType instance. The type of this element is ValueType. Since ValueType is abstract, the actual type of ValueListItem MUST be a sub-type of ValueType. The rim.xsd schema defines the following concrete sub-types of ValueType:

- AnyValueType – This concrete sub-type of ValueType is used as a container for any well-formed XML element value in any namespace
- ParameterValueType – This concrete sub-type of ValueType is used as a container for Parameter definitions for a QueryDefinition instance
- StringValueType – This concrete sub-type of ValueType is used as a container for a string value

3.5 ExtensibleObjectType

This type is the root type for most other types in rim.xsd. It allows any type of information to be added to instances of this type using Slot sub-elements. It is an important extensibility mechanism with ebRIM.

3.5.1 Syntax

```xml
<complexType name="ExtensibleObjectType" abstract="true">
    <sequence>
        <element ref="tns:Slot" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

3.5.2 Example

The following example shows how a <rim:Organization> instance which is of type ExtensibleObjectType MAY use Slot sub-elements to define a tax payer id for the organization.

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

3.5.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot</td>
<td>Slot</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
3.6 IdentifiableObjectType

**Extends:** ExtensibleObjectType

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

### 3.6.1 Syntax

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

### 3.6.2 Example

```xml
<rim:Organization
  id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
  ...
</rim:Organization>
```

### 3.6.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute id</td>
<td>string</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute id – Specifies the unique identifier for an IdentifiableType instance. An IdentifiableType instance MUST have an id and that id MUST conform to the rules defined in section title “Unique ID Generation” in [ebRS]

3.7 RegistryObjectType

**Extends:** IdentifiableType

This type extends IdentifiableObjectType and is the common base type for all query-able metadata elements in ebRIM.

### 3.7.1 Syntax

```xml
<complexType name="RegistryObjectType">
  <complexContent>
    <extension base="tns:IdentifiableType">
      <sequence>
        <element ref="tns:Name" minOccurs="0" maxOccurs="1"/>
        <element ref="tns:Description" minOccurs="0" maxOccurs="1"/>
        <element name="VersionInfo" type="tns:VersionInfoType" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```
3.7.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Classification Type</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>International StringType</td>
<td>0..1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ExternalIdentifier</td>
<td>ExternalIdentifierType</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ExternalLink</td>
<td>ExternalLink Type</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>lid</td>
<td>string</td>
<td>0..1.</td>
<td>Client or Server</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>International StringType</td>
<td>0..1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>objectType</td>
<td>objectReferenceType</td>
<td>0..1</td>
<td>Client or Server</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>owner</td>
<td>string</td>
<td>0..1</td>
<td>Server</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>objectReferenceType</td>
<td>0..1</td>
<td>Server</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

- 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.
- 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 locales and character sets.
- Element ExternalIdentifier - A RegistryObjectType instance MAY have zero or more ExternalIdentifier instances that are composed within the RegistryObject. A ExternalIdentifier instance represents an alternate identifier for the RegistryObject in addition to the identifier specified by its id attribute value.
- Attribute lid - A RegistryObjectType instance MUST have a lid (Logical Id) attribute. The lid is used to refer to a logical RegistryObject in a version independent manner.
All versions of a RegistryObject MUST have the same value for the lid attribute. Note that this is in contrast with the id attribute that MUST be unique for each version of the same logical RegistryObject.

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

If the client assigns the lid attribute when submitting the original version of a RegistryObject, it must guarantee that it is a globally unique.

A server MUST honor a client specified LID. If the client does not specify a LID then the server MUST assign a LID and the value of the LID attribute MUST be identical to the value of the id attribute of the first (originally created) version of the logical RegistryObject. Make sure this is consistent with latest spec??

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 locales and character sets.

Attribute objectType - A RegistryObjectType instance has an objectType attribute.

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

A server MUST 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.

The objectType attribute MUST be assigned by the server for all RegistryObjectType instances that are not instances of ExtrinsicObjectType.

The objectType attribute MAY be assigned by the client for all RegistryObjectType instances that are instances of ExtrinsicObjectType. In such cases it represents the objectType associated with the repository item for the ExtrinsicObjectType instance.

A client SHOULD specify the objectType for an ExtrinsicObject during submission whenever possible.

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

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

Attribute owner – Specified the identifier associated with the registered user that own the RegistryObjectType instance. It is used for authorization of access and may be referenced within custom access control policies.

Attribute status - A RegistryObjectType instance MUST have a life cycle status indicator. The status is assigned by the server.

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

A client SHOULD NOT set the status on a RegistryObject when submitting the object as this is the responsibility of the server.
A server MUST ignore the status on a RegistryObject when it is set by the client during submission or update of the object.

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

A Registry MUST 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.

The following table lists pre-defined choices for the RegistryObject status attribute:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved</td>
<td>Indicates that the objects has been approved after being submitted</td>
</tr>
<tr>
<td>Deprecated</td>
<td>Indicates that the objects has been deprecated or marked as obsolete</td>
</tr>
<tr>
<td>Submitted</td>
<td>Indicates that the objects has been submitted to the server.</td>
</tr>
<tr>
<td>Withdrawn</td>
<td>Indicates that the objects has been withdrawn from the server. This SHOULD be used with ExtrinsicObjects when their repository item has been removed or withdrawn.</td>
</tr>
</tbody>
</table>

Element VersionInfo - Provides information about the specific version of a RegistryObject. The VersionInfo attribute is set by the server.

3.8 VersionInfoType

This type represents information about a specific version of a RegistryObject.

3.8.1 Syntax

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

3.8.2 Example

```
<rim:Organization ...>
  ...
  <rim:VersionInfo versionName="1.1" comment="Initial version"/>
  ...
</rim:Organization>
```

3.8.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>LongName</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Attribute comment - Represents a client-specified comment associated with the VersionInfo for a specific RegistryObject version. It is analogous to a commit comment in version control systems.

- The value of the comment attribute MAY be indirectly provided by the client when the client specifies a value for the comment attribute of the <rim:Request> object when making a request to the server.
- The value for this attribute MUST be set by the Registry implementation based upon the <rim:Request> comment attribute value provided by the client if any.

Attribute versionName - Represents the version name identifying the VersionInfo for a specific RegistryObject version.

- The value for this attribute SHOULD NOT be specified by the client
- A server MUST ignore the value for this attribute if specified by the client
- The value for this attribute MUST be automatically generated by the server and MUST be defined for RegistryObjectType instances returned by server responses.

3.9 objectReferenceType

**Extends:** 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.

### 3.9.1 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*.

### 3.9.2 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.

### 3.9.3 Encoding of objectReferenceType

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

- 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.

```xml
<rim:Organization primaryContact="urn:acme:person:Danyal" ...>
  ...
</rim:Organization>
<rim:Person id="urn:acme:person:Danyal" ...>
  ...
</rim:Person>
```

- 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 object identified by the lid parameter of the Query. The query when invoked matches the latest version of the Person object representing Danyal.

```xml
<rim:Organization
  primaryContact="urn:acme:dynamicRef:LatestVersionOfDanyal" ...>
  ...
</rim:Organization>
<rim:DynamicObjectRef id="urn:acme:dynamicRef:LatestVersionOfDanyal">
  <rim:Query queryDefinition="urn:acme:QueryDefinition:FindLatestVersion">
    <rim:Slot name="lid">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="rim:StringValueType">
          <rim:Value>urn:acme:person:Danyal</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
  </rim:Query>
</rim:DynamicObjectRef>
<rim:Person lid="urn:acme:person:Danyal" id="urn:acme:person:Danyal:1.8"...>
  <!-- latest version of object with lid "urn:acme:person:Danyal" -->
  ...
</rim:Person>
```

- A static or dynamic reference to a local reference target MAY be encoded to contain an HTTP GET URL for the local object as defined by the HTTP binding in [ebRS].

- A static or dynamic reference to a remote reference target MUST be encoded to contain an HTTP GET URL for the remote object as defined by the HTTP binding in [ebRS].

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
3.9.4 Syntax

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

3.10 ObjectRefType

This type represents an object reference as is 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
are needed.

3.10.1 Syntax

```xml
<complexType name="ObjectRefType">
  <complexContent>
    <extension base="tns:IdentifiableType"/>
  </complexContent>
</complexType>

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

<element name="ObjectRefList" type="tns:ObjectRefListType"/>
```

3.10.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (inherited)</td>
<td>xs:string</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Attribute id - Every ObjectRef instance MUST have an id attribute. The id attribute MUST contain the value of the id attribute of the RegistryObject being referenced.

### 3.11 DynamicObjectRefType

**Extends:** 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.

#### 3.11.1 Syntax

```xml
<complexType name="DynamicObjectRefType">
  <complexContent>
    <extension base="tns:ObjectRefType">
      <sequence>
        <element ref="tns:Query" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

#### 3.11.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>QueryType</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

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

### 3.12 ExtrinsicObjectType

**Extends:** 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.

#### 3.12.1 Syntax

```xml
<complexType name="ExtrinsicObjectType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <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:LongName" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```
3.12.2 Example

```xml
<ExtrinsicObject mimeType="text/xml"
  objectType="urn:freebxml:registry:sample:profile:cpp:objectType:cppa:CPP"
  id="urn:freebxml:registry:sample:profile:cpp:instance:cpp1"
  id="urn:freebxml:registry:sample:profile:cpp:instance:cpp1" >
  <RepositoryItem>...binary encoding of repository item</RepositoryItem>
</ExtrinsicObject>
```

3.12.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>mimeType</td>
<td>LongName</td>
<td>0..1</td>
<td>application/octet-stream</td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>RepositoryItem</td>
<td>xs:base64Binary</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>RepositoryItemRef</td>
<td>SimpleLinkType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
</tbody>
</table>

- **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.

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

- **Element repositoryItemRef** – This element MAY be specified as an alternative to the repositoryItem element. It is SimpleLinkType. It uses Xlink 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 MUST convert a repositoryItemRef element to a repositoryItem element prior to submitting it to the server.

3.13 RegistryPackageType

**Extends:** 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 RegistryObjectType instance MAY be a member of multiple RegistryPackageType instances.
- A RegistryPackageType instance MAY have multiple RegistryObjectType instances as its member.
- 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.
- 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.
### 3.13.1 Syntax

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

### 3.13.2 Example

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

```xml
<RegistryPackage id="urn:acme:RegistryPackage:photos" ...>
  ...
  <RegistryPackage id="urn:acme:RegistryPackage:photos:summer-2008">
    ...
    <RegistryObjectList>
      <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>
      </RegistryObject>
    </RegistryObjectList>
  </RegistryPackage>
</RegistryPackage>
```

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.

```xml
<RegistryPackage id="urn:acme:RegistryPackage:photos" .../>
<RegistryPackage id="urn:acme:RegistryPackage:photos:summer-2008" />
<Association
  sourceObject="urn:acme:RegistryPackage:photos"
  targetObject="urn:acme:RegistryPackage:photos:summer-2008"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"/>
<ExtrinsicObject mimeType="image/jpeg" id="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg">
  <repositoryItem>
    ...binary encoding of photo repository item
  </repositoryItem>
</ExtrinsicObject>
<Association
  sourceObject="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
  targetObject="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"/>
```
### 3.13.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegistryObjectList</td>
<td>RegistryObjectList</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Element RegistryObjectList – This element allows clients to specify members of the RegistryPackage instance using a simpler alternative to “HasMember” AssociationType instances.
  - A server MUST replace the RegistryObjectList to 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.

### 3.14 ExternalIdentifierType

**Extends:** RegistryObjectType

This type allows additional identifier 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.

#### 3.14.1 Syntax

```xml
<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:LongName" use="required"/>  
    </extension>  
  </complexContent>  
</complexType>  

<element name="ExternalIdentifier" type="tns:ExternalIdentifierType"/>
```

#### 3.14.2 Example

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

```xml
<Organization ...>  
  ...  
  <ExternalIdentifier ...  
    identificationScheme="urn:acme:ClassificationScheme:TaxPayerId"  
    value="1234567890"/>  
  </ExternalIdentifier>  
  ...  
</Organization>
```
3.14.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>identificationScheme</td>
<td>objectRefType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>registryObject</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>value</td>
<td>LongName</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute identificationScheme - Each ExternalIdentifier instance MUST 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.

- Attribute registryObject - Each ExternalIdentifier instance MAY have a registryObject attribute specified. This attribute references the parent RegistryObjectType instance for which this is an ExternalIdentifier.
  - This attribute MUST be specified when a client submits an ExternalIdentifier separately from its parent RegistryObjectType instance
  - 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 MUST set this attributes value to the value of the id attribute of the parent RegistryObjectType instance.
  - Attribute value - Each ExternalIdentifier instance MUST have a value attribute that provides the identifier value for this ExternalIdentifier (e.g., the tax payer id in example above).

3.15 ExternalLinkType

**Extends:** RegistryObjectType

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

3.15.1 Syntax

```xml
<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>
```

3.15.2 Example

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

```xml
<Organization ...>
...
```
3.15.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalRef</td>
<td>SimpleLinkType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Element ExternalRef - Each ExternalLink instance MUST have an ExternalRef sub-element defined. This element provides a URI to the external resource pointed to by this ExternalLink instance.
4 Association Information Model

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 as Association element for that type.

4.1 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.

4.2 Type of an Association

An AssociationType instance MUST have an type attribute that identifies the type of that association. The value of this attribute MUST be the id of a ClassificationNode under the canonical AssociationType ClassificationScheme.

4.3 AssociationType

Extends: RegistryObjectType

4.3.1 Syntax

```xml
<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>

<element name="Association" type="tns:AssociationType"/>
```

4.3.2 Example

The following examples shows an Organization instance that has an “OffersService” association with a Service that it offers.

```xml
<Organization ... id="urn:acme:Organization:acme-inc" ... />
<Service ... id="urn:acme:Service:stock-quote" ... />
<Association 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" .../>
```
### 4.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceObject</td>
<td>objectRefType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>targetObject</td>
<td>objectRefType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>type</td>
<td>objectRefType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute `sourceObject` - Each Association MUST have a `sourceObject` attribute that references the RegistryObjectType instance that is the source of that Association.
- Attribute `targetObject` - Each Association MUST have a `targetObject` attribute that references the RegistryObjectType instance that is the target of that Association.
- Attribute `type` - Each Association MUST have a `type` attribute that identifies the type of that association.
  - The value of the `type` attribute MUST be a reference to a ClassificationNode within the canonical AssociationType ClassificationScheme.
  - A server MUST 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.

### 4.4 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.
5 Classification Information Model

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 extend 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.

Illustration 3 Below shows RegistryObjectType instances representing Organizations as shaded boxes. Each Organization represents an automobile manufacturer. Organization is classified by the ClassificationNode named “Automotive” under the ClassificationScheme instance with name “Industry.” Furthermore, the US Automobile manufacturers are classified by the “US” ClassificationNode under the ClassificationScheme with name “Geography.” Similarly, a European automobile manufacturer is classified by the “Europe” ClassificationNode under the ClassificationScheme with name “Geography.”

The example shows how a RegistryObject may be classified by multiple ClassificationNodeType instances under multiple ClassificationScheme instances (e.g., Industry, Geography).
Illustration 3: Classification Example

Illustration 4 Shows the Classification information model.

Illustration 4: Classification Information Model
5.1 TaxonomyElementType

Extends: RegistryObjectType

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

Extends: RegistryObjectType

5.1.1 Syntax

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

5.1.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Node</td>
<td>ClassificationNodeType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

5.2 ClassificationSchemeType

Extends: 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.

5.2.1 Syntax

```xml
<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>
```
5.2.2 Example

The following examples shows a ClassificationScheme to represent gender values.

```
<ClassificationScheme id="urn:acme:GenderScheme" isInternal="true"
    nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode" />
    <Name>
        <LocalizedString charset="UTF-8" value="GenderScheme"/>
    </Name>
    <ClassificationNode id="urn:acme:Gender:Male" code="Male" />
    <ClassificationNode id="urn:acme:Gender:Female" code="Female" />
    <ClassificationNode id="urn:acme:Gender:Female" code="Other" />
</ClassificationScheme>
```

5.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>isInternal</td>
<td>xs:boolean</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>nodeType</td>
<td>objectRefType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Attribute isInternal** - When submitting a ClassificationSchemeType instance the client MUST 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.

- **Attribute nodeType** - When submitting a ClassificationScheme instance the Submitting Organization MUST declare the structure of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The value of the nodeType attribute MUST be a reference to a ClassificationNodeType instance within the canonical NodeType ClassificationScheme. A server MUST 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.

The following table lists the canonical ClassificationNode defined as values for the NodeType ClassificationScheme:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniqueCode</td>
<td>Indicates that the code for each ClassificationNode in the ClassificationScheme is unique within the scope of the ClassificationScheme</td>
</tr>
<tr>
<td>EmbeddedPath</td>
<td>Indicates that the code assigned to each node of the taxonomy also encodes its path.</td>
</tr>
<tr>
<td>NonUniqueCode</td>
<td>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.</td>
</tr>
</tbody>
</table>
5.3 ClassificationNodeType

Extends: TaxonomyElementType

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

5.3.1 Syntax

```
<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:LongName" use="required"/>
    </extension>
  </complexContent>
</complexType>

<element name="ClassificationNode" type="tns:ClassificationNodeType"/>
```

5.3.2 Example

The following examples shows a ClassificationScheme to represent gender values.

```
<ClassificationScheme id="urn:acme:GenderScheme" ...>
  ...
  <ClassificationNode id="urn:acme:Gender:Male" code="Male" ...>
    <Name>
      <LocalizedString charset="UTF-8" value="Male"/>
    </Name>
  </ClassificationNode>
</ClassificationScheme>
```

5.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>LongName</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>0..1</td>
<td></td>
<td>Registry</td>
<td>No</td>
</tr>
</tbody>
</table>

- Attribute code - A ClassificationNodeType instance MUST have a code attribute. The code attribute contains a code that represents a value within a ClassificationScheme.
  - The code attribute of a ClassificationNodeType instance MUST be unique with respect to all sibling ClassificationNodes that are immediate children of the same parent TaxonomyElementType instance.
- Attribute parent - A ClassificationNodeType instance MAY have a parent attribute. The parent attribute either references the parent TaxonomyElementType instance for the ClassificationNodeType instance.
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 5.3.4.

- A server MUST set the path attribute for any ClassificationNodeType instance when it is submitted by a client.
- The path attribute MUST be ignored by the server if it is specified by the client during the submission of the ClassificationNodeType instance.
- The path attribute of a ClassificationNode MUST be unique within a server.

### 5.3.4 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.

#### Example of Canonical Path Representation

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

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

### 5.4 ClassificationType

**Extends:** 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.

#### 5.4.1 Syntax

```xml
<complexType name="ClassificationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="classificationScheme" type="tns:objectReferenceType" use="optional"/>
      <attribute name="classifiedObject"/>
    </extension>
  </complexContent>
</complexType>
```
5.4.2 Example

The following examples shows how a Person instance is classified using the sample Gender ClassificationScheme used in earlier examples.

```
<Person id="urn:acme:person:Danyal" ...>
  ...
  <Classification classifiedObject="urn:acme:person:Danyal"
    classificationNode="urn:acme:Gender:Male"
    ...
</Person>
```

5.4.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>classificationNode</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>classifiedObject</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>classificationScheme</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>nodeRepresentation</td>
<td>LongName</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
</tbody>
</table>

- Attribute classificationNode - If the ClassificationType instance represents an internal classification, then the `classificationNode` attribute is required.
  - The `classificationNode` value MUST reference a ClassificationNodeType instance.
- 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.
- Attribute classificationScheme - If the ClassificationType instance represents an external classification, then the `classificationScheme` attribute is required.
  - The `classificationScheme` value MUST reference a ClassificationScheme instance.
- 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.
6 Provenance Information Model

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 provenance of RegistryObjectType instances.

6.1 PersonType

Extends: RegistryObjectType

This type represents a person.

6.1.1 Syntax

```xml
<complexType name="PersonType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element ref="tns:PersonName" minOccurs="0" maxOccurs="1"/>
        <element ref="tns:PostalAddress" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.1.2 Example

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

6.1.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmailAddress</td>
<td>EmailAddressType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>PersonName</td>
<td>PersonNameType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>PostalAddress</td>
<td>PostalAddressType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>TelephoneNumberType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>
● Element EmailAddress - A PersonType instance MAY have any number of EmailAddress sub-elements. Each EmailAddress provides an email address for that person. A Person SHOULD have at least one EmailAddress.

● Element PersonName – A PersonType instance SHOULD have a personName sub-element that provides the name for that person.

● Element PostalAddress - A PersonType instance MAY have any number of PostalAddress sub-elements. Each PostalAddress element provides a postal address for that person. A PersonType instance SHOULD have at least one PostalAddress.

● Element TelephoneNumber - A PersonType instance MAY have any number of TelephoneNumber sub-elements. Each TelephoneNumber element provides a TelephoneNumber for that person. A Person SHOULD have at least one TelephoneNumber.

### 6.2 PersonNameType

**Extends:** ExtensibleObjectType

This represents the name for a PersonType instance.

#### 6.2.1 Syntax

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

#### 6.2.2 Example

```xml
<Person id="urn:acme:person:Danyal" ...>
  ...
  <PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
  ...
</Person>
```

#### 6.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstName</td>
<td>ShortName</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>lastName</td>
<td>ShortName</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>middleName</td>
<td>ShortName</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute firstName - A PersonName instance SHOULD have a firstName attribute that is the given name of the person.
- Attribute lastName - A PersonName instance SHOULD have a lastName attribute that is the family name of the person.
• Attribute middleName - A PersonName instance SHOULD have a middleName attribute that is the middle name of the person.

6.3 PostalAddressType

Extends: ExtensibleObjectType

This type represents a postal or mailing address.

6.3.1 Syntax

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

<element name="PostalAddress" type="tns:PostalAddressType"/>
```

6.3.2 Example

```xml
<Person id="urn:acme:person:Danyal" ...>
  ...
  <PostalAddress streetNumber="10" street="Street 1" city="Islamabad"
    stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
  ...
</Person>
```

6.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>city</td>
<td>ShortName</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>country</td>
<td>ShortName</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>postalCode</td>
<td>ShortName</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>stateOrProvince</td>
<td>ShortName</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>street</td>
<td>ShortName</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>streetNumber</td>
<td>String32</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Attribute city - A PostalAddressType instance MAY have a city attribute identifying the city for that address.

• Attribute country - A PostalAddressType instance MAY have a country attribute identifying the country for that address.

• Attribute postalCode - A PostalAddressType instance MAY have a postalCode attribute identifying the postal code (e.g., zip code) for that address.
• Attribute stateOrProvince - A PostalAddressType instance MAY have a stateOrProvince attribute identifying the state, province or region for that address.

• Attribute street - A PostalAddressType instance MAY have a street attribute identifying the street name for that address.

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

6.4 TelephoneNumberType

Extends: ExtensibleObjectType

This class defines attributes of a telephone number.

6.4.1 Syntax

<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>

6.4.2 Example

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

6.4.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>areaCode</td>
<td>String8</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>countryCode</td>
<td>String8</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>extension</td>
<td>String8</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>number</td>
<td>String16</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>type</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Attribute areaCode - A TelephoneNumberType instance MAY have an areaCode attribute that provides the area code for that telephone number.

• Attribute countryCode - A TelephoneNumberType instance MAY have a countryCode attribute that provides the country code for that telephone number.
● Attribute extension - A TelephoneNumberType instance MAY have an extension attribute that provides the extension number, if any, for that telephone number.

● 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.

● Attribute type - A TelephoneNumberType instance MAY have a type attribute that provides the type for the TelephoneNumber. The value of the type attribute MUST be a reference to a ClassificationNode in the canonical PhoneType ClassificationScheme.

6.5 EmailAddressType

Extends: ExtensibleObjectType

This class defines attributes of an email address.

6.5.1 Syntax

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

6.5.2 Example

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

6.5.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>ShortName</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>type</td>
<td>objectRefType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

● Attribute address - An EmailAddressType instance MUST have an address attribute that provides the actual email address.

● Attribute type - An EmailAddressType instance MAY have a type attribute that provides the type for that email address. The value of the type attribute MUST be a reference to a ClassificationNode in the canonical EmailType ClassificationScheme.

6.6 OrganizationType

Extends: RegistryObjectType
This type represents an organization or entity.

### 6.6.1 Syntax

```xml
<complexType name="OrganizationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element ref="tns:PostalAddress" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="tns:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="primaryContact" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

### 6.6.2 Example

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

### 6.6.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmailAddress</td>
<td>EmailAddressType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>PostalAddress</td>
<td>PostalAddressType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>primaryContact</td>
<td>objectRefType</td>
<td>No</td>
<td></td>
<td>Client</td>
<td>No</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>TelephoneNumberType</td>
<td>0..*</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **EmailAddress** - An `OrganizationType` instance MAY have any number of `EmailAddress` sub-elements. Each `EmailAddress` provides an email address for that person. A `OrganizationType` SHOULD have at least one `EmailAddress`.

- **PostalAddress** - An `OrganizationType` instance MAY have any number of `PostalAddress` sub-elements. Each `PostalAddress` element provides a postal address for that person. A `OrganizationType` instance SHOULD have at least one `PostalAddress`.

- **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.
6.7 Associating Organization With Persons

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

- The type attribute of the Association MAY reference the canonical ClassificationNode with id "urn:oasis:names:tc:ebxml-regrep:AssociationType:AffiliatedWith" or one of its descendants.
- The sourceObject SHOULD reference the child PersonType instance.
- The targetObject SHOULD reference the OrganizationType instance.

6.8 Associating Organization With Organizations

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

- 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:HasParent" or one of its descendants.
- The sourceObject SHOULD reference the child OrganizationType instance.
- The targetObject SHOULD reference the parent OrganizationType instance.

6.9 Associating Organizations With RegistryObjects

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 chapter.

6.9.1 ResponsibleFor Relationships

An organization that is the authoritative source for a RegistryObjectType instance is referred to as the Responsible Organization for that RegistryObjectType instance. The term Responsible Organization has its origins in [11179-6].

- A RegistryObjectType instance SHOULD be related to its responsible organization using the canonical AssociationType with id "urn:oasis:names:tc:ebxml-regrep:AssociationType:ResponsibleFor".
- The sourceObject SHOULD reference the OrganizationType instance for the Responsible Organization.
- The targetObject SHOULD reference the RegistryObjectType instance.
6.9.2 **SubmitterOf Relationships**

An organization that has submitted a RegistryObjectType instance on behalf of a Responsible Organization is referred to as the *Submitting Organization* for that RegistryObjectType instance. The term *Submitting Organization* has its origins in [11179-6].

- A RegistryObjectType instance SHOULD be related to its submitting organization using the canonical AssociationType with id "urn:oasis:names:tc:ebxml-regrep:AssociationType:SubmitterOf".
- The sourceObject SHOULD reference the OrganizationType instance for the Submitting Organization.
- The targetObject SHOULD reference the RegistryObjectType instance.

6.9.3

*Illustration 5: Organization to RegistryObject Association*
7 Service Information Model

This chapter describes the parts of the information model that support the description of services within an ebXML RegRep server. Although service information model aligns with [WSDL2] model, it may be used to describe any type of service in addition to web services.

7.1 ServiceType

Extends: 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.

7.1.1 Syntax

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

```xml
<Service id="urn:acme:Service:StockQuoteService" ...>
  ...
  <ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:free" .../>
  <ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:premium" .../>
</Service>
```

### 7.1.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceEndpoint</td>
<td>ServiceEndpointType</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>serviceInterface</td>
<td>objectReferenceType</td>
<td>0..1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

- Element ServiceEndpoint – Represents a physical endpoint for the service that MAY be used by clients to access the service
- Attribute serviceInterface – References the abstract interface description for the service
  - MUST reference a ServiceInterfaceType instance if specified

### 7.2 ServiceEndPointType

**Extends:** RegistryObjectType

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

#### 7.2.1 Syntax

```xml
<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>
```

#### 7.2.2 Example

```xml
<Service id="urn:acme:Service:StockQuoteService" ...>
  ...
  <ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:free"
    address="http://acme.com/StockQuoteService/free"
    serviceBinding="urn:acme:ServiceBinding:soap:StockQuoteService"/>
</Service>
```

#### 7.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
</table>

regrep-rim  
Copyright © OASIS® 2008. All Rights Reserved.
Attribute address – Represents the endpoint address URI that a client of the service endpoint may use to access the service endpoint

Attribute serviceBinding – References the ServiceBindingType instance that represents a protocol-specific binding information for the ServiceEndpointType instance

○ MUST reference a ServiceBindingType instance

7.3 ServiceBindingType

Extends: RegistryObjectType

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

7.3.1 Syntax

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

7.3.2 Example

```xml
<ServiceBinding id="urn:acme:ServiceBinding:soap:StockQuoteService"
    serviceInterface="urn:acme:ServiceInterface:StockQuoteService" .../>
```

7.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceInterface</td>
<td>objectReferenceType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Attribute serviceInterface – References a ServiceInterfaceType instance which represents the abstract service interface for the service

○ MUST reference a ServiceInterfaceType instance if specified
7.4 Service Interface Type

Extends: RegistryObjectType

This type represents an abstract service interface for a service.

7.4.1 Syntax

```xml
<complexType name="ServiceInterfaceType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <element name="ServiceInterface" type="tns:ServiceInterfaceType"/>
    </extension>
  </complexContent>
</complexType>
```

7.4.2 Example

```xml
<ServiceInterface id="urn:acme:ServiceInterface:StockQuoteService" .../>
.../
</ServiceInterface>
```

7.4.3 Description

No attributes or elements beyond those inherited from RegistryObjectType are defined for this type.
This chapter describes the information model for defining and invoking parameterized queries in ebXML RegRep. The following significant types are defined by the Query Information Model:

- **QueryDefinitionType** - Represents the definition of a parameterized query
- **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 as QueryDefinitionType instances as canonical queries for that profile. Deployments MAY also define additional as QueryDefinitionType instances. Finally, clients MAY submit additional as QueryDefinitionType instances.

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

### 8.1 QueryDefinitionType

**Extends:** 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.

#### 8.1.1 Syntax

```xml
<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>
```

#### 8.1.2 Example

```xml
<QueryDefinition id="urn:oasis:names:tc:ebxml-regrep:query:FindObjectByIdAndType">
  <Parameter parameterName="id" datatype="string" minOccurs="1" maxOccurs="1" defaultValue="%">\n    </Parameter>
  <Parameter parameterName="type" datatype="taxonomyElement" minOccurs="1" maxOccurs="1">\n    </Parameter>
  <QueryExpression xsi:type="rim:StringQueryExpressionType">
    SELECT Object(ro) FROM ...
  </QueryExpression>
</QueryDefinition>
```
8.1.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>ParameterType</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>QueryExpression</td>
<td>QueryExpressionType</td>
<td>0..*</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

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

8.2 ParameterType

Extends: ExtensibleObjectType

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

8.2.1 Syntax

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

8.2.2 Example

```xml
<QueryDefinition id="urn:oasis:names:tc:ebxml-regrep:query:FindObjectByIdAndType">
  <Parameter parameterName="id" datatype="string" minOccurs="1" maxOccurs="1" defaultVALUE="%">
    ...
  </Parameter>
  <QueryExpression .../>
</QueryDefinition>
```

8.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Data Type</td>
<td>Value</td>
<td>Client</td>
<td>Is Mandatory</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
<td>-------</td>
<td>--------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>dataType</td>
<td>xs:string</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>defaultValue</td>
<td>xs:string</td>
<td>0..1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>InternationalStringType</td>
<td>0..1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>minOccurs</td>
<td>xs:nonNegativeInteger</td>
<td>0..1</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>maxOccurs</td>
<td>xs:nonNegativeInteger</td>
<td>0..1</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>InternationalStringType</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>parameterName</td>
<td>xs:string</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

- **Attribute dataType** – Specifies the data type for the parameter. *Need more here??*
- **Attribute defaultValue** - Specifies the default value for the parameter. This value MUST be used as parameter value when the query is invoked if the client does not specify a value for this parameter.
- **Element Description** - Specifies a human-friendly description of the parameter that indicates the what the parameter value represents and what kind of value is expected. The description MAY be provided in multiple locales and character sets.
- **Attribute minOccurs** – Specifies the minimum number of values expected for the parameter.
- **Attribute maxOccurs** - Specifies the maximum number of values expected for the parameter.
- **Element Name** - Specifies a human-friendly name for the parameter. The name MAY be provided in multiple locales and character sets.
- **Attribute parameterName** – Specifies the canonical name of the parameter. The canonicalName identifies the parameter in a locale-insensitive manner
  - SHOULD match a declared parameter name within the query expression for the QueryDefinitionType instance

### 8.3 QueryExpressionType

**Extends:** 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:

- **StringQueryExpressionType** – This type MAY be used to represent non-XML query syntaxes such as SQL-92 and EJBQL.
- **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.

### 8.3.1 Syntax

```xml
<complexType name="QueryExpressionType" abstract="true">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <extension base="tns:ExtensibleObjectType">
```

---

Copyright © OASIS® 2008. All Rights Reserved.
8.3.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryLanguage</td>
<td>objectReferenceType</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

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

8.4 StringQueryExpressionType

Extends: QueryExpressionType

This type represents a non-XML query syntaxes such as SQL-92 and EJQL.

8.4.1 Syntax

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

8.4.2 Example

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

8.4.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
</table>

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.

8.5 XMLQueryExpressionType

Extends: QueryExpressionType

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

8.5.1 Syntax

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

8.5.2 Example

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

8.5.3 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.

8.6 QueryType

Extends: ExtensibleObjectType

This type represents the invocation of a parameterized query.

8.6.1 Syntax

```
<complexType name="QueryType">
```

8.6.2 Example

```xml
<Query queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:FindObjectByIdAndType">
  <Slot name="id">
    <ValueList>
      <ValueListItem xsi:type="rim:StringValue"/>
      <Value>urn:acme:person:Danyal</Value>
    </ValueList>
  </Slot>
</Query>
```

8.6.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryDefinition</td>
<td>objectReferenceType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute `queryDefinition` – References the parameterized query to be invoked by the server.
  - The value of this attribute MUST be a reference to a QueryDefinitionType instance that is supported by the server.
- Element Slot (Inherited) - Each Slot element specifies a parameter value for a parameter supported by the QueryDefinitionType instance.
  - The slot name MUST match a parameterName attribute within a Parameter's definition within the QueryDefinitionType instance.
  - The slot value's type MUST match the dataType attribute for the Parameter's definition within the QueryDefinitionType instance.
  - A server MUST NOT treat the order of parameters as significant.
9 Event Information Model

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

- **AuditableEventType** – Represents a server event that is typically a consequence of a client request.
- **SubscriptionType** – Represents a client's subscription to receive notification of AuditableEventType instances based upon a specified selection criteria.
- **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.
- **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 1 shows how a Subscription may be defined that uses a pre-configured AdhocQuery instance as a selector to select the AuditableEvents of interest to the subscriber and one or more Actions to deliver the selected events to the subscriber. The Action may deliver the events by using its endPoint attribute to invoke a registered ServiceBinding to a registered Service or by sending the events to an email address.

![Event Information Model Diagram]

9.1 AuditableEventType

**Extends:** RegistryObjectType

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

AuditableEventType instances represent a change in the life cycle of a RegistryObjectType instance. For example a client request could Create, Update, Deprecate or Delete a RegistryObjectType instance. An AuditableEvent is are created when a request creates or alters the state of a RegistryObjectType instance. Read-only requests typically do not generate an AuditableEventType instance.
9.1.1 Syntax

```xml
<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="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>

<element name="AuditableEvent" type="tns:AuditableEventType"/>
```

9.1.2 Example

The following example shows an AuditableEventType instance that logs the creation of two objects within the context of a client request.

```xml
<AuditableEvent requestid="urn:uuid:24cee176-9098-4931-894f-fea5dab1732a"
                timestamp="2008-01-10T19:20:30+01:00" user="123456">
  ...>
  <Action eventType="urn:oasis:names:tc:ebxml-regrep:EventType:Created">
    <AffectedObjects>
      <ObjectRef id="urn:acme:person:Danyal" />
      <ObjectRef id="urn:acme:person:Danyal" />
    </AffectedObjects>
  </Action>
</AuditableEvent>
```

9.1.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>ActionType</td>
<td>1..*</td>
<td></td>
<td>Registry</td>
<td>No</td>
</tr>
<tr>
<td>requestId</td>
<td>URI</td>
<td>1</td>
<td></td>
<td>Registry</td>
<td>No</td>
</tr>
<tr>
<td>timeStamp</td>
<td>xs:dateTime</td>
<td>1</td>
<td></td>
<td>Registry</td>
<td>No</td>
</tr>
<tr>
<td>user</td>
<td>objectReferenceType</td>
<td>1</td>
<td></td>
<td>Registry</td>
<td>No</td>
</tr>
</tbody>
</table>

- Element Action – Represents an action taken by the server within the context of an AuditableEventType instance. An AuditableEventType instance MAY have one or more Action instances.
- Attribute requestId – Specifies the id of the request that generated the AuditableEventType instance.
- Attribute timestamp – Specifies the timestamp that records the date and time the event occurred.
- 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.
9.2 ActionType

Represents an action taken by the server within the context of an AuditableEventType instance.

9.2.1 Syntax

```xml
<complexType name="ActionType">
  <sequence>
    <element name="AffectedObjects" type="tns:ObjectRefListType" minOccurs="1" maxOccurs="1"/>
  </sequence>
  <attribute name="eventType" type="tns:objectReferenceType" use="required"/>
</complexType>
```

9.2.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffectedObjects</td>
<td>ObjectRefListType</td>
<td>1..*</td>
<td>Registry</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>eventType</td>
<td>URI</td>
<td>1</td>
<td>Registry</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

- Element AffectedObject – 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.

- Attribute eventType – Specifies the type of event associated with the Action within an AuditableEventType instance.
  - The value of the eventType attribute MUST be a reference to a ClassificationNode in the canonical EventType ClassificationScheme.
  - A Registry MUST support the event types as defined by the EventType ClassificationScheme.
  - The canonical EventType ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it.

The following table lists pre-defined auditable event types:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved</td>
<td>An Event that marks the approval of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Created</td>
<td>An Event that marks the creation of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Deleted</td>
<td>An Event that marks the deletion of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Deprecated</td>
<td>An Event that marks the deprecation of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Downloaded</td>
<td>An Event that marks the downloading of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Relocated</td>
<td>An Event that marks the relocation of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Undeprecated</td>
<td>An Event that marks the undeprecation of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Updated</td>
<td>An Event that marks the updating of a RegistryObjectType instance.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Versioned</td>
<td>An Event that that marks the creation of a new version of a RegistryObjectType instance.</td>
</tr>
</tbody>
</table>

### 9.3 SubscriptionType

**Extends:** 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.

#### 9.3.1 Syntax

```xml
<complexType name="SubscriptionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="DeliveryInfo" type="tns:DeliveryInfoType" minOccurs="1" maxOccurs="1" />
        <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" default="P1D"/>
    </extension>
  </complexContent>
</complexType>
```

#### 9.3.2 Example

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

```xml
<Subscription id="urn:acme:Subscription:subscribeToDanyal">
  <startTime>2008-01-10T19:20:30+01:00</startTime>
  <endTime>2009-01-10T19:20:30+01:00</endTime>
  <DeliveryInfo deliveryMode="Pull">
    <NotifyTo>
      http://www.acme.com/notificationListener
    </NotifyTo>
  </DeliveryInfo>
  <Selector queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:FindObjectByIdAndType">
    <Slot name="id">
      <ValueList>
        <Value xsi:type="rim:StringValueType">
          urn:acme:person:Danyal
        </Value>
      </ValueList>
    </Slot>
  </Selector>
</Subscription>
```
9.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliverInfo</td>
<td>DeliveryInfoType</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>endTime</td>
<td>xs:dateTime</td>
<td>1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>notificationInterval</td>
<td>xs:duration</td>
<td>0..1</td>
<td>P1D (once a day)</td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>Selector</td>
<td>QueryType</td>
<td>0..1</td>
<td></td>
<td>Client</td>
<td>Yes</td>
</tr>
<tr>
<td>startTime</td>
<td>xs:dateTime</td>
<td>0..1</td>
<td>Time of submission</td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Attribute startTime, endTime – Define the time window within which the subscription is valid.
  - The Subscription validity window MUST be inclusive of the startTime and endTime
- 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.
  - A server MUST deliver notifications that match the Selector query for a valid SubscriptionType instance to the endpoint specified by the DeliveryInfo element of the SubscriptionType instance
- Attribute notificationInterval – Specifies the duration that a server MUST 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.
  - A server MUST deliver any pending notifications within the interval specified by this attribute.
- Element Selector – Specifies the query that the server MUST 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 then the event matches the subscription.
  - A server MUST deliver notifications for a valid SubscriptionType instance if the selector query for it matches an object that was affected by the event.

9.4 DeliveryInfoType

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:

- SOAP service endpoint
- REST service endpoint
- E-mail address endpoint

9.4.1 Syntax

```xml
<complexType name="DeliveryInfoType">
<complexContent>
<extension base="tns:ExtensibleObjectType">
```

```xml
<complexType name="DeliveryInfoType">
<complexContent>
<extension base="tns:ExtensibleObjectType">
```

October 15, 2008
9.4.2 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryMode</td>
<td>string</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NotifyTo</td>
<td>wsa:EndpointReferenceType</td>
<td>1</td>
<td>Client</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

- Attribute deliveryMode – Specifies the modality of how notifications are to be delivered to the subscriber.
  - Push – Indicates that the server MUST provide complete RegistryObjectType instances in notifications delivered to the subscriber when this mode is specified.
  - Pull - Indicates that the server MUST 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.
  - None – Indicates that the server MUST not deliver any notification for the subscription to the client. This mode may be used to suspend delivery of notifications temporarily.

- Element NotifyTo – Specifies the endpoint reference for the endpoint where the server should deliver notifications for the Subscription.
  - The type of this element is defined by [WSA-Core]
  - If endpoint is a SOAP or REST web service then the endpoint reference MUST be a URL
  - If endpoint is an email address then the endpoint reference MUST be a URL with protocol prefix of "mailto:" and specify the email address for the remainder of the URL (e.g. mailto:danyal@home.com)

9.5 NotificationType

Extends: 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. A server MAY batch notifications for a subscription to optimize delivery of notifications as long as pending notifications are delivered within a subscription's specified notificationInterval. The Notification message contains a list of ObjectRefType instances if the subscription's deliveryMode is "Pull". The Notification message contains a list of complete
RegistryObjectType instances if the subscription's deliveryMode is “Push”. A Notification does not include information on AuditableEventType instances. A client MAY get access to AuditableEventType instances for a RegistryObjectType instance using the Query protocol if needed.

### 9.5.1 Syntax

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

### 9.5.2 Example

The following example shows 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.

```xml
<Notification subscription="urn:acme:Subscription:subscribeToDanyal" ...>
  <IdentifiableList>
    <ObjectRef id="urn:acme:person:Danyal" />  
  </IdentifiableList>
</Notification>
```

### 9.5.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IdentifiableList</td>
<td>IdentifiableListType</td>
<td>1</td>
<td>Server</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>subscription</td>
<td>objectReferenceType</td>
<td>1</td>
<td>Server</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

- Element IdentifiableList – List of IdentifiableType instances containing the RegistryObjectType instances if deliveryMode was “Push”, or ObjectRefType instances if deliveryMode was “Pull”.
- Attribute subscription – References the SubscriptionType instance for which this is a Notification.
This chapter describes the information model that support 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.

### 10.1 Federation Configuration

A federation is created by the creation of a FederationType instance. Membership of a registry within a federation is established by creating an Association between the RegistryType instance for the registry seeking membership and the FederationType instance. The Association MUST have its associationType be the id of the canonical ClassificationNode "HasFederationMember", the federation instance as its sourceObject and the Registry instance as its targetObject as shown in the illustration:

![Federation Information Model](image)

### 10.2 RegistryType

**Super Classes:** RegistryObject

Registry instances are used to represent a ebXML RegRep server.

#### 10.2.1 Syntax

```xml
<complexType name="RegistryType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="operator" />
    </extension>
  </complexContent>
</complexType>
```
10.2.2 Example

The following example describes 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 the ebXML RegRep specifications. The server performs replication synchronization once a day (P1D) and performs cataloging of submitted content immediately when content is submitted.

```xml
<Registry id="urn:acme:Registry:serviceRegistry"
    operator="urn:acme:Organization:acme-inc"
    specificationVersion="4.0"
    conformanceProfile="registryFull"
    replicationSyncLatency="P1D"
    catalogingLatency="PT0S"
    ...
    ...
</Registry>
```

10.2.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalogingLatency</td>
<td>xs:duration</td>
<td>0..1</td>
<td>P1D (once a day)</td>
<td>Server</td>
<td>Yes</td>
</tr>
<tr>
<td>conformanceProfile</td>
<td>objectReferenceType</td>
<td>1</td>
<td></td>
<td>Server</td>
<td>Yes</td>
</tr>
<tr>
<td>operator</td>
<td>objectReferenceType</td>
<td>1</td>
<td></td>
<td>Server</td>
<td>Yes</td>
</tr>
<tr>
<td>replicationSyncLatency</td>
<td>xs:duration</td>
<td>0..1</td>
<td>PT0S (immediately)</td>
<td>Server</td>
<td>Yes</td>
</tr>
<tr>
<td>specificationVersion</td>
<td>objectReferenceType</td>
<td>1</td>
<td></td>
<td>Server</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- 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.
• 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 [ebRS].

• Attribute operator - A RegistryType instance MUST 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.

• 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.

• Attribute specificationVersion - A RegistryType instance MUST have an attribute named
  specificationVersion that is the version of the ebXML RegRep Specifications it implements.

10.3 FederationType

Extends: ExtensibleObjectType

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.

10.3.1 Syntax

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

10.3.2 Example

The following example shows a Federation with two independently-operated ebXML RegRep servers as
members.

```xml
<Federation id="urn:acme:Federation:supplierFederation"
  replicationSyncLatency="P1D" ...>
  ...
</Federation>

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

<Association
  sourceObject="urn:acme:Federation:supplierFederation"
  targetObject="urn:supplierInc:Registry:supplier-inc"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>
```
### 10.3.3 Description

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Cardinality</th>
<th>Default Value</th>
<th>Specified By</th>
<th>Mutable</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicationSyncLatency</td>
<td>xs:duration</td>
<td>No</td>
<td>P1D (1 day)</td>
<td>Client</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- 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 MUST synchronize itself with the current state of the Federation. Members of the Federation MAY use this parameter to periodically synchronize the federation metadata they MUST cache locally about the state of the Federation and its members. Such synchronization MAY be based upon the registry event notification capability.
11 Access Control Information Model

This chapter defines the Information Model used to control access to RegistryObjects and RepositoryItems managed by it. It also defines a normative profile of [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 MUST support the roles of both Enforcement Point (PEP) and a Policy Decision Point (PDP) as defined in [XACML].

The Access Control Model attempts to reuse terms defined by [XACML] wherever possible. The definitions of some key terms are duplicated here from [XACML] for convenience of the reader:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Performing an action. An example is a user performing a delete action on a RegistryObject.</td>
</tr>
<tr>
<td>Access Control</td>
<td>Controlling access in accordance with a policy. An example is preventing a user from performing a delete action on a RegistryObject that is not owned by that user.</td>
</tr>
<tr>
<td>Action</td>
<td>An operation on a resource. An example is the delete action on a RegistryObject.</td>
</tr>
</tbody>
</table>
| Attribute    | Characteristic of a subject, resource, action. Some examples are:  
• id attribute of a subject  
• role attribute of a subject  
• 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 RegistryObject resource  
• A RepositoryItem resource |
| Subject      | An actor whose attributes may be referenced by within a Policy definition. Example:                                                       |
• The registered user associated with a client request |
11.1 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 [ebRS].

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

11.2 Assigning Access Control Policy to a RegistryObject

An Access Control Policy MAY be assigned to a RegistryObjectType instance using a special Association with the canonical associationType of “AccessControlPolicyFor” as defined in the canonical AssociationTypeScheme ClassificationScheme. This association references the ExtrinsicObject representing the Access Control Policy via its sourceObject attribute and references the RegistryObjectType instance via its targetObject attribute.

If a RegistryObjectType instance does not have an Access Control Policy explicitly associated with it, then it is implicitly associated with the default Access Control Policy defined for the server.

Illustration 8: Assigning Access Control Policy to a RegistryObject

Illustration 8 shows a UML instance diagram where an Organization instance org is associated with an ExtrinsicObject instance accessControlPolicy as its Access Control Policy object using an Association of type “AccessControlPolicyFor”.

Illustration 8: Assigning Access Control Policy to a RegistryObject
11.3 Default Access Control Policy

A server MUST 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:

- Only a Registered user is granted access to actions that modify the state of any resource.
- An unauthenticated client is granted access to read actions that do not modify the state of any resource.
- A server MUST assign the default RegistryGuest role to the identity associated with an unauthenticated client.
- A registered user has access to all actions on Registry Objects submitted by the user.
- The Registry Administrator has access to all actions on all Registry Objects.

11.4 Root Access Control Policy

A server SHOULD have a root Access Control Policy that bootstraps the Access Control Model by controlling access to Access Control Policies.

As described in earlier, an access control policy is an ExtrinsicObject that contains a pointer to a repository item. The lifecycle of access control policies is managed using the standard protocols defined by the LifeCycleManager interface defined by [ebRS].

To define who may perform lifecycle management operations on access control policies pertaining to specified resources, it is necessary to have one or more administrative Access Control Policies. Such policies restrict clients from managing access control policies for resources they are not authorized for.

This version of the Registry specifications defines a single Root Access Control Policy that allows all actions on Access Control Policies for a resource if one of the following conditions is met:

- Subject is the owner of the resource
- Subject has a role of RegistryAdministrator

11.5 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 judgement on when to reuse or extend an existing Access Control Policy and when to create a new one.

11.6 Action Matching

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

The following table specifies the actions that a server MUST support as valid values for identifying an action within an XACML file. The supported values are listed in the "Action ID" column. A server MUST
specify the action identifier in an `<xacmlc:Request>` using the standard action attribute named "urn:oasis:names:tc:xacml:1.0:action:action-id".

<table>
<thead>
<tr>
<th>Action ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being newly created by a submitObjects operation.</td>
</tr>
<tr>
<td>Read</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being read by a executeQuery operation.</td>
</tr>
<tr>
<td>Replaced</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being replaced by a submitObjects operation.</td>
</tr>
<tr>
<td>Update</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being updated by an updateObjects operation.</td>
</tr>
<tr>
<td>Delete</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being deleted by a removeObjects operation.</td>
</tr>
<tr>
<td>Reference</td>
<td>A server MUST specify this as value for action-id attribute in an <code>&lt;xacmlc:Request&gt;</code> to a PDP if the resource is being referenced by another resource within an submitObjects or updateObjects operation.</td>
</tr>
</tbody>
</table>

11.6.1 Action Attribute: `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 MUST be "urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source". The value of this attribute MUST be the value of the id attribute for the object that is the source of the reference. A server MUST specify this attribute for a reference action.

11.6.2 Action Attribute: `reference-source-attribute`

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

11.6.3 Example

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

```xml
<Target>
  <Actions>
    <Action>
      <ActionMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```
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.

A server MUST specify the subject identifier in an `<xacmlc:Request>` using the standard subject attribute named "urn:oasis:names:tc:xacml:1.0:subject:subject-id".

A server MUST specify a subject role in an `<xacmlc:Request>` using the standard subject attribute named "urn:oasis:names:tc:xacml:2.0:subject:role".

An Access Control Policy that uses Role Bases Access Control MUST specify a Permission PolicySet for each role as described in [XACML-RBAC].

This specification does not define how roles are defined or assigned to a subject. Implementations SHOULD to provide that functionality in an implementation-specific manner.

### 11.7.1 Example

The following example shows a Subject that matches a registered user with id "urn:acme:person:Danyal":

```xml
<Target>
  <Subjects>
    <Subject>
      <SubjectMatch>
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
          urn:acme:person:Danyal
        </AttributeValue>
      </SubjectMatch>
    </Subject>
  </Subjects>
</Target>
```

The following example shows a Subject that matches a subject role “employee”:

```xml
<Target>
  <Subjects>
    <Subject>
      <!-- Subject attribute designator for subject role -->
    </Subject>
  </Subjects>
</Target>
```
11.8 Resource Matching

A server MUST specify the following resource attributes in an <xacmlc:Request> as described in table below:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Identifier</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>urn:oasis:names:tc:xacml:2.0:resource:resource-id</td>
<td>Value MUST be the value of the id attribute of the RegistryObject resource</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>lid</td>
<td>urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:lid</td>
<td>Value MUST be the value of the lid attribute of the RegistryObject resource</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>objectType</td>
<td>urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:resource:objectType</td>
<td>Value MUST be the value of the objectType attribute the RegistryObject resource</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>owner</td>
<td>urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:owner</td>
<td>Value MUST be the value of the owner attribute of the RegistryObject resource</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
</tbody>
</table>

An XACML Access Control Policy MAY use resource attribute defined above within an <xacml:ResourceMatch> element.

In addition, 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.

11.8.1 Example

The following example uses XPATH expression to match resource if it has a Slot with name “someSlotName”.

```xml
<SubjectMatch>
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
  </AttributeValue>
  <SubjectAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
    DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
</SubjectMatch>
```
11.9 Canonical XACML Functions

Section A.3 of [XACML] defines a set of standard functions. This section defines addition XACML functions that MUST 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 MUST be set to "Indeterminate".

11.9.1 Function AssociationExists


<table>
<thead>
<tr>
<th>Parameter / Return</th>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter 1</td>
<td>sourceObject</td>
<td>Specifies a value for the sourceObject attribute of AssociationType. MAY use '%' and '_' as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 2</td>
<td>targetObject</td>
<td>Specifies a value for the targetObject attribute of AssociationType. MAY use '%' and '_' as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 3</td>
<td>type</td>
<td>Specifies the path attribute value for a ClassificationNode in the AssociationType ClassificationScheme. MAY use '%' and '_' as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
</tbody>
</table>

This attribute is used to match the type attribute of AssociationType. The type parameter MUST also match ClassificationNodes that are descendants of ClassificationNode specified by the type parameter.

This parameter is optional and MAY be omitted.

Returns MUST return "True" if and only if an AssociationType instance exists that matches the specified
### 11.9.2 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>
<thead>
<tr>
<th>Parameter / Return</th>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter 1</td>
<td>node1</td>
<td>Specifies the id of a ClassificationNode.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 2</td>
<td>node2</td>
<td>Specifies the id of a ClassificationNode.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Returns</td>
<td></td>
<td>MUST return “True” if and only if ClassificationNode with id matching node2 value is same as or descendent of ClassificationNode with id matching node1. MUST return “False” otherwise.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#boolean">http://www.w3.org/2001/XMLSchema#boolean</a></td>
</tr>
</tbody>
</table>

### 11.9.3 Function HasClassification

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

<table>
<thead>
<tr>
<th>Parameter / Return</th>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter 1</td>
<td>classifiedObject</td>
<td>Specifies a value for the classifiedObject attribute of ClassificationType. MAY use ‘%’ and ‘_’ as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 2</td>
<td>classificationNode</td>
<td>Specifies the id of targetObject for ClassificationType. MAY use ‘%’ and ‘_’ as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Returns</td>
<td></td>
<td>MUST return “True” if and only if an ClassificationType instance exists that matches the specified classifiedObject and classificationNode. The classificationNode parameter MUST also match ClassificationNodes that are descendants of the classificationNode specified by the classificationNode parameter.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#boolean">http://www.w3.org/2001/XMLSchema#boolean</a></td>
</tr>
</tbody>
</table>
11.9.4 Function HasSlot

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

<table>
<thead>
<tr>
<th>Parameter / Return</th>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter 1</td>
<td>registryObject</td>
<td>Specifies the id of the parent RegistryObjectType instance for the Slot. MAY use '%' and '_' as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 2</td>
<td>slotName</td>
<td>Specifies a value for the name attribute of Slot. MAY use '%' and '_' as wildcard to match multiple or single characters.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
</tr>
<tr>
<td>Parameter 3</td>
<td>slotValue</td>
<td>Specifies a value for the Slot. This parameter MAY be omitted.</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td></td>
<td>MUST return “True” if and only if a RegistryObjectType instance exists that has a Slot whose name matches the specified slotName and which has a value matches the specifies slotValue. MUST return “False” otherwise.</td>
<td><a href="http://www.w3.org/2001/XMLSchema#boolean">http://www.w3.org/2001/XMLSchema#boolean</a></td>
</tr>
</tbody>
</table>

11.10 Constraints on XACML Binding

This specification normatively defines the following constraints on the binding of the Access Control Model to [XACML]. These constraints MAY be relaxed in future versions of this specification.

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

11.11 Resolving Policy References

An XACML PolicySet MAY reference XACML Policy objects defined outside the repository item containing the XACML PolicySet. A server implementation MUST 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 section 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 MUST automatically catalog every PolicySet upon submission to contain a special Slot with name ComposedPolicies. The value of this Slot MUST 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.
Appendix A. Acknowledgments

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

Contributors:

- Rob Atkinson, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia
- Simon Cox, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia
- Lydia Gietler, Danish Ministry of the Environment
- Aleksei Valikov, Disy Informationssysteme GmbH
Appendix B. Revision History

[optional; should not be included in OASIS standards]
Appendix C. Non-Normative Text