ebXML Registry Information Model

Committee Draft 01, 10 February, 2005

Document identifier:

regrep-rim-3.0-cd-01

Location:

http://www.oasis-open.org/committees/regrep/documents/3.0/specs/regrep-rim-3.0-cd-01.pdf

Editors:

Name	Affiliation
Sally Fuger	Individual
Farrukh Najmi	Sun Microsystems
Nikola Stojanovic	RosettaNet

Contributors:

Name	Affiliation
Diego Ballve	Individual
Ivan Bedini	France Telecom
Kathryn Breininger	The Boeing Company
Joseph Chiusano	Booz Allen Hamilton
Peter Kacandes	Adobe Systems
Paul Macias	LMI Government Consulting
Carl Mattocks	Individual
Matthew MacKenzie	Adobe Systems
Monica Martin	Sun Microsystems
Richard Martell	Galdos Systems Inc
Duane Nickull	Adobe Systems

Abstract:

This document defines the types of metadata and content that can be stored in an ebXML Registry.

A separate document, ebXML Registry: Service and Protocols [ebRS], defines the services and protocols for an ebXML Registry.

Status:

This document is an OASIS ebXML Registry Technical Committee Approved Draft Specification.

Committee members should send comments on this specification to the regrep@lists.oasis-open.org list. Others should subscribe to and send comments to the regrep-comment@lists.oasis-open.org list. To subscribe, send an email message to regrep-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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 OASIS ebXML Registry TC web page (http://www.oasis-open.org/committees/regrep/).

Table of Contents

Illustration Index

1 Introduction

An ebXML Registry is an information system that securely manages any content type and the standardized metadata that describes it.

The ebXML Registry provides a set of services that enable sharing of content and metadata between organizational entities in a federated environment.

This document defines the types of metadata and content that can be stored in an ebXML Registry.

A separate document, ebXML Registry: Services and Protocols [ebRS], defines the services provided by an ebXML Registry and the protocols used by clients of the registry to interact with these services.

1.1 Audience

The target audience for this specification is the community of software developers who are:

- Implementers of ebXML Registry Services
- Implementers of ebXML Registry Clients

1.2 Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC 2119 [RFC2119].

The term *“repository item*” is used to refer to content (e.g. an XML document) that resides in a repository for storage and safekeeping. Each repository item is described by a RegistryObject instance. The RegistryObject catalogs the RepositoryItem with metadata.

1.3 Notational Conventions

Throughout the document the following conventions are employed to define the data structures used. The following text formatting conventions are used to aide readability:

1.3.1 UML Diagrams

Unified Modeling Language [UML] diagrams are used as a way to concisely describe concepts. They are not intended to convey any specific Implementation or methodology requirements.

1.3.2 Identifier Placeholders

Listings may contain values that reference ebXML Registry objects by their id attribute. These id values uniquely identify the objects within the ebXML Registry. For convenience and better readability, these key values are replaced by meaningful textual variables to represent such id values.
For example, the placeholder in the listing below refers to the unique id defined for an example Service object:

```
<rim:Service id="${EXAMPLE_ SERVICE_ID}">
```

1.3.3 Constants

Constant values are printed in the `Courier New font` always, regardless of whether they are defined by this document or a referenced document.

1.3.4 Bold Text

Bold text is used in listings to highlight those aspects that are most relevant to the issue being discussed. In the listing below, an example value for the contentLocator slot is shown in italics if that is what the reader should focus on in the listing:

```
<rim:Slot name="urn:oasis:names:tc:ebxml-
regrep:rim:RegistryObject:contentLocator">
...
</rim:Slot>
```

1.3.5 Example Values

These values are represented in *italic* font. In the listing below, an example value for the contentLocator slot is shown in italics:

```
<rim:Slot name="urn:oasis:names:tc:ebxml-
regrep:rim:RegistryObject:contentLocator">
        <rim:ValueList>
                <rim:Value>http://example.com/myschema.xsd</rim:Value>
        </rim:ValueList>
</rim:Slot>
```

1.4 XML Schema Conventions

This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative text to describe the syntax and semantics of XML-encoded objects and protocol messages. In cases of disagreement between the ebXML Registry schema documents and schema listings in this specification, the schema documents take precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.

Conventional XML namespace prefixes are used throughout this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example. The use of these namespace prefixes in instance documents is non-normative. However, for consistency and understandability instance documents SHOULD use these namespace prefixes.

1.4.1 Schemas Defined by ebXML Registry

Prefix	XML Namespace	Comments
`rim:`	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	This is the Registry Information Model namespace [ebRIM]. The prefix is generally elided in mentions of Registry Information Model elements in text.
`rs:`	urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0	This is the ebXML Registry namespace that defines base types for registry service requests and responses [ebRS]. The prefix is generally elided in mentions of ebXML Registry protocol-related elements in text.
`query:`	urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0	This is the ebXML Registry query namespace that is used in the query protocols used between clients and the QueryManager service [ebRS].

Prefix	XML Namespace	Comments
`lcm:`	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0	This is the ebXML Registry Life Cycle Management namespace that is used in the life cycle management protocols used between clients and the LifeCycleManager service [ebRS].
`cms:`	urn:oasis:names:tc:ebxml-regrep:xsd:cms:3.0	This is the ebXML Registry Content Management Services namespace that is used in the content management protocols used between registry and pluggable content managent services [ebRS].

1.4.2 Schemas Used By ebXML Registry

Prefix	XML Namespace	Comments
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
`samlp:`	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
`ecp:`	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 Enhanced Client Proxy profile namespace, specified in this document and in a schema [SAMLECP-xsd].
`ds:`	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
`xenc:`	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
`SOAP-ENV:`	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
`paos:`	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS (reverse SOAP) namespace.
`xsi:`	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.
`wsse:`	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	This namespace is defined by the Web Services Security: SOAP Message Security 1.0 specification [WSS-SMS]. It is used by registry to secure soap message communication.

Prefix	XML Namespace	Comments
`wsu:`	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	This namespace is defined by the Web Services Security: SOAP Message Security 1.0 specification [WSS-SMS]. It is used by registry to secure soap message communication.

1.5 RepositoryItems and RegistryObjects

An ebXML Registry is capable of storing any type of electronic content such as XML documents, text documents, images, sound and video. Instances of such content are referred to as a RepositorytItems. RepositorytItems are stored in a content *repository* provided by the ebXML Registry.

In addition to the RepositoryItems, an ebXML Registry is also capable of storing standardized metadata that MAY be used to further describe RepositoryItems. Instances of such metadata are referred to as a RegistryObjects (or one of its sub-types, as described later in this document). RegistryObjects are stored in the *registry* provided by the ebXML Registry.

To illustrate these concepts consider this familiar metaphor:

- An ebXML Registry is like your local library.
- The repository is like the bookshelves in the library.
- The repository items in the repository are like book on the bookshelves. The repository items can contain any type of electronic content just like the books in the bookshelves can contain any type of information.
- The registry is like the card catalog. It is organized for finding things quickly.
- A RegistryObject is like a card in the card catalog. All RegistryObjects conform to a standard just like the cards in the card catalog conform to a standard.
- Every repository item MUST have a RegistryObject that describes it, just like every book must have a card in the card catalog.

To summarize, ebXML Registry stores any type of content as RepositoryItems in a repository and stores standardized metadata describing the content as RegistryObjects in a registry.

1.6 Canonical ClassificationSchemes

This specification uses several standard ClassificationSchemes as a mechanism to provides extensible enumeration types. These ClassificationSchemes are referred to as *canonical ClassificationSchemes*. The enumeration values within canonical ClassificationSchemes are defined using standard ClassificationNodes that are referred to as *canonical ClassificationNodes*.

This section lists the canonical ClassificationSchemes that are required to be present in all ebXML Registries. These Canonical ClassificationSchemes MAY be extended by adding additional ClassificationNodes. However, a ClassificationNode defined normatively in the links below MUST NOT be modified within a registry. In particular they MUST preserve their canonical id attributes in all registries.

Note that all files listed in the Location column are relative to the following URL:

http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/

ClassificationScheme Name	Location / Description
AssociationType	SubmitObjectsRequest_AssociationTypeScheme.xml Defines the types of associations between RegistryObjects.
ContentManagementService	SubmitObjectsRequest_CMSScheme.xml Defines the types of content management services.

ClassificationScheme Name	Location / Description
DataType	SubmitObjectsRequest_DataTypeScheme Defines the data types for attributes in classes defined by this document.
DeletionScopeType	SubmitObjectsRequest_DeletionScopeTypeScheme.xml Defines the values for the deletionScope attribute in RemoveObjectsRequest protocol message.
EmailType	SubmitObjectsRequest_EmailTypeScheme.xml Defines the types of email addresses.
ErrorHandlingModel	SubmitObjectsRequest_ErrorHandlingModelScheme.xml Defines the types of error handling models for content management services.
ErrorSeverityType	SubmitObjectsRequest_ErrorSeverityTypeScheme.xml Defines the different error severity types encountered by registry during processing of protocol messages.
EventType	SubmitObjectsRequest_EventTypeScheme.xml Defines the types of events that can occur in a registry.
InvocationModel	SubmitObjectsRequest_InvocationModelScheme.xml Defines the different ways that a content management service may be invoked by the registry.
NodeType	SubmitObjectsRequest_NodeTypeScheme.xml Defines the different ways in which a ClassificationScheme may assign the value of the code attribute for its ClassificationNodes.
NotificationOptionType	SubmitObjectsRequest_NotificationOptionTypeScheme.xml Defines the different ways in which a client may wish to be notified by the registry of an event within a Subscription.
ObjectType	SubmitObjectsRequest_ObjectTypeScheme.xml Defines the different types of RegistryObjects a registry may support.
PhoneType	SubmitObjectsRequest_PhoneTypeScheme.xml Defines the types of telephone numbers.
QueryLanguage	SubmitObjectsRequest_QueryLangScheme Defines the query languages supported by a registry.
ResponseStatusType	SubmitObjectsRequest_ResponseStatusTypeScheme.xml Defines the different types of status for a RegistryResponse.
StatusType	SubmitObjectsRequest_StatusTypeScheme.xml Defines the different types of status for a RegistryObject.
SubjectGroup	SubmitObjectsRequest_SubjectGroupScheme Defines the groups that a User may belong to for access control purposes.
SubjectRole	SubmitObjectsRequest_SubjectRoleScheme Defines the roles that may be assigned to a User for access control purposes.

400

1.7 Registry Information Model: Overview

The ebXML Registry Information Model defined in this document defines the classes and their relationships that are used to represent RegistryObject metadata.

1.7.1 Class Relationships View

Figure 1 provides a high level overview of the metadata classes 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.

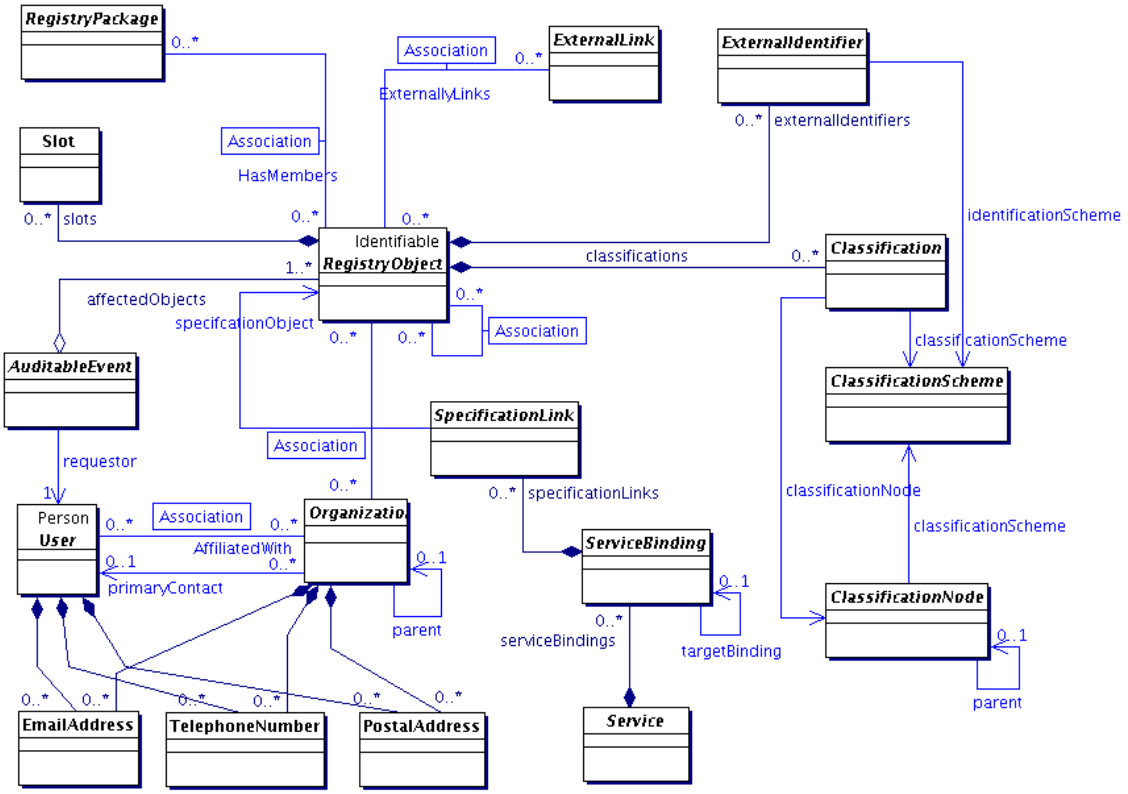


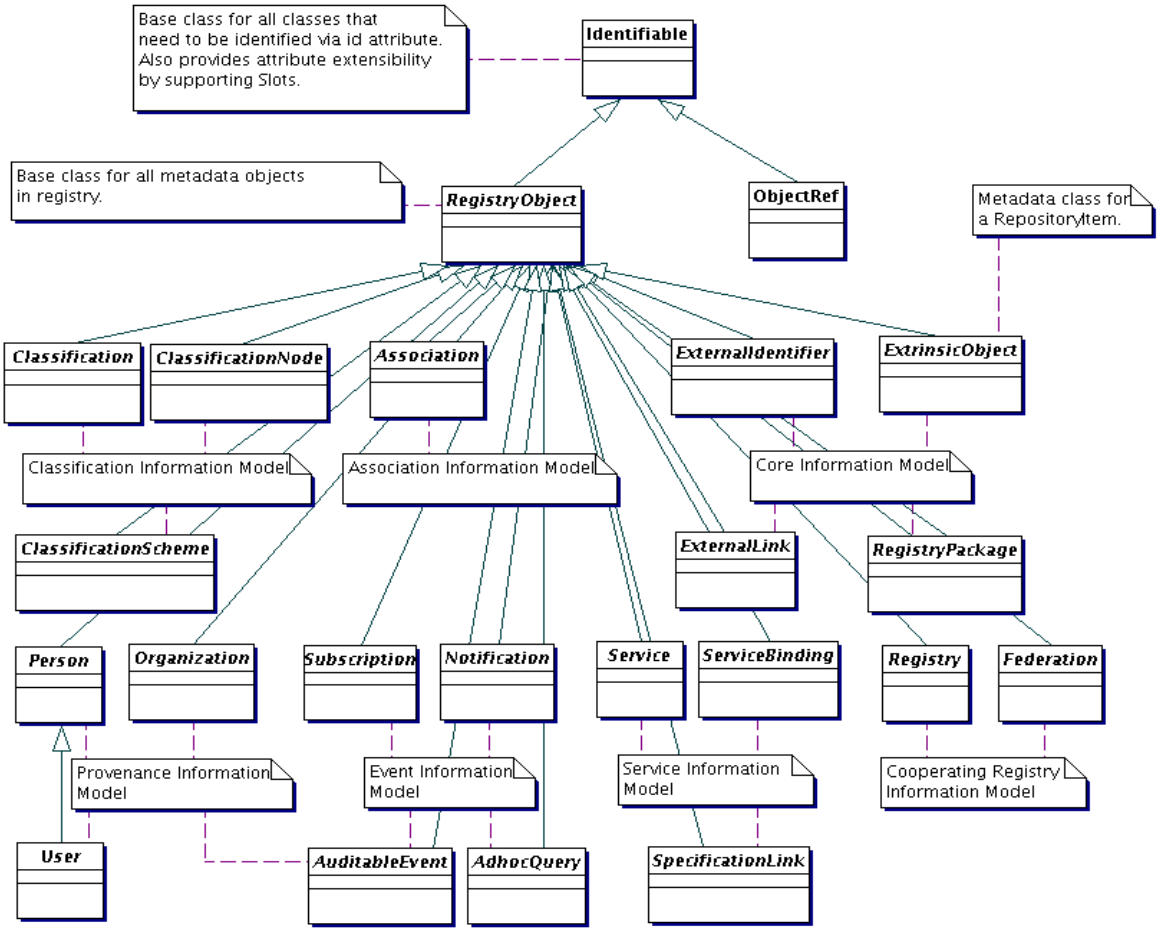
Figure 1: Information Model Relationships View

1.7.2 Class Inheritance View

Figure 2 shows the inheritance or “Is-A” relationships between the classes 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 Figure 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.

1.7.2.1 Class Identifiable

The RegistryObject class and some other classes in RIM are derived from a class called *Identifiable*. This class provides the ability to identify objects by an id attribute and also provides attribute extensibility by allowing dynamic, instance-specific attributes called Slots.



Base class for all classes that need to be identified via id attribute. Also provides attribute extensibility by supporting Slots.
Identifiable
Base class for all metadata objects in registry.
RegistryObject
ObjectRef
Metadata class for a RepositoryItem.
Classification
ClassificationNode
Association
ExternalIdentifier
ExtrinsicObject
Classification Information Model
Association Information Model
Core Information Model
ClassificationScheme
ExternalLink
RegistryPackage
Person
Organization
Subscription
Notification
Service
ServiceBinding
Registry
Federation
Provenance Information Model
Event Information Model
Service Information Model
Cooperating Registry Information Model
User
AuditableEvent
AdhocQuery
SpecificationLink

Figure 2: Information Model *Inheritance* View

The RegistryObject sub-classes are shown in related groups as follows:

- Core Information Model: Defines core metadata classes in the model including the common base classes.
- Association Information Model: Defines classes that enable RegistryObject instances to be associated with each other.
- Classification Information Model: Defines classes that enable RegistryObjects to be classified.
- Provenance Information Model: Defines classes that enable the description of provenance or source information about a RegistryObject.
- Service Information Model: Defines classes that enable service description.
- Event Information Model: Defines classes that enable the event subscription and notification feature defined in [ebRS].
- Cooperating Registries Information Model: Defines classes that enable the cooperating registries feature defined in [ebRS].

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

2 Core Information Model

This section covers the most commonly used information model classes defined by [ebRIM].

2.1 Attributes of Information Model Classes

Information model classes are defined in terms of their attributes. These attributes provide information on the state of the instances of these classes. Implementations of a registry typically map class attributes to attributes and elements in an XML store or columns in a relational store.

Since the model supports inheritance between classes, a class in the model inherits attributes from its super classes if any, in addition to defining its own specialized attributes.

The following is the description of the columns of many tables that summarize the attributes of a class:

Column	Description
Attribute	The name of the attribute
Data Type	The data type for the attribute
Required	Specifies whether the attribute is required to be specified
Default Value	Specifies the default value in case the attribute is omitted
Specified By	Indicates whether the attribute is specified by the client or specified by the registry. In some cases it may be both.
Mutable	Specifies whether an attribute may be changed once it has been set to a certain value

2.2 Data Types

The following table lists the various data types used by the attributes within information model classes:

Data Type	XML Schema Data Type	Description	Length
Boolean	boolean	Used for a true or false value	
String4	string	Used for 4 character long strings	4 characters
String8	string	Used for 8 character long strings	8 characters
String16	string	Used for 16 character long strings	16 characters
String32	string	Used for 32 character long strings	32 characters
String	string	Used for unbounded Strings	unbounded
ShortName	string	A short text string	64 characters
Language	language	A string that identifies a local language. Values MUST be natural language identifiers as defined by [RFC 3066]	32 character
LongName	string	A long text string	256 characters
FreeFormText	string	A very long text string for free-form text	1024 characters
UUID	anyURI	A URI of the form urn:uuid:<uuid> where <uuid> MUST be a DCE 128 Bit Universally unique Id.	64 characters
ObjectRef	referenceURI	In XML Schema the referenceURI attribute value is a URI that references an ObjectRef within the XML document. If no such ObjectRef exists in the XML document then the value implicitly references a RegistryObject by the value of its id attribute within the registry.	64 characters
URI	anyURI	Used for URL and URN values	256 characters

URN	anyURI	Must be a valid URN	256 characters
Integer	integer	Used for integer values	4 bytes
DateTime	dateTime	Used for a timestamp value such as Date	
Set	sequence	As defined by OCL. An unordered Collection in which an object can occur only once.	
Bag	sequence	As defined by OCL. An unordered Collection in which the same object can occur multiple times.	
Sequence	sequence	As defined by OCL. An ordered Collection in which the same object can occur multiple times.	

2.3 Internationalization (I18N) Support

Some information model classes have String attributes that are I18N capable and may be localized into multiple native languages. Examples include the name and description attributes of the RegistryObject class in 2.5.

The information model defines the InternationalString and the LocalizedString interfaces to support I18N capable attributes within the information model classes. These classes are defined below.

2.3.1 Class InternationalString

This class is used as a replacement for the String type whenever a String attribute needs to be I18N capable. An instance of the InternationalString class composes within it a Set of LocalizedString instances, where each String is specific to a particular locale.

2.3.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
localizedStrings	Set of LocalizedString	No		Client	Yes

2.3.1.2 Attribute localizedStrings

Each InternationalString instance MAY have a *localizedStrings* attribute that is a Set of zero or more LocalizedString instances.

2.3.2 Class LocalizedString

This class is used as a simple wrapper class that associates a String with its locale. The class is needed in the InternationalString class where a Set of LocalizedString instances are kept. Each LocalizedString instance has a charset and lang attribute as well as a value attribute of type String.

2.3.2.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
lang	language	No	en-US	Client	Yes
charset	String	No	UTF-8	Client	Yes
value	String	Yes		Client	Yes

2.3.2.2 Attribute lang

Each LocalizedString instance MAY have a *lang* attribute that specifies the language used by that LocalizedString.

2.3.2.3 Attribute charset

Each LocalizedString instance MAY have a *charset* attribute that specifies the name of the character set used by that LocalizedString. The value of this attribute SHOULD be registered with IANA at:

http://www.iana.org/assignments/character-sets

2.3.2.4 Attribute value

Each LocalizedString instance MUST have a *value* attribute that specifies the string value used by that LocalizedString.

2.4 Class Identifiable

The Identifiable class is the common super class for most classes in the information model. Information model Classes whose instances have a unique identity are descendants of the Identifiable Class.

2.4.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
home	URI	No	Base URI of local registry	Client	Yes
id	URN	Yes		Client or registry	No
slots	Set of Slot	No		Client	Yes

2.4.2 Attribute id

Each Identifiable instance MUST have a unique identifier which is used to refer to that object.

Note that classes in the information model that do not inherit from Identifiable class do not require a unique id. Examples include classes such as TelephoneNumber, PostalAddress, EmailAddress and PersonName.

An Identifiable instance MUST have an id that MUST conform to the rules defined in section title "Unique ID Generation" in [ebRS].

2.4.3 Attribute home

An Identifiable instance MAY have a *home* attribute. The *home* attribute, if present, MUST contain the base URL to the home registry for the RegistryObject instance. The home URL MUST be specified for instances of the Registry class that is defined later in this specification.

The base URL of a registry is:

- Used as the URL prefix for SOAP and HTTP interface bindings to the registry.
- Used to qualify the id of an Identifiable instance by its registry within a federated registry environment.

2.4.4 Attribute slots

An Identifiable instance MAY have a Set of zero or more Slot instances that are composed within the Identifiable instance. These Slot instances serve as extensible attributes that MAY be defined for the

Identifiable instance.

2.5 Class RegistryObject

Super Classes: Identifiable

The RegistryObject class extends the Identifiable class and serves as a common super class for most classes in the information model.

2.5.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classifications	Set of Classification	No		Client	Yes
description	InternationalString	No		Client	Yes
externalIdentifers	Set of ExternalIdentifier	No		Client	Yes
lid	URN	Yes for READs, No for WRITEs.		Client or registry	No
name	InternationalString	No		Client	Yes
objectType	ObjectRef	Yes for READs, No for WRITEs.		Client or Registry	No
status	ObjectRef	Yes for READs, No for WRITEs.		Registry	Yes
versionInfo	VersionInfo	Yes for READs, No for WRITEs.		Registry	No

2.5.2 Composed Object

A RegistryObject instance MAY have instances of other RegistryObjects and other classes composed within it as defined in this specification. In such a relationship the composing object is refered to as the *Composite* object as defined in section 3.4 of [UML]. The composed object is refered to in this document and other ebXML Registry specification as *Composed* object. The relationship between the Composite and Composed object is refered as a composition relationship as defined in section 3.4.8 of [UML].

Composition relationship implies that deletes and copies of the Composite object are cascaded to implicitly delete or copy the composed object. In comparison a UML Aggregation implies no such cascading.

The following classes defined by [RIM] are compsed types and follow the rules defined by UML composition relationships. The classes are listed in the order of their being defined in this document. Note that abstract classes are not included in this list since an abstract class cannot have any instances.

- InternationalString
- LocalizedString
- VersionInfo
- Slot
- ExternalIdentifier
- Classification

- PostalAddress
- TelephoneNumber
- EmailAddress
- PersonName
- ServiceBinding
- SpecificationLink
- QueryExpression
- NotifyAction

2.5.3 Attribute classifications

Each RegistryObject instance MAY have a Set of zero or more Classification instances that are composed within the RegistryObject. These Classification instances classify the RegistryObject.

2.5.4 Attribute description

Each RegistryObject instance MAY have textual description in a human readable and user-friendly form. This attribute is I18N capable and therefore of type InternationalString.

2.5.5 Attribute externalIdentifier

Each RegistryObject instance MAY have a Set of zero or more ExternalIdentifier instances that are composed within the RegistryObject. These ExternalIdentifier instances serve as alternate identifiers for the RegistryObject.

2.5.6 Attribute lid

Each RegistryObject 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 submitter when creating the original version of a RegistryObject. If the submitter assigns the lid attribute, she must guarantee that it is a globally unique URN. A registry MUST honor a valid submitter-supplied LID. If the submitter does not specify a LID then the registry 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.

Note that classes in the information model that do not inherit from RegistryObject class do not require a lid. Examples include Entity classes such as TelephoneNumber, PostalAddress, EmailAddress and PersonName.

2.5.7 Attribute name

Each RegistryObject instance MAY have a human readable name. The name does not need to be unique with respect to other RegistryObject instances. This attribute is I18N capable and therefore of type InternationalString.

2.5.8 Attribute objectType

Each RegistryObject instance has an *objectType* attribute. The value of the objectType attribute MUST be a reference to a ClassificationNode in the canonical ObjectType ClassificationScheme. A Registry MUST support the object types as defined by the ObjectType ClassificationScheme. The canonical ObjectType ClassificationScheme may easily be extended by adding additional ClassificationNodes to the canonical ObjectType ClassificationScheme.

The *objectType* for almost all objects in the information model matches the ClassificationNode that corresponds to the name of their class. For example the *objectType* for a Classification is a reference to the ClassificationNode with code “Classification” in the canonical ObjectType ClassificationScheme. The only exception to this rule is that the *objectType* for an ExtrinsicObject or an ExternalLink instance MAY be defined by the submitter and indicates the type of content associated with that object.

A registry MUST set the correct objectType on a RegistryObject when returning it as a response to a client request. A client MAY set the objectType on a RegistryObject when submitting the object. A client SHOULD set the objectType when the object is an ExternalLink or an ExtrinsicObject since content pointed to or described by these types may be of arbitrary objectType.

2.5.9 Attribute status

Each RegistryObject instance MUST have a life cycle status indicator. The status is assigned by the registry. A registry 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 registry. A registry 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.

2.5.9.1 Pre-defined RegistryObject Status Types

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

Name	Description
Approved	Status of a RegistryObject that catalogues content that has been submitted to the registry and has been subsequently approved.
Deprecated	Status of a RegistryObject that catalogues content that has been submitted to the registry and has been subsequently deprecated.
Submitted	Status of a RegistryObject that catalogues content that has been submitted to the registry.
Withdrawn	Status of a RegistryObject that catalogues content that has been withdrawn from the registry. A repository item has been removed but its ExtrinsicObject still exists.

2.5.10 Attribute versionInfo

Each RegistryObject instance MAY have a *versionInfo* attribute. The value of the versionInfo attribute MUST be of type VersionInfo. The versionInfo attribute provides information about the specific version of a RegistryObject. The versionInfo attribute is set by the registry.

2.6 Class VersionInfo

VersionInfo class encapsulates information about the specific version of a RegistryObject.

The attributes of the VersionInfo class are described below.

2.6.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable

versionName	String16	Yes	1.1	Registry	Yes
comment	LongName	No		Registry	Yes

2.6.2 Attribute versionName

Each VersionInfo instance MUST have versionName. This attribute defines the version name identifying the VersionInfo for a specific RegistryObject version. The value for this attribute MUST be automatically generated by the Registry implementation.

2.6.3 Attribute comment

Each VersionInfo instance MAY have comment. This attribute defines the comment associated with the VersionInfo for a specific RegistryObject version. The value of the comment attribute is indirectly provided by the client as the value of the comment attribute of the <rim:Request> object. 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.

2.7 Class ObjectRef

Super Classes: Identifiable

The information model supports the ability for an attribute in an instance of an information model class to reference a RegistryObject instance using an object reference. An object reference is modeled in this specification with the ObjectRef class.

An instance of the ObjectRef class is used to reference a RegistryObject. A RegistryObject MAY be referenced via an ObjectRef instance regardless of its location within a registry or that of the object referring to it.

2.7.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
id	URN	Yes		Client	Yes
home	URI	No	Base URI of local registry	Client	Yes
createReplica	Boolean	No	false	Client	Yes

2.7.2 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.

2.7.3 Attribute *home*

Every ObjectRef instance MAY optionally have a *home* attribute specified. The *home* attribute if present MUST contain the base URI to the home registry for the referenced RegistryObject. The base URI to a registry is described by the REST interface as defined in [ebRS].

2.7.3.1 Local Vs. Remote ObjectRefs

When the *home* attribute is specified, and matches the base URI of a remote registry, then ObjectRef is referred to as a remote ObjectRef.

If the *home* attribute is null then its default value is the base URI to the current registry. When the *home* attribute is null or matches the base URI of the current registry, then the ObjectRef is referred to as a local

ObjectRef.

2.7.4 Attribute createReplica

Every ObjectRef instance MAY have a *createReplica* attribute. The *createReplica* attribute is a client supplied hint to the registry. When createReplica is true a registry SHOULD create a local replica for the RegistryObject being referenced if it happens to be a remote ObjectRef.

2.8 Class Slot

Slot instances provide a dynamic way to add arbitrary attributes to RegistryObject instances. This ability to add attributes dynamically to RegistryObject instances enables extensibility within the information model.

A slot is composed of a name, a slotType and a Bag of values.

2.8.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
name	LongName	Yes		Client	No
slotType	LongName	No		Client	No
values	Sequence of LongName	Yes		Client	No

2.8.2 Attribute *name*

Each Slot instance MUST have a name. The name is the primary means for identifying a Slot instance within a RegistryObject. Consequently, the name of a Slot instance MUST be locally unique within the RegistryObject instance.

2.8.3 Attribute slotType

Each Slot instance MAY have a slotType that allows different slots to be grouped together. The slotType attribute MAY also be used to indicate the data type or value domain for the slot value(s).

2.8.4 Attribute values

A Slot instance MUST have a Sequence of values. The Sequence of values MAY be empty. Since a Slot represent an extensible attribute whose value MAY be a Sequence, therefore a Slot is allowed to have a Sequence of values rather than a single value.

2.9 Class ExtrinsicObject

Super Classes: RegistryObject

The ExtrinsicObject class is the primary metadata class for a RepositoryItem.

2.9.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
contentVersionInfo	VersionInfo	Yes for READs, No for WRITEs.		Registry	No
isOpaque	Boolean	No	false	Client	No

mimeType	LongName	No	application/octet-stream	Client	No

Note that attributes inherited from super classes are not shown in the table above.

2.9.2 Attribute contentVersionInfo

Each ExtrinsicObject instance MAY have a *contentVersionInfo* attribute. The value of the *contentVersionInfo* attribute MUST be of type VersionInfo. The *contentVersionInfo* attribute provides information about the specific version of the RepositoryItem associated with an ExtrinsicObject. The *contentVersionInfo* attribute is set by the registry.

2.9.3 Attribute isOpaque

Each ExtrinsicObject instance MAY have an isOpaque attribute defined. This attribute determines whether the content catalogued by this ExtrinsicObject is opaque to (not readable by) the registry. In some situations, a Submitting Organization may submit content that is encrypted and not even readable by the registry.

2.9.4 Attribute mimeType

Each ExtrinsicObject instance MAY have a mimeType attribute defined. The mimeType provides information on the type of repository item catalogued by the ExtrinsicObject instance. The value of this attribute SHOULD be a registered MIME media type at http://www.iana.org/assignments/media-types.

2.10 Class RegistryPackage

Super Classes: RegistryObject

RegistryPackage instances allow for grouping of logically related RegistryObject instances even if individual member objects belong to different Submitting Organizations.

2.10.1 Attribute Summary

The RegistryPackage class defines no new attributes other than those that are inherited from RegistryObject super class. The inherited attributes are not shown here.

2.11 Class ExternalIdentifier

Super Classes: RegistryObject

ExternalIdentifier instances provide the additional identifier information to RegistryObject such as DUNS number, Social Security Number, or an alias name of the organization. The attribute *identificationScheme* is used to reference the identification scheme (e.g., “DUNS”, “Social Security #”), and the attribute *value* contains the actual information (e.g., the DUNS number, the social security number). Each RegistryObject MAY contain 0 or more ExternalIdentifier instances.

2.11.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
identificationScheme	ObjectRef	Yes		Client	Yes
registryObject	ObjectRef	Yes		Client	No
value	LongName	Yes		Client	Yes

Note that attributes inherited from the super classes of this class are not shown.

2.11.2 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 RegistryObject referenced by the RegistryObject attribute.

2.11.3 Attribute registryObject

Each ExternalIdentifier instance MUST have a *registryObject* attribute that references the parent RegistryObject for which this is an ExternalIdentifier.

2.11.4 Attribute value

Each ExternalIdentifier instance MUST have a *value* attribute that provides the identifier value for this ExternalIdentifier (e.g., the actual social security number).

2.12 Class ExternalLink

Super Classes: RegistryObject

ExternalLinks use URIs to associate content in the registry with content that MAY reside outside the registry. For example, an organization submitting an XML Schema could use an ExternalLink to associate the XML Schema with the organization's home page.

2.12.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
externalURI	URI	Yes		Client	Yes

2.12.2 Attribute externalURI

Each ExternalLink instance MUST have an externalURI attribute defined. The externalURI attribute provides a URI to the external resource pointed to by this ExternalLink instance. If the URI is a URL then a registry MUST validate the URL to be resolvable at the time of submission before accepting an ExternalLink submission to the registry.

3 Association Information Model

A RegistryObject instance MAY be associated with zero or more RegistryObject instances. The information model defines the Association class, an instance of which MAY be used to associate any two RegistryObject instances.

3.1 Example of an Association

One example of such an association is between two ClassificationScheme instances, where one ClassificationScheme supersedes the other ClassificationScheme as shown in Figure 3. This may be the case when a new version of a ClassificationScheme is submitted.

In Figure 3, we see how an Association is defined between a new version of the NAICS ClassificationScheme and an older version of the NAICS ClassificationScheme.

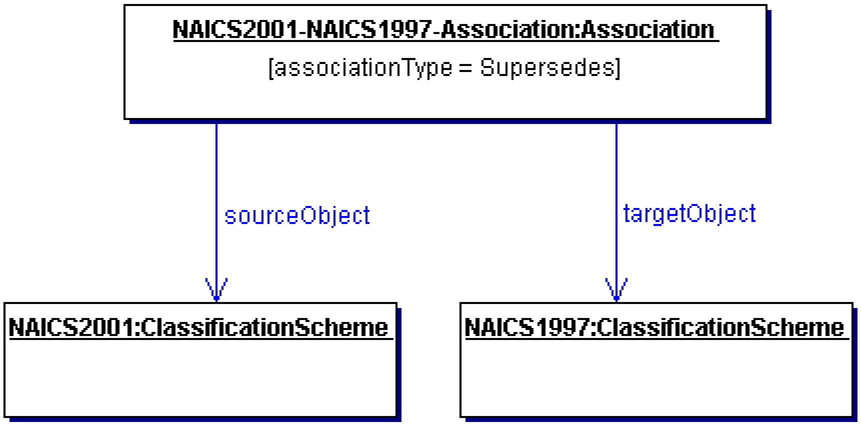

NAICS2001-NAICS1997-Association:Association
[associationType = Supersedes]
sourceObject
targetObject
NAICS2001:ClassificationScheme
NAICS1997:ClassificationScheme

Figure 3: Example of RegistryObject Association

3.2 Source and Target Objects

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

In the example in Figure 3, it is important to make the newer version of NAICS ClassificationScheme be the sourceObject and the older version of NAICS be the targetObject because the associationType implies that the sourceObject supersedes the targetObject (and not the other way around).

3.3 Association Types

Each Association MUST have an associationType 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.

3.4 Intramural Association

A common use case for the Association class is when a User “u” creates an Association “a” between two RegistryObjects “o1” and “o2” where Association “a” and RegistryObjects “o1” and “o2” are objects that were created by the same User “u”. This is the simplest use case, where the Association is between two objects that are owned by the same User that is defining the Association. Such Associations are referred to as intramural Associations.

Figure 4 below, extends the previous example in Figure 3 for the intramural Association case.

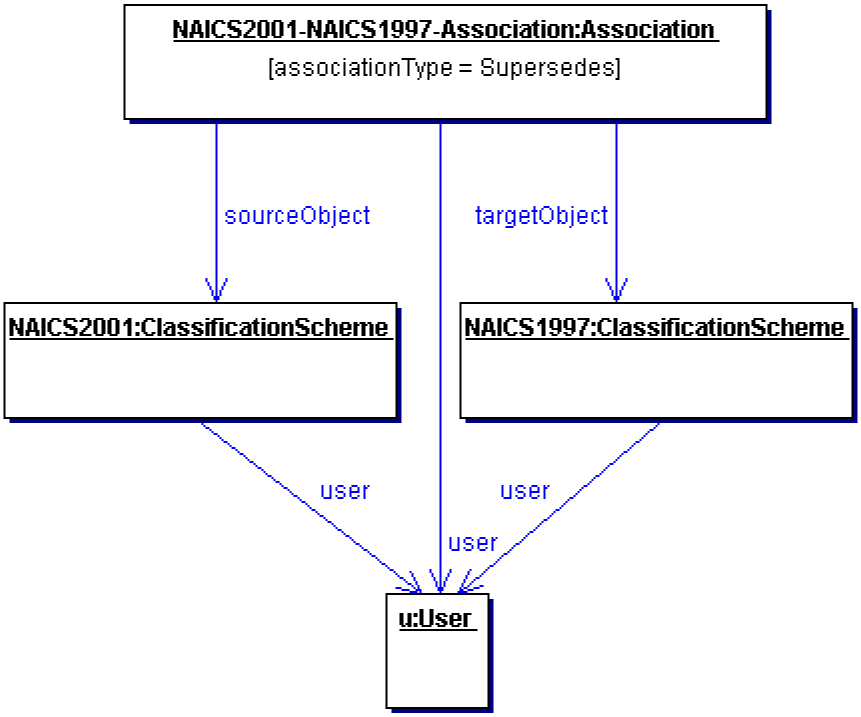

NAICS2001-NAICS1997-Association:Association
[associationType = Supersedes]
sourceObject
targetObject
NAICS2001:ClassificationScheme
NAICS1997:ClassificationScheme
user
user
user
u:User

Figure 4: Example of Intramural Association

3.5 Extramural Association

The information model also allows more sophisticated use cases. For example, a User “u1” creates an Association “a” between two RegistryObjects “o1” and “o2” where Association “a” is owned by User “u1”, but RegistryObjects “o1” and “o2” are owned by User “u2” and User “u3” respectively.

In this use case an Association is defined where either or both objects that are being associated are owned by a User different from the User defining the Association. Such Associations are referred to as extramural Associations.

Figure 5 below, extends the previous example in Figure 4 for the extramural Association case. Note that it is possible for an extramural Association to have two distinct Users rather than three distinct Users as shown in Figure 5. In such case, one of the two users owns two of the three objects involved (Association, sourceObject and targetObject).

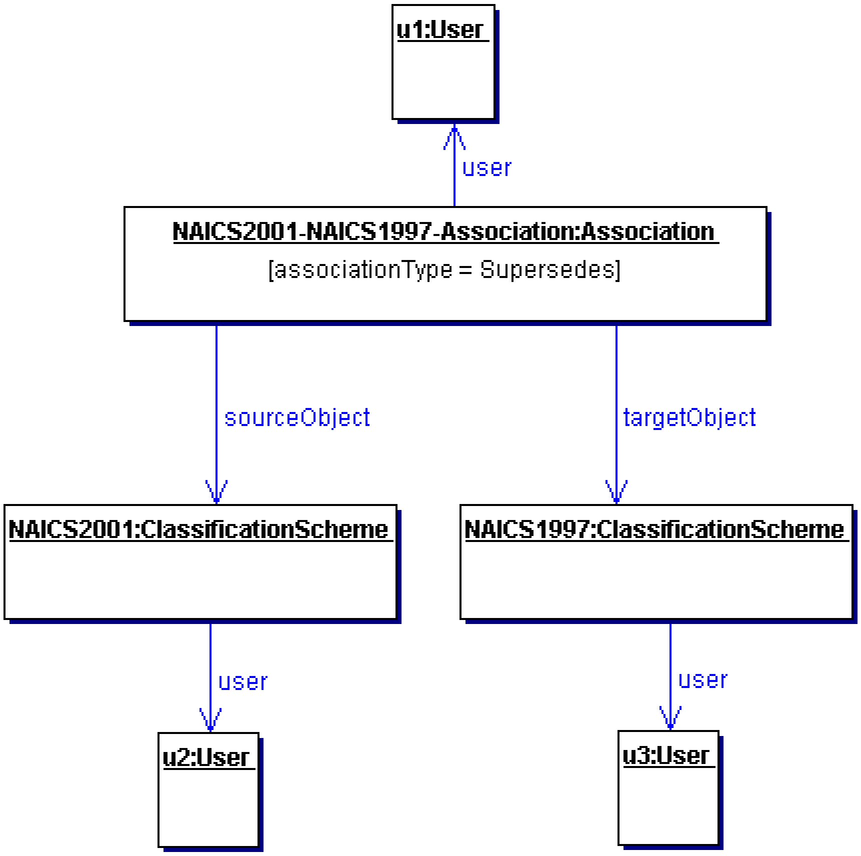

u1:User
user
NAICS2001-NAICS1997-Association:Association
[associationType = Supersedes]
sourceObject
targetObject
NAICS2001:ClassificationScheme
NAICS1997:ClassificationScheme
user
user
u2:User
u3:User

Figure 5: Example of Extramural Association

3.5.1 Controlling Extramural Associations

The owner of a RegistryObject MAY control who can create extramural associations to that RegistryObject using custom access control policies using the reference access control feature described in section 9.2.4.

3.6 Class Association

Super Classes: RegistryObject

Association instances are used to define many-to-many associations among RegistryObjects in the information model.

An instance of the Association class represents an association between two RegistryObjects.

3.6.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
associationType	ObjectRef	Yes		Client	No
sourceObject	ObjectRef	Yes		Client	No
targetObject	ObjectRef	Yes		Client	No

3.6.2 Attribute associationType

Each Association MUST have an *associationType* attribute that identifies the type of that association. The value of the associationType attribute MUST be a reference to a ClassificationNode within the canonical AssociationType ClassificationScheme. While the AssociationType scheme MAY easily be extended, a Registry MUST support the canonical association types as defined by the canonical AssociationType ClassificationScheme.

3.6.3 Attribute sourceObject

Each Association MUST have a *sourceObject* attribute that references the RegistryObject instance that is the source of that Association.

3.6.4 Attribute targetObject

Each Association MUST have a *targetObject* attribute that references the RegistryObject instance that is the target of that Association.

4 Classification Information Model

This section describes how the information model supports Classification of RegistryObject.

A RegistryObject MAY be classified in many ways. For example the RegistryObject for the same Collaboration Protocol Profile (CPP) may be classified by its industry, by the products it sells and by its geographical location.

A general ClassificationScheme can be viewed as a tree structure. In the example shown in Figure 6, RegistryObject instances representing Collaboration Protocol Profiles are shown as shaded boxes. Each Collaboration Protocol Profile represents an automobile manufacturer. Each Collaboration Protocol Profile 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 ClassificationNode instances under multiple ClassificationScheme instances (e.g., Industry, Geography).

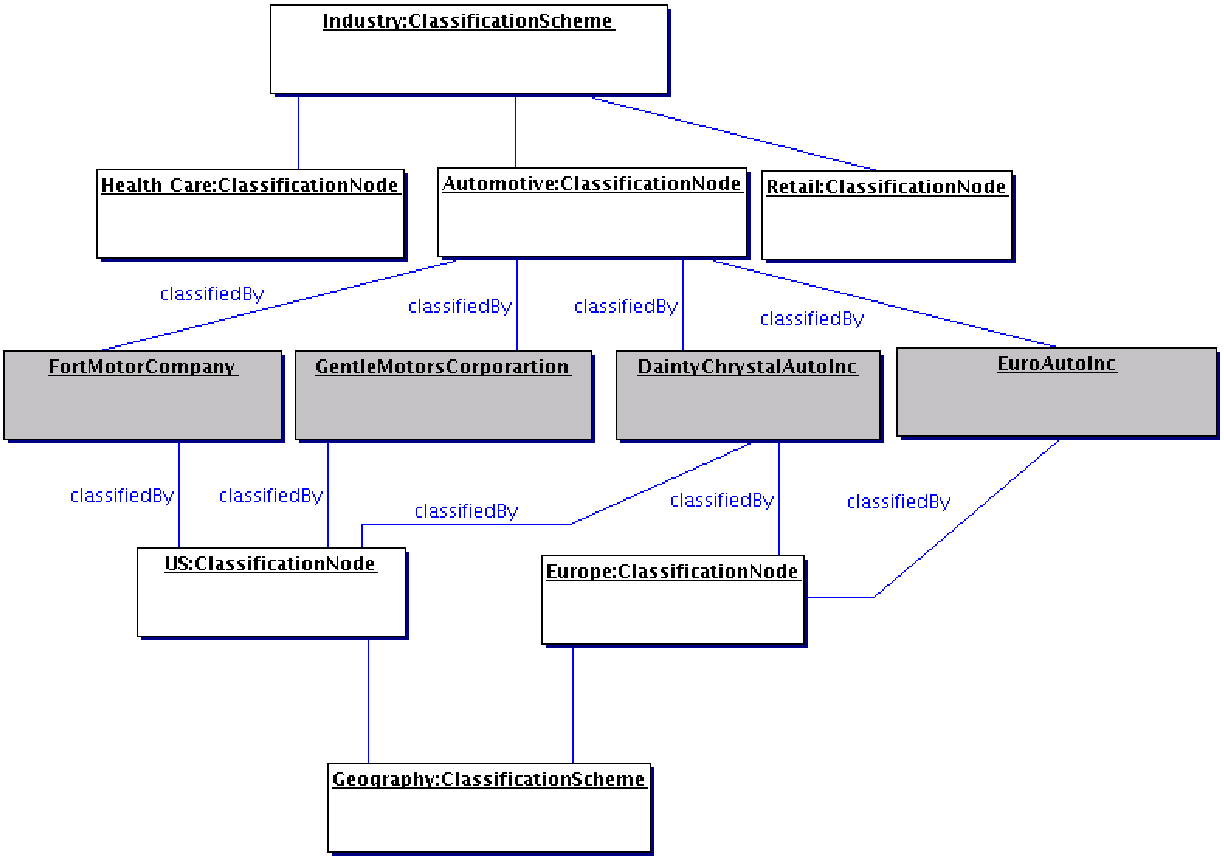


Figure 6: Example showing a *Classification* Tree

It is important to point out that the shaded nodes (FortMotorCompnay, GentleMotorsCorporation etc.) are not part of the ClassificationScheme tree. The leaf nodes of the ClassificationScheme tree are Health Care, Automotive, Retail, US and Europe. The shaded nodes are associated with the ClassificationScheme tree via a Classification Instance that is not shown in the picture.

In order to support a general ClassificationScheme that can support single level as well as multi-level Classifications, the information model defines the classes and relationships shown in Figure 7.

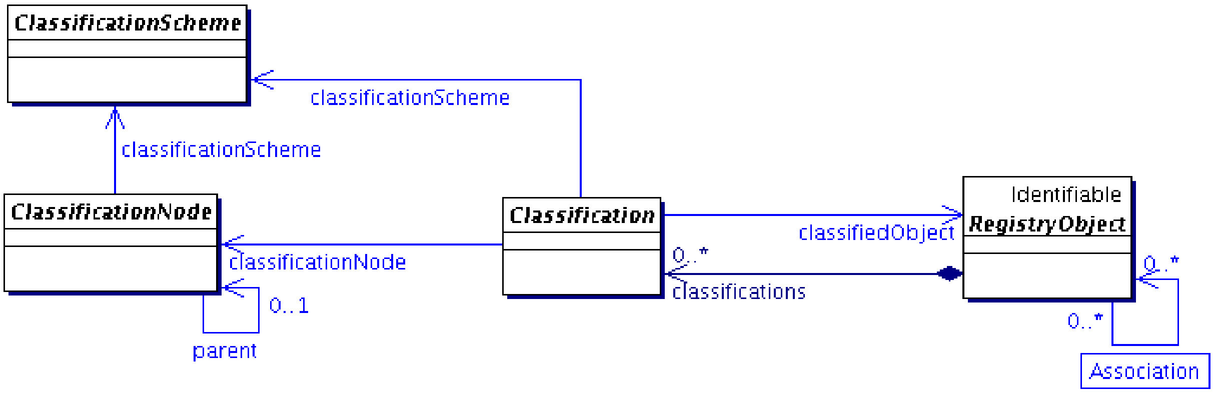


Figure 7: Information Model *Classification* View

A Classification is somewhat like a specialized form of an Association. Figure 8 shows an example of an ExtrinsicObject Instance for a Collaboration Protocol Profile (CPP) object that is classified by a ClassificationNode representing the Industry that it belongs to.

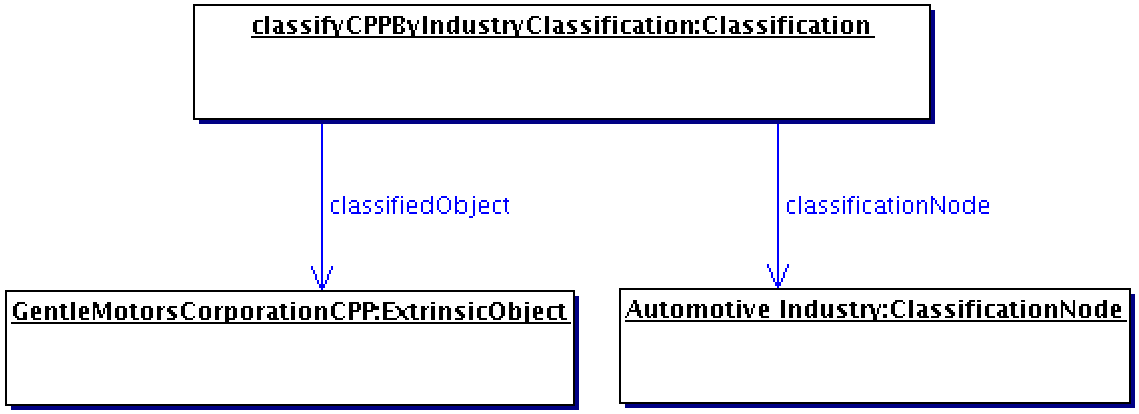

classifyCPPByIndustryClassification:Classification
classifiedObject
classificationNode
GentleMotorsCorporationCPP:ExtrinsicObject
Automotive Industry:ClassificationNode

Figure 8: Classification *Instance* Diagram

4.1 Class ClassificationScheme

Super Classes: RegistryObject

A ClassificationScheme instance describes a taxonomy. The taxonomy hierarchy may be defined internally to the registry by instances of ClassificationNode, or it may be defined externally to the Registry, in which case the structure and values of the taxonomy elements are not known to the Registry.

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

4.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
isInternal	Boolean	Yes		Client	No
nodeType	ObjectRef	Yes		Client	No

Note that attributes inherited by a ClassificationScheme class from the RegistryObject class are not shown.

4.1.2 Attribute isInternal

When submitting a ClassificationScheme instance the submitter MUST declare whether the ClassificationScheme instance represents an internal or an external taxonomy. This allows the registry to validate the subsequent submissions of ClassificationNode and Classification instances in order to maintain the type of ClassificationScheme consistent throughout its lifecycle.

4.1.3 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 ClassificationNode within the canonical NodeType ClassificationScheme. A Registry 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 canonical values are defined for the NodeType ClassificationScheme:

- ***UniqueCode***: This value indicates that each node of the taxonomy has a unique code assigned to it.
- ***EmbeddedPath***: This value indicates that the unique code assigned to each node of the taxonomy also encodes its path. This is the case in the NAICS taxonomy.
- ***NonUniqueCode***: In some cases nodes are not unique, and it is necessary to use the full path (from ClassificationScheme to the node of interest) in order to identify the node. 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.

4.2 Class ClassificationNode

Super Classes: RegistryObject

ClassificationNode instances are used to define tree structures where each node in the tree is a ClassificationNode. Such ClassificationScheme trees are constructed with ClassificationNode instances under a ClassificationScheme instance, and are used to define Classification schemes or ontologies.

4.2.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
parent	ObjectRef	No		Client	No
code	LongName	No		Client	No
path	String	No		Registry	No

4.2.2 Attribute parent

Each ClassificationNode MAY have a *parent* attribute. The parent attribute either references a parent ClassificationNode or a ClassificationScheme instance in case of first level ClassificationNode instances.

4.2.3 Attribute code

Each ClassificationNode MAY have a *code* attribute. The code attribute contains a code within a standard coding scheme. The code attribute of a ClassificationNode MUST be unique with respect to all sibling ClassificationNodes that are immediate children of the same parent ClassificationNode or ClassificationScheme.

4.2.4 Attribute path

Each ClassificationNode MAY have a *path* attribute. A registry MUST set the path attribute for any ClassificationNode that has a non-null code attribute value, when the ClassificationNode is retrieved from the registry. The path attribute MUST be ignored by the registry when it is specified by the client at the time the object is submitted to the registry. The path attribute contains the canonical path from the root ClassificationScheme or ClassificationNode within the hierarchy of this ClassificationNode as defined by the parent attribute. The path attribute of a ClassificationNode MUST be unique within a registry. The path syntax is defined in 4.2.5.

4.2.5 Canonical Path Syntax

The path attribute of the ClassificationNode class contains an absolute path in a canonical representation that uniquely identifies the path leading from the root ClassificationScheme or ClassificationNode to that ClassificationNode.

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

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

```

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

4.2.5.1 Example of Canonical Path Representation

The following canonical path represents what the *path* attribute would contain for the ClassificationNode with code “United States” in the sample Geography scheme in section 4.2.5.2.

```
/Geography-id/NorthAmerica/UnitedStates
```

4.2.5.2 Sample Geography Scheme

Note that in the following examples, the *id* attributes have been chosen for ease of readability and are therefore not valid id values.

```
<ClassificationScheme id='Geography-id' name="Geography”/>

<ClassificationNode id="NorthAmerica-id" parent="Geography-id"
code=NorthAmerica" />
<ClassificationNode id="UnitedStates-id" parent="NorthAmerica-id"
code="UnitedStates" />

<ClassificationNode id="Asia-id" parent="Geography-id"
code="Asia" />
<ClassificationNode id="Japan-id" parent="Asia-id" code="Japan" />
<ClassificationNode id="Tokyo-id" parent="Japan-id"
code="Tokyo" />
```

4.3 Class Classification

Super Classes: RegistryObject

A Classification instance classifies a RegistryObject instance by referencing a node defined within a particular ClassificationScheme. An internal Classification will always reference the node directly, by its id, while an external Classification will reference the node indirectly by specifying a representation of its value that is unique within the external classification scheme.

The attributes for the Classification class are intended to allow for representation of both internal and external classifications in order to minimize the need for a submission or a query to distinguish between internal and external classifications.

In Figure 6, Classification instances are not explicitly shown but are implied as associations between the RegistryObject instances (shaded leaf node) and the associated ClassificationNode.

4.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classificationScheme	ObjectRef	for external classifications	null	Client	No
classificationNode	ObjectRef	for internal classifications	null	Client	No
classifiedObject	ObjectRef	Yes		Client	No
nodeRepresentation	LongName	for external classifications	null	Client	No

Note that attributes inherited from the super classes of this class are not shown.

4.3.2 Attribute classificationScheme

If the Classification instance represents an external classification, then the *classificationScheme* attribute is required. The classificationScheme value MUST reference a ClassificationScheme instance.

4.3.3 Attribute classificationNode

If the Classification instance represents an internal classification, then the *classificationNode* attribute is required. The *classificationNode* value MUST reference a ClassificationNode instance.

4.3.4 Attribute classifiedObject

For both internal and external classifications, the *classifiedObject* attribute is required and it references the RegistryObject instance that is classified by this Classification.

4.3.5 Attribute nodeRepresentation

If the Classification instance represents an external classification, then the *nodeRepresentation* attribute is required. It is a representation of a taxonomy element from a classification scheme. It is the responsibility of the registry to distinguish between different types of *nodeRepresentation*, like between the classification scheme node code and the classification scheme node canonical path. This allows the client to transparently use different syntaxes for *nodeRepresentation*.

4.3.6 Context Sensitive *Classification*

Consider the case depicted in Figure 9 where a Collaboration Protocol Profile for ACME Inc. is classified by the “Japan” ClassificationNode under the “Geography” Classification scheme. In the absence of the context for this Classification its meaning is ambiguous. Does it mean that ACME is located in Japan, or does it mean that ACME ships products to Japan, or does it have some other meaning? To address this ambiguity a Classification MAY optionally be associated with another ClassificationNode (in this example named isLocatedIn) that provides the missing context for the Classification. Another Collaboration Protocol Profile for MyParcelService MAY be classified by the “Japan” ClassificationNode where this Classification is associated with a different ClassificationNode (e.g., named shipsTo) to indicate a different context than the one used by ACME Inc.

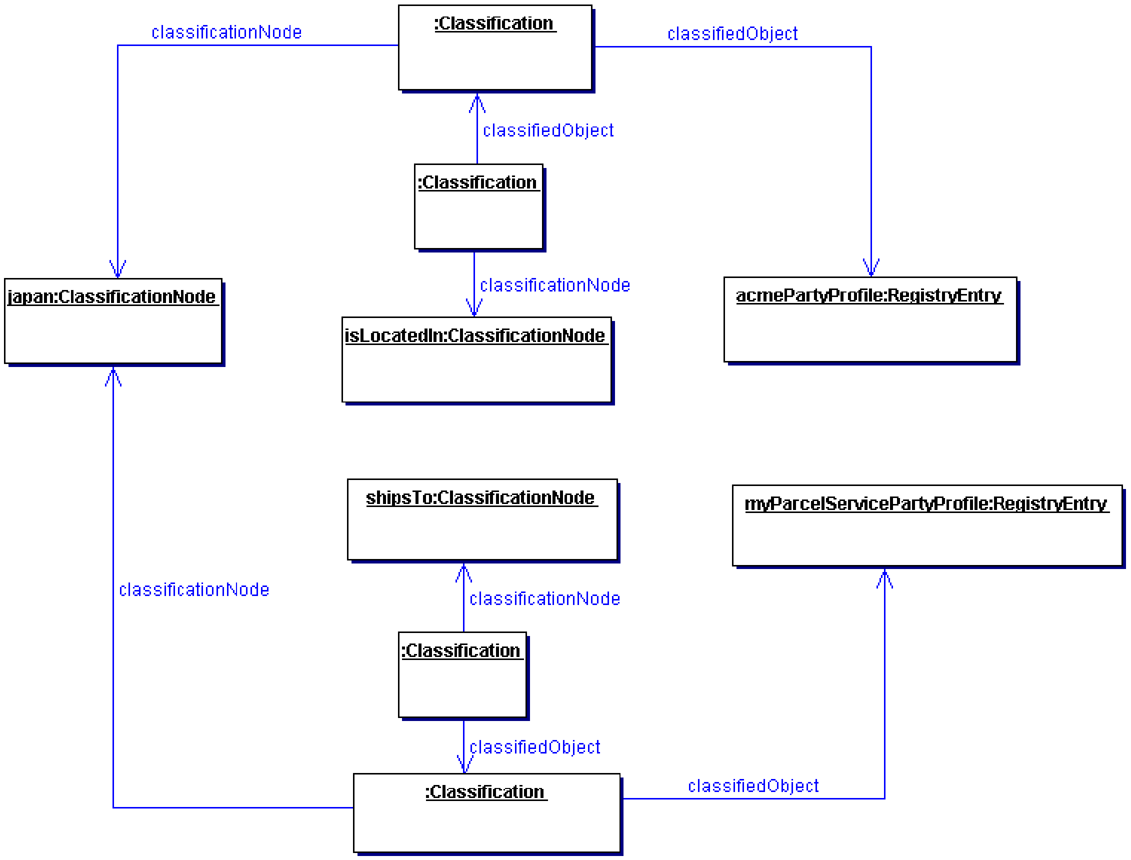


Figure 9: Context Sensitive *Classification*

Thus, in order to support the possibility of Classification within multiple contexts, a Classification is itself classified by any number of Classifications that bind the first Classification to ClassificationNodes that provide the missing contexts.

In summary, the generalized support for *Classification* schemes in the information model allows:

- A RegistryObject to be classified by defining an internal Classification that associates it with a ClassificationNode in a ClassificationScheme.
- A RegistryObject to be classified by defining an external Classification that associates it with a value in an external ClassificationScheme.
- A RegistryObject to be classified along multiple facets by having multiple Classifications that associate it with multiple ClassificationNodes or value within a ClassificationScheme.
- A Classification defined for a RegistryObject to be qualified by the contexts in which it is being classified.

4.4 Example of Classification Schemes

The following table lists some examples of possible ClassificationSchemes enabled by the information model. These schemes are based on a subset of contextual concepts identified by the ebXML Business

Process and Core Components Project Teams. This list is meant to be illustrative not prescriptive.

Classification Scheme	Usage Example	Standard Classification Schemes
Industry	Find all Parties in Automotive industry	NAICS
Process	Find a ServiceInterface that implements a Process	
Product / Services	Find a Business that sells a product or offers a service	UNSPSC
Locale	Find a Supplier located in Japan	ISO 3166
Temporal	Find Supplier that can ship with 24 hours	
Role	Find All Suppliers that have a Role of “Seller”	

Table 1: Sample Classification Schemes

5 Provenance Information Model

This chapter describes the classes that enable the description of the parties responsible for creating, publishing, or maintaining a RegistryObject or RepositoryItem. This includes information about:

- The registered user that is the submitter of a RegistryObject or RepositoryItem.
- The organization that is the submitter submitted the object on behalf of (Submitting Organization)
- The organization that is responsible for the maintainence of the submitted object (Responsible Organization)
- Any other persons that have some relationship with the submitted object

5.1 Class Person

Super Classes: RegistryObject

Person instances represent persons or humans.

5.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
addresses	Set of PostalAddress	No		Client	Yes
emailAddresses	Set of EmailAddress	No		Client	Yes
personName	PersonName	No		Client	No
telephoneNumbers	Set of TelephoneNumber	No		Client	Yes

5.1.2 Attribute addresses

Each Person instance MAY have an attribute addresses that is a Set of PostalAddress instances. Each PostalAddress provides a postal address for that user. A Person SHOULD have at least one PostalAddress.

5.1.3 Attribute emailAddresses

Each Person instance MAY have an attribute emailAddresses that is a Set of EmailAddress instances. Each EmailAddress provides an email address for that person. A Person SHOULD have at least one EmailAddress.

5.1.4 Attribute personName

Each Person instance MAY have a *personName* attribute that provides the name for that user.

5.1.5 Attribute telephoneNumbers

Each Person instance MAY have a *telephoneNumbers* attribute that contains the Set of TelephoneNumber instances defined for that user. A Person SHOULD have at least one TelephoneNumber.

5.2 Class User

Super Classes: Person

User instances represent users that have registered with a registry. User instances are also used in an AuditableEvent to keep track of the identity of the requestor that sent the request that generated the

AuditableEvent. User class is a sub-class of Person class that inherits all attributes of the Person class and does not add any new attributes.

5.2.1 Associating Users With Organizations

A user MAY be affiliated with zero or more organizations. Each such affiliation is modeled in ebRIM using an Association instance between a User instance and an Organization instance. The associationType in such cases SHOULD be either the canonical “AffiliatedWith” associationType or a ClassificationNode that is a descendant of the ClassificationNode representing the canonical “AffiliatedWith” associationType.

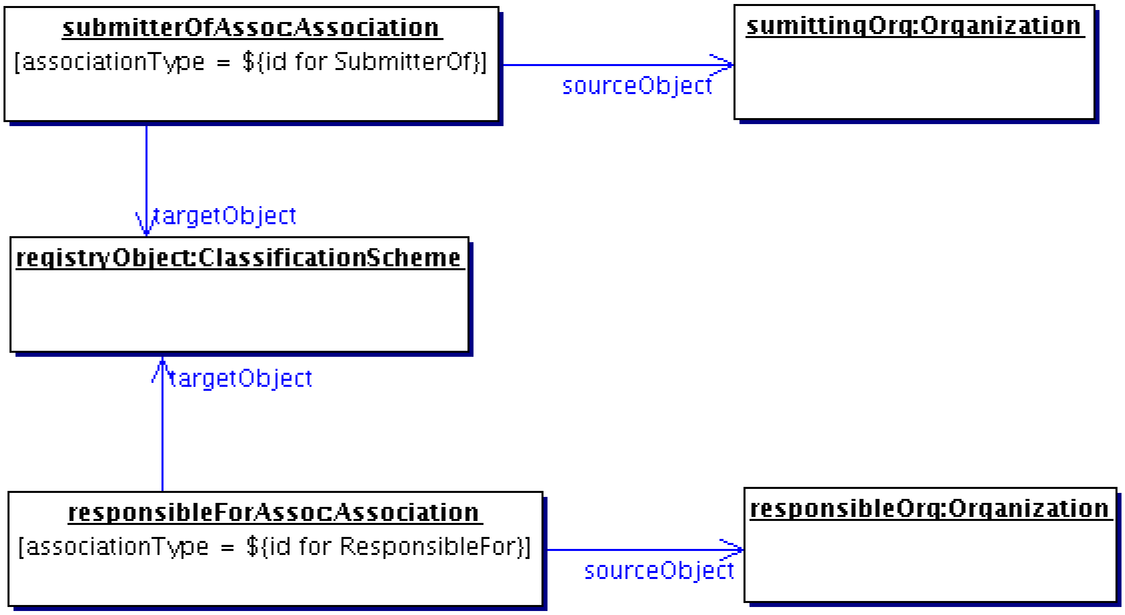

submitterOfAssocAssociation
[associationType = ${id for SubmitterOf}]
sourceObject
sumittingOrg:Organization
targetObject
registryObject:ClassificationScheme
targetObject
responsibleForAssocAssociation
[associationType = ${id for ResponsibleFor}]
sourceObject
responsibleOrg:Organization

Figure 10: User Affiliation With Organization Instance Diagram

5.3 Class Organization

Super Classes: RegistryObject

Organization instances provide information on organizations such as a Submitting Organization. Each Organization instance MAY have a reference to a parent Organization.

5.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
addresses	Set of PostalAddress	No		Client	Yes
emailAddresses	Set of EmailAddress	No		Client	Yes
parent	ObjectRef	No		Client	Yes
primaryContact	ObjectRef	No		Client	No
telephoneNumbers	Set of TelephoneNumber	No		Client	Yes

5.3.2 Attribute addresses

Each Organization instance MAY have an *addresses* attribute that is a Set of PostalAddress instances.

Each PostalAddress provides a postal address for that organization. An Organization SHOULD have at least one PostalAddress.

5.3.3 Attribute emailAddresses

Each Organization instance MAY have an attribute *emailAddresses* that is a Set of EmailAddress instances. Each EmailAddress provides an email address for that Organization. An Organization SHOULD have at least one EmailAddress.

5.3.4 Attribute parent

Each Organization instance MAY have a *parent* attribute that references the parent Organization instance, if any, for that organization.

5.3.5 Attribute primaryContact

Each Organization instance SHOULD have a *primaryContact* attribute that references the Person instance for the person that is the primary contact for that organization.

5.3.6 Attribute telephoneNumbers

Each Organization instance MUST have a *telephoneNumbers* attribute that contains the Set of TelephoneNumber instances defined for that organization. An Organization SHOULD have at least one telephone number.

5.4 Associating Organizations With RegistryObjects

An organization MAY be associated with zero or more RegistryObject instances. Each such association is modeled in ebRIM using an Association instance between an Organization instance and a RegistryObject instance. The associationType in such cases MAY be (but is not restricted to) either the canonical “SubmitterOf” associationType or the canonical “ResponsibleFor” associationType. The “SubmitterOf” associationType indicates the organization that submitted the RegistryObject (via a User). The “ResponsibleFor” associationType indicates the organization that is designated as the organization responsible for the ongoing maintenance of the RegistryObject.

Associations between Organizations and RegistryObjects do not entitle organizations to any special privileges with respect to the RegistryObject. Such privileges are defined by the Access Control Policies defined for the RegistryObject as described in chapter 9.

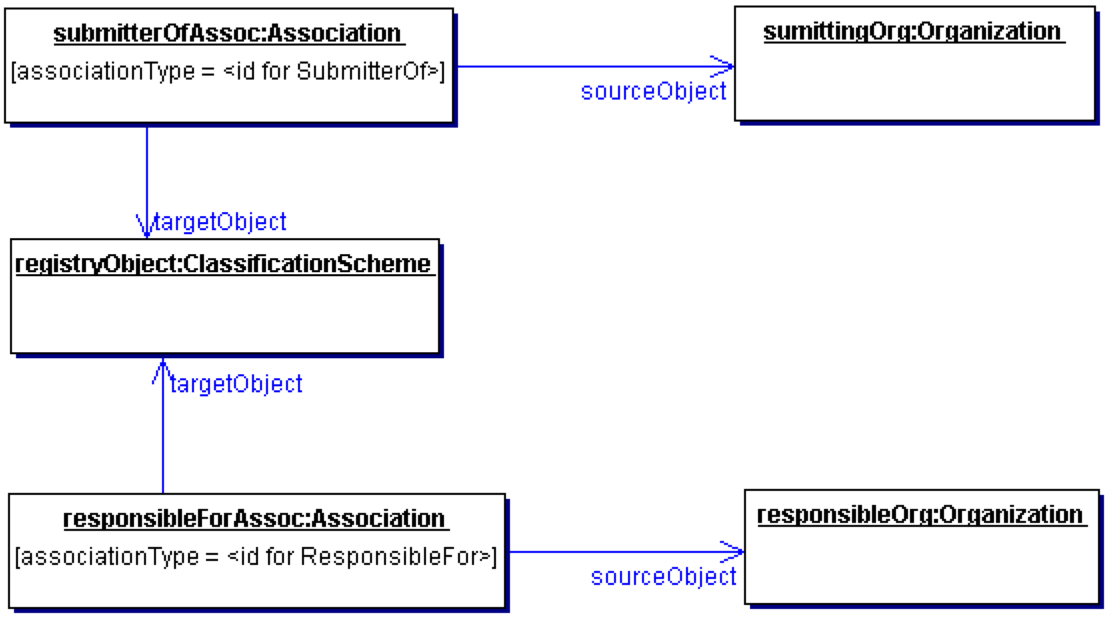

submitterOfAssoc:Association
[associationType = <id for SubmitterOf>]
sumittingOrg:Organization
sourceObject
targetObject
registryObject:ClassificationScheme
targetObject
responsibleForAssoc:Association
[associationType = <id for ResponsibleFor>]
responsibleOrg:Organization
sourceObject

Figure 11: Organization to RegistryObject Association Instance Diagram

5.5 Class PostalAddress

PostalAddress defines attributes of a postal address.

5.5.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
city	ShortName	No		Client	Yes
country	ShortName	No		Client	Yes
postalCode	ShortName	No		Client	Yes
slots	Set of Slot	No		Client	Yes
stateOrProvince	ShortName	No		Client	Yes
street	ShortName	No		Client	Yes
streetNumber	String32	No		Client	Yes

5.5.2 Attribute city

Each PostalAddress MAY have a *city* attribute identifying the city for that address.

5.5.3 Attribute country

Each PostalAddress MAY have a *country* attribute identifying the country for that address.

5.5.4 Attribute postalCode

Each PostalAddress MAY have a *postalCode* attribute identifying the postal code (e.g., zip code) for that address.

5.5.5 Attribute stateOrProvince

Each PostalAddress MAY have a *stateOrProvince* attribute identifying the state, province or region for that address.

5.5.6 Attribute street

Each PostalAddress MAY have a *street* attribute identifying the street name for that address.

5.5.7 Attribute streetNumber

Each PostalAddress MAY have a *streetNumber* attribute identifying the street number (e.g., 65) for the street address.

5.6 Class TelephoneNumber

This class defines attributes of a telephone number.

5.6.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
areaCode	String8	No		Client	Yes
countryCode	String8	No		Client	Yes
extension	String8	No		Client	Yes
number	String16	No		Client	Yes
phoneType	ObjectRef	No		Client	Yes

5.6.2 Attribute areaCode

Each TelephoneNumber instance MAY have an *areaCode* attribute that provides the area code for that telephone number.

5.6.3 Attribute countryCode

Each TelephoneNumber instance MAY have a *countryCode* attribute that provides the country code for that telephone number.

5.6.4 Attribute extension

Each TelephoneNumber instance MAY have an *extension* attribute that provides the extension number, if any, for that telephone number.

5.6.5 Attribute number

Each TelephoneNumber instance MAY have a *number* attribute that provides the local number (without area code, country code and extension) for that telephone number.

5.6.6 Attribute phoneType

Each TelephoneNumber instance MAY have a *phoneType* attribute that provides the type for the TelephoneNumber. The value of the phoneType attribute MUST be a reference to a ClassificationNode in the canonical PhoneType ClassificationScheme.

5.7 Class EmailAddress

This class defines attributes of an email address.

5.7.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
address	ShortName	Yes		Client	Yes
slots	Set of Slot	No		Client	Yes
type	ObjectRef	No		Client	Yes

5.7.2 Attribute address

Each EmailAddress instance MUST have an *address* attribute that provides the actual email address.

5.7.3 Attribute type

Each EmailAddress 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 as referenced in appendix .

5.8 Class PersonName

This class defines attributes for a person's name.

5.8.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
firstName	ShortName	No		Client	Yes
lastName	ShortName	No		Client	Yes
middleName	ShortName	No		Client	Yes
slots	Set of Slot	No		Client	Yes

5.8.2 Attribute firstName

Each PersonName SHOULD have a *firstName* attribute that is the first name of the person.

5.8.3 Attribute lastName

Each PersonName SHOULD have a *lastName* attribute that is the last name of the person.

5.8.4 Attribute middleName

Each PersonName SHOULD have a *middleName* attribute that is the middle name of the person.

6 Service Information Model

This chapter describes the classes in the information model that support the registration of service descriptions. The service information model is flexible and supports the registration of web services as well as other types of services.

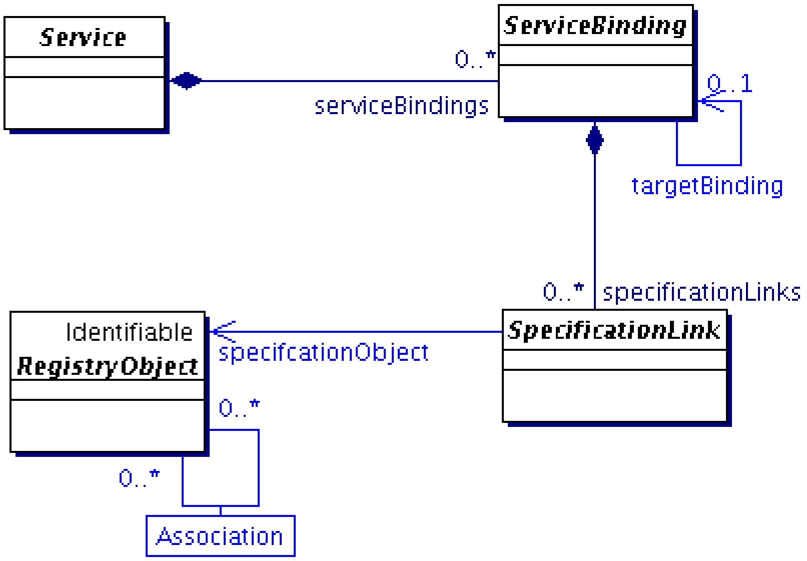


Figure 12: Service Information Model

6.1 Class Service

Super Classes: RegistryObject

Service instances describe services, such as web services.

6.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
serviceBindings	Set of ServiceBinding	Yes, Set may be empty		Client	Yes

6.1.2 Attribute serviceBindings

A Service MAY have a *serviceBindings* attribute that defines the service bindings that provide access to that Service.

6.2 Class ServiceBinding

Super Classes: RegistryObject

ServiceBinding instances are RegistryObjects that represent technical information on a specific way to access a Service instance. An example is where a ServiceBinding is defined for each protocol that may be used to access the service.

6.2.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
accessURI	URI	No		Client	Yes
service	ObjectRef	Yes		Client	No
specificationLinks	Set of SpecificationLink	Yes, Set may be empty		Client	Yes
targetBinding	ObjectRef	No		Client	Yes

6.2.2 Attribute accessURI

A ServiceBinding MAY have an *accessURI* attribute that defines the URI to access that ServiceBinding. This attribute is ignored if a targetBinding attribute is specified for the ServiceBinding. If the URI is a URL then a registry MUST validate the URL to be resolvable at the time of submission before accepting a ServiceBinding submission to the registry.

6.2.3 Attribute service

A ServiceBinding MUST have a *service* attribute whose value MUST be the id of its parent Service.

6.2.4 Attribute specificationLinks

A ServiceBinding MAY have a *specificationLinks* attribute defined that is a Set of references to SpecificationLink instances. Each SpecificationLink instance links the ServiceBinding to a particular technical specification that MAY be used to access the Service for the ServiceBinding.

6.2.5 Attribute targetBinding

A ServiceBinding MAY have a *targetBinding* attribute defined that references another ServiceBinding. A targetBinding MAY be specified when a service is being redirected to another service. This allows the rehosting of a service by another service provider.

6.3 Class SpecificationLink

Super Classes: RegistryObject

A SpecificationLink provides the linkage between a ServiceBinding and one of its technical specifications that describes how to use the service using the ServiceBinding. For example, a ServiceBinding MAY have SpecificationLink instances that describe how to access the service using a technical specification such as a WSDL document or a CORBA IDL document.

6.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
serviceBinding	ObjectRef	Yes		Client	No
specificationObject	ObjectRef	Yes		Client	Yes
usageDescription	InternationalString	No		Client	Yes
usageParameters	Bag of FreeFormText	No		Client	Yes

6.3.2 Attribute serviceBinding

A SpecificationLink instance MUST have a *serviceBinding* attribute that provides a reference to its parent ServiceBinding instances. It value MUST be the id of the parent ServiceBinding object.

6.3.3 Attribute specificationObject

A SpecificationLink instance MUST have a *specificationObject* attribute that provides a reference to a RegistryObject instance that provides a technical specification for the parent ServiceBinding. Typically, this is an ExtrinsicObject instance representing the technical specification (e.g., a WSDL document). It may also be an ExternalLink object in case the technical specification is a resource that is external to the registry.

6.3.4 Attribute usageDescription

A SpecificationLink instance MAY have a *usageDescription* attribute that provides a textual description of how to use the optional usageParameters attribute described next. The usageDescription is of type InternationalString, thus allowing the description to be in multiple languages.

6.3.5 Attribute usageParameters

A SpecificationLink instance MAY have a *usageParameters* attribute that provides a Bag of Strings representing the instance specific parameters needed to use the technical specification (e.g., a WSDL document) specified by this SpecificationLink object.

7 Event Information Model

This chapter defines the information model classes that support the registry Event Notification feature. These classes include AuditableEvent, Subscription, Selector and Action. They constitute the foundation of the Event Notification information model.

Figure 13 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.

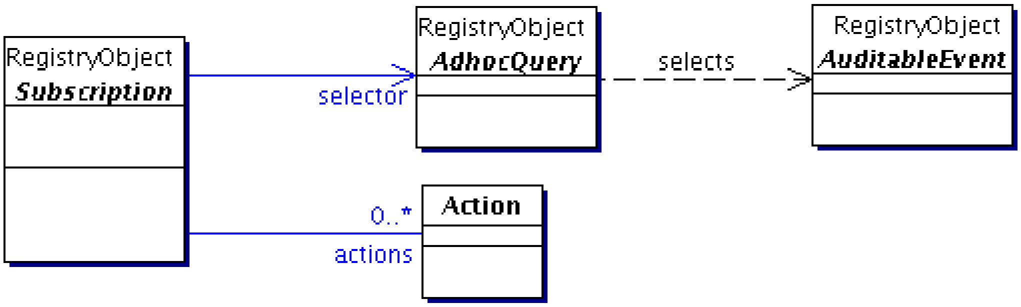


Figure 13: Event Information Model

7.1 Class AuditableEvent

Super Classes: RegistryObject

AuditableEvent instances provide a long-term record of events that effected a change in a RegistryObject. A RegistryObject is associated with an ordered Set of AuditableEvent instances that provide a complete audit trail for that RegistryObject.

AuditableEvents are usually a result of a client-initiated request. AuditableEvent instances are generated by the Registry Service to log such Events.

Often such events effect a change in the life cycle of a RegistryObject. For example a client request could Create, Update, Deprecate or Delete a RegistryObject. An AuditableEvent is typically created when a request creates or alters the content or ownership of a RegistryObject. Read-only requests typically do not generate an AuditableEvent.

7.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
eventType	ObjectRef	Yes		Registry	No
affectedObjects	Set of ObjectRef	Yes		Registry	No
requestId	URI	Yes		Registry	No
timestamp	dateTime	Yes		Registry	No
user	ObjectRef	Yes		Registry	No

7.1.2 Attribute eventType

Each AuditableEvent MUST have an *eventType* attribute which identifies the type of event recorded by the AuditableEvent. 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 canonical EventType ClassificationScheme. The canonical EventType ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to the canonical EventType ClassificationScheme.

7.1.2.1 Pre-defined Auditable Event Types

The following table lists pre-defined auditable event types. A Registry MUST support the event types listed below. A Registry MAY support additional event types as long as they are ClassificationNodes within the canonical EventType ClassificationScheme.

Name	Description
Approved	An Event that marks the approval of a RegistryObject.
Created	An Event that marks the creation of a RegistryObject.
Deleted	An Event that marks the deletion of a RegistryObject.
Deprecated	An Event that marks the deprecation of a RegistryObject.
Downloaded	An Event that marks the downloading of a RegistryObject.
Relocated	An Event that marks the relocation of a RegistryObject.
Undeprecated	An Event that marks the undeprecation of a RegistryObject.
Updated	An Event that that marks the updating of a RegistryObject.
Versioned	An Event that that marks the creation of a new version of a RegistryObject.

7.1.3 Attribute affectedObjects

Each AuditableEvent MUST have an *affectedObjects* attribute that identifies the Set of RegistryObjects instances that were affected by this event.

7.1.4 Attribute requestId

Each AuditableEvent MUST have a *requestId* attribute that identifies the client request instance that affected this event.

7.1.5 Attribute timestamp

Each AuditableEvent MUST have a *timestamp* attribute that records the date and time that this event occurred.

7.1.6 Attribute user

Each AuditableEvent MUST have a *user* attribute that identifies the User that sent the request that generated this event affecting the RegistryObject instance.

7.2 Class Subscription

Super Classes: RegistryObject

Subscription instances are RegistryObjects that define a User's interest in certain types of AuditableEvents. A User MAY create a subscription with a registry if he or she wishes to receive notification for a specific type of event.

7.2.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
actions	Set of Action	Yes, may be empty		Client	Yes
endTime	dateTime	No		Client	Yes
notificationInterval	duration	No	P1D (1 day)	Client	No
selector	ObjectRef	Yes		Client	No
startTime	dateTime	No	Current time	Client	Yes

7.2.2 Attribute actions

A Subscription instance MUST have an *actions* attribute that is a Set of zero or more Action instances. An Action instance describes what action the registry must take when an event matching the Subscription transpires. The Action class is described in section 7.5.

7.2.3 Attribute endTime

This attribute denotes the time after which the subscription expires and is no longer active. If this attribute is missing the subscription never expires.

7.2.4 Attribute notificationInterval

This attribute denotes the duration that a registry MUST wait between delivering successive notifications to the client. The client specifies this attribute in order to control the frequency of notification communication between registry and client.

7.2.5 Attribute selector

This attribute defines the selection criteria that determine which events match this Subscription and are of interest to the User. The *selector* attribute references a pre-defined query that is stored in the registry as an instance of the AdhocQuery class. This AdhocQuery instance specifies or "selects" events that are of interest to the subscriber. The AdhocQueryClass is described in section 7.3.

7.2.5.1 Specifying Selector Query Parameters

The selector query MAY be configured as a paremeterized stored query as defined by [ebRS]. A Subscription MUST specify the parameters values for stored parameterized queries as Slots as defined in section title "Specifying Query Invocation Parameters" in [ebRS]. These parameter value Slots if specified MUST be specified on the Subscription object.

7.2.6 Attribute startTime

This attribute denotes the time at which the subscription becomes active. If this attribute is missing subscription starts immediately.

7.3 Class AdhocQuery

Super Classes: RegistryObject

The AdhocQuery class is a container for an ad hoc query expressed in a query syntax that is supported by an ebXML Registry. Instances of this class MAY be used for discovery of RegistryObjects within the registry. Instances of AdhocQuery MAY be stored in the registry like other RegistryObjects. Such stored AdhocQuery instances are similar in purpose to the concept of stored procedures in relational databases.

7.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
queryExpression	QueryExpression	Required when defining a new AdhocQuery. Not required when invoking a stored query.		Client	No

7.3.2 Attribute queryExpression

Each AdhocQuery instance MAY have a *queryExpression* attribute that contains the query expression for the AdhocQuery depending upon the use case as follows. When an AdhocQuery is submitted to the registry it MUST contain a queryExpression. When a stored AdhocQuery is included in an AdhocQueryRequest to invoke a stored query as defined by the stored query feature defined in [ebRS] it SHOULD NOT contain a queryExpression.

7.4 Class QueryExpression

The QueryExpression class is an extensible wrapper that can contain a query expression in any supported query syntax such as SQL or Filter Query syntax.

7.4.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
queryLanguage	ObjectRef	Required		Client	No
<any>	anyType	Required		Client	No

7.4.2 Attribute queryLanguage

The queryLanguage attribute specifies the query language that the query expression conforms to. The value of this attribute MUST be a reference to a ClassificationNode within the canonical QueryLanguage ClassificationScheme. A Registry MUST support the query languages as defined by the canonical QueryLanguage ClassificationScheme. The canonical QueryLanguage ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it to allow a registry to support additional query language syntaxes.

7.4.3 Attribute <any>

This attribute is extensible and therefor MAY be of any type depending upon the queryLanguage specified. For SQL queryLanguage it MUST be an SQL query string. For Filter query it MUST be a FilterQueryType defined by [RR-QUERY-XSD].

7.5 Class Action

The Action class is an abstract super class that specifies what the registry must do when an event matching the action’s Subscription transpires. A registry uses Actions within a Subscription to asynchronously deliver event Notifications to the subscriber.

If no Actions are defined within the Subscription it implies that the user does not wish to be notified asynchronously by the registry and instead intends to periodically poll the registry and pull the pending Notifications.

This class does not currently define any attributes.

7.6 Class NotifyAction

Super Classes: Action

The NotifyAction class is a sub-class of Action class. An instance of NotifyAction represents an Action that the registry MUST perform in order to notify the subscriber of a Subscription of the events of interest to that subscriber.

7.6.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
endPoint	URI	YES		Client	
notificationOption	ObjectRef	No	Reference to ObjectRefs ClassificationNode	Client	Yes

7.6.2 Attribute endPoint

This attribute specifies a URI that identifies a service end point that MAY be used by the registry to deliver notifications. Currently this attribute can either be a “mailto” URI (e.g. mailto:someone@acme.com) or a “urn:uuid” URI.

If endpoint is a “mailto” URI then the registry MUST use the specified email address to deliver the notification via email. Email configuration parameters such as the “from” email address and SMTP server configuration MAY be specified in a registry specific manner.

If endpoint is a “urn:uuid” URI then it MUST be a reference to a ServiceBinding object to a Service that implements the RegistryClient interface as defined by [ebRS]. In this case the registry MUST deliver the notification by web service invocation as defined by the ServiceBinding object.

7.6.3 Attribute notificationOption

This attribute controls the specific type of event notification content desired by the subscriber. It is used by the subscriber to control the granularity of event notification content communicated by the registry to the subscriber. The value of the notificationOption attribute MUST be a reference to a ClassificationNode within the canonical NotificationOptionType ClassificationScheme. A Registry MUST support the notificationOption types as defined by the NotificationOptionType ClassificationScheme. The canonical NotificationOptionType ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it.

7.6.3.1 Pre-defined notificationOption Values

The following canonical values are defined for the NotificationOptionType ClassificationScheme:

Name	Description
ObjectRefs	Indicates that the subscriber wants to receive only references to RegistryObjects that match the Subscription within a notification.
Objects	Indicates that the subscriber wants to receive actual RegistryObjects that match the Subscription within a notification.

7.7 Class Notification

Super Classes: RegistryObject

The Notification class represents a Notification from the registry regarding an event that matches a Subscription. A registry may uses a Notification instance to notify a client of an event that matches a Subscription they have registered. This is a *push* model of notification. A client may also *pull* events from the registry using the AdhocQuery protocol defined by [ebRS].

7.7.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
subscription	ObjectRef	YES		Registry	No
registryObjectList	Set of Identifiable	No		Registry	No

7.7.2 Attribute subscription

This attribute specifies a reference to a Subscription instance within the registry. This is the Subscription that matches the event for which this Notification is about.

7.7.3 Attribute registryObjectList

This attribute specifies a Set of ObjectRefs or a Set of RegistryObject instances that represent the objects that were impacted by the event that matched the Subscription. The registry MUST include ObjectRef or RegistryObject instances as Set elements depending upon the notificationOption specified for the Subscription.

8 Cooperating Registries Information Model

This chapter describes the classes in the information model that support the cooperating registries capability defined by [ebRS].

8.1 Class Registry

Super Classes: RegistryObject

Registry instances are used to represent a single physical OASIS ebXML Registry.

8.1.0.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
catalogingLatency	duration	No	P1D (1 day)	Registry	Yes
conformanceProfile	String16	No	"registry Lite"	Registry	Yes
operator	ObjectRef	Yes		Registry	Yes
replicationSyncLatency	duration	No	P1D (1 day)	Registry	Yes
specificationVersion	Sring8	Yes		Registry	Yes

8.1.1 Attribute catalogingLatency

Each Registry instance MAY have an attribute named *catalogingLatency* that specifies the maximum latency between the time a submission is made to the registry and the time it gets cataloged by any cataloging services defined for the objects within the submission.

8.1.2 Attribute conformanceProfile

Each Registry instance MAY have an attribute named *conformanceProfile* that declares the conformance profile that the registry supports. The conformance profiles choices are "registryLite" and "registryFull" as defined by [ebRS].

8.1.3 Attribute operator

Each Registry instance MUST have an attribute named *operator* that is a reference to the Organization instance representing the organization for the registry'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.

8.1.4 Attribute replicationSyncLatency

Each Registry 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 registry gets updated to synchronize with the new state of the original object.

8.1.5 Attribute specificationVersion

Each Registry instance MUST have an attribute named *specificationVersion* that is the version of the ebXML Registry Services Specification [ebRS].

8.2 Class Federation

Super Classes: RegistryObject

Federation instances are used to represent a registry federation.

8.2.0.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
replicationSyncLatency	duration	No	P1D (1 day)	Client	Yes

8.2.1 Attribute replicationSyncLatency

Each Federation 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.

8.2.2 Federation Configuration

A federation is created by the creation of a Federation instance. Membership of a registry within a federation is established by creating an Association between the Registry instances for the registry seeking membership with the Federation 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 Figure 14.

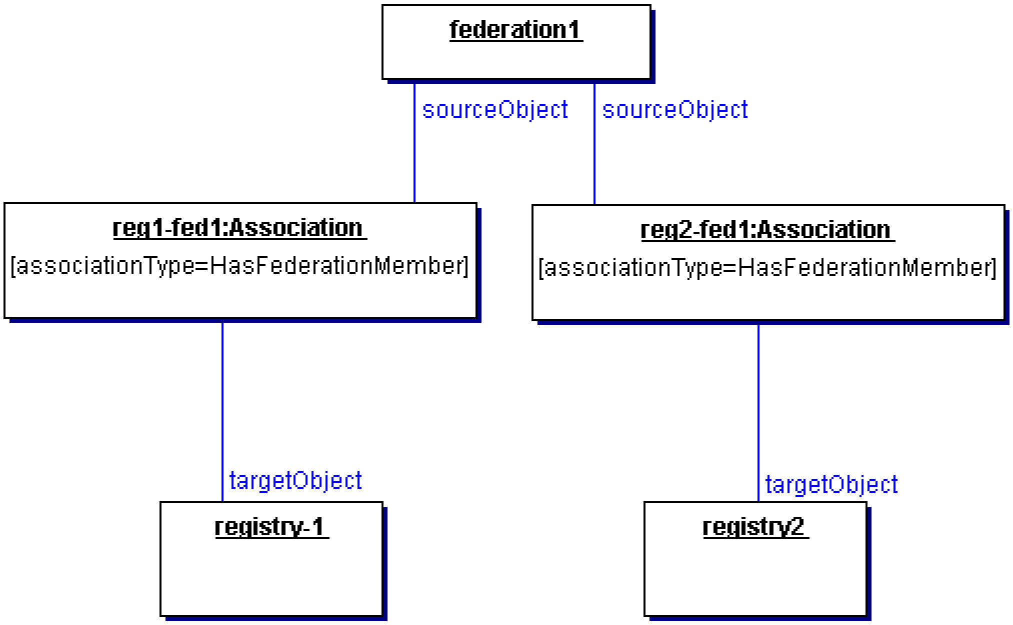


Figure 14: Federation Information Model

9 Access Control Information Model

This chapter defines the Access Control Information Model used by the registry to control access to RegistryObjects and RepositoryItems managed by it. The Access Control features of the registry require that it function as both a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP) as defined in [XACML].

This specification first defines an abstract Access Control Model that enables access control policies to be defined and associated with RegistryObjects.

Next, it defines a normative and required binding of that abstract model to [XACML] .

Finally, it defines how a registry MAY support additional bindings to custom access control technologies.

9.1 Terminology

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:

Term	Description
Access	Performing an ***action***. An example is a user performing a *delete action* on a RegistryObject.
Access Control	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.
Action	An operation on a ***resource***. An example is the *delete action* on a RegistryObject.
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: • A User instance within the registry

9.2 Use Cases for Access Control Policies

The following are some common use cases for access control policy:

9.2.1 Default Access Control Policy

Define a default access control policy that gives *read access* to any one and access to all actions to owner of the resource and Registry Administrator. This access control policy implicitly applies to any resource that does not explicitly have a custom Access Control Policy defined for it.

9.2.2 Restrict Read Access To Specified Subjects

Define a custom access control policy to restrict *read access* to a resource to specified user(s), group(s) and/or role(s).

9.2.3 Grant Update and/or Delete Access To Specified Subjects

Define a custom access control policy to grant *update* and/or *delete access* to a resource to specified user (s), group(s) and/or role(s).

9.2.4 Reference Access Control

Define a custom access control policy to restrict *reference access* to a resource to specified user(s), group(s) and/or role(s). For example a custom access control policy MAY be defined to control who can create an extramural association to a RegistryObject. Another example is to control who can add members to a RegistryPackage.

9.3 Resources

A registry MUST control access to the following types of resources:

- *RegistryObject resource* is any instance of RegistryObject class or its sub-classes. Each RegistryObject resource references an Access Control Policy that controls all access to that object.
- *RepopositoryItem resource* is any instance of RepositoryItem class. By default, access control to a RepositoryItem is managed by the same Access Control Policy as its ExtrinsicObject.

A registry MUST support the following resource attributes.

9.3.1 Resource Attribute: *owner*

The *owner* attribute of a Resource carries the value of id attribute of the User instance within the registry that represents the owner of the resource.

9.3.2 Resource Attribute: *selector*

The *selector* attribute of a Resource carries a string representing a query as define by a sub-type of AdhocQueryType in [ebRS]. The registry MUST use this query is a filter to select the resources that match it.

9.3.3 Resource Attribute: *<attribute>*

The resource attribute <attribute> represents any attribute defined by the RegistryObject type or one of its sub-types. For example, it could be the targetObject attribute in case the resource is an Association object.

9.4 Actions

A registry MUST support the following actions as operations on RegistryObject and RepositoryItem resources managed by the registry.

9.4.1 Create Action

The *create action* creates a RegistryObject or a RepositoryItem. A submitObjects operation performed on the LifeCycleManager interface of the registry result in a *create action.*

9.4.2 Read Action

The *read action* reads a RegistryObject or a RepositoryItem without having any impact on its state. An operation performed on the QueryManager interface of the registry result in a *read action.* A registry MUST first perform the query for the read action and then MUST filter out all resources matching the query for which the client does not have access for the read action.

9.4.3 Update Action

The *update action* updates or modifies the state of a RegistryObject or a RepositoryItem. An updateObjects operation performed on the LifeCycleManager interface of the registry result in a *update action.* A registry MUST evaluate access control policy decision based upon the state of the resource *before* and not the *after* performing the update action.

9.4.4 Delete Action

The *delete action* deletes a RegistryObject or a RepositoryItem. A removeObjects operation performed on the LifeCycleManager interface of the registry results in a *delete action.*

9.4.5 Approve Action

The *approve action* approves a RegistryObject. An approveObjects operation performed on the LifeCycleManager interface of the registry result in an *approve* action*.*

9.4.6 Reference Action

The *reference action* creates a reference to a RegistryObject. A submitObjects or updateObjects operation performed on the LifeCycleManager interface of the registry MAY result in a *reference action.* An example of a reference action is when an Association is created that references a RegistryObject resource as its source or target object.

9.4.7 Deprecate Action

The *deprecate action* deprecates a RegistryObject. A deprecateObjects operation performed on the LifeCycleManager interface of the registry result in a *deprecate action.*

9.4.8 Undeprecate Action

The *undeprecate action* undeprecates a previously deprecated RegistryObject. An undeprecateObjects operation performed on the LifeCycleManager interface of the registry result in an *undeprecate* action*.*

9.4.9 Action Attribute: *action-id*

This attribute identifies the specific action being performed by the subject on one or more resources. A Registry MUST support access control for all the types of actions identified in this document above.

9.4.10 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 value of this attribute MUST be the value of the id attribute for the object that is the source of the reference.

9.4.11 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 Class that the reference-source object is an instance of. The value of this attribute MUST be the name of an attribute within the RIM Class that is the Class 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.

9.5 Subjects

A registry MUST support the following Subject attributes within its Access Control Policies. In addition a registry MAY support additional subject attributes.

9.5.1 Attribute *id*

The *identity* attribute of a Subject carries the value of *id* attribute of a User instance within the registry.

9.5.2 Attribute *group*

The *group* attribute of a Subject carries the value of the code attribute of a ClassificationNode within the canonical SubjectGroup ClassificationScheme (see appendix) within the registry. A registry MUST NOT allow anyone but a subject with the canonical RegistryAdministrator role to assign roles to users.

9.5.2.1 Assigning Groups To Users

Arbitrary groups MAY be defined by extending the canonical SubjectGroup ClassificationScheme. Groups MAY be assigned to registered users by classifying their User instance with a ClassificationNode within the canonical SubjectGroup ClassificationScheme.

9.5.3 Attribute *role*

The *role* attribute of a Subject carries the value of the code attribute of a ClassificationNode within the canonical SubjectRole ClassificationScheme (see appendix) within the registry.

9.5.3.1 Assigning Roles To Users

Arbitrary roles MAY be defined by extending the canonical SubjectRole ClassificationScheme. Roles MAY be assigned to registered users by classifying their User instance with a ClassificationNode within the canonical SubjectRole ClassificationScheme. A registry MUST NOT allow anyone but a subject with the canonical RegistryAdministrator role to assign roles to users. A registry MAY use registry specific means to assign RegistryAdministrator roles.

9.6 Abstract Access Control Model

Every RegistryObject is associated with exactly one Access Control Policy that governs “who” is authorized to perform “what” action on that RegistryObject. The abstract Access Control Model allows the Access Control Policy to be defined in any arbitrary format as long as it is represented in the registry as a repositoryItem and its corresponding ExtrinsicObject. The objectType attribute of this ExtrinsicObject MUST reference a descendent of the “xacml” node (e.g. “Policy” or PolicySet”) in the canonical ObjectType ClassificationScheme. This distinguishes XACML “Policy” or PolicySet” Access Control Policy objects from other ExtrinsicObject instances.

9.6.1 Access Control Policy for a RegistryObject

A RegistryObject MAY be associated with an Access Control Policy by a special Association with the canonical associationType of AccessControlPolicyFor. This association has the reference to the ExtrinsicObject representing the Access Control Policy as the value of its sourceObject and has the reference to the RegistryObject as the value of its targetObject attribute.

If a RegistryObject 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 registry.

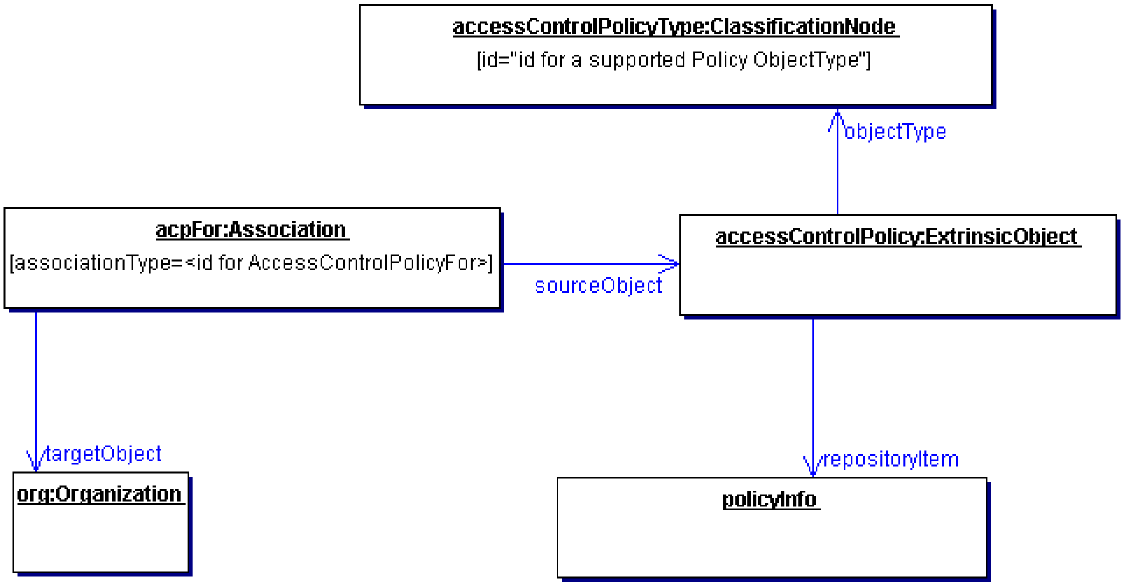

accessControlPolicyType:ClassificationNode
[id="id for a supported Policy ObjectType"]
objectType
acpFor:Association
[associationType=<id for AccessControlPolicyFor>]
sourceObject
accessControlPolicy:ExtrinsicObject
targetObject
org:Organization
repositoryItem
policyInfo

Figure 15: Instance Diagram for Abstract Access Control Information Model

Figure 15 shows an instance diagram where an Organization instance *org* references an ExtrinsicObject instance *accessControlPolicy* as its Access Control Policy object. The *accessControlPolicy* object has its objectType attribute referencing a node in the canonical ObjectType ClassificationScheme that represents a supported Access Control Policy format. The *accessControlPolicy* ExtrinsicObject has a repositoryItem defining its access control policy information in a specific format.

9.6.2 Access Control Policy for a RepositoryItem

By default, access control to a RepositoryItem is managed by the Access Control Policy associated with its ExtrinsicObject that provides metadata for the RepositoryItem. A RepositoryItem MAY have an Access Control Policy separate from its ExtrinsicObject. In such case, the Access Control Policy for the RepositoryItem is referenced via a Special Slot on its ExtrinsicObject. This special Slot has "repositoryItemACP" as its name and the id of the ExtrinsicObject representing the Access Control Policy for the RepositoryItem as its value.

9.6.3 Default Access Control Policy

A registry MUST support the default Access Control Policy.

The default Access Control Policy applies to any RegistryObject that does not explicitly have an Access Control Policy associated with it.

The following list summarizes the default Access Control Policy semantic that a registry SHOULD implement:

- Only a Registered User is granted access to create actions.
- An unauthenticated Registry Client is granted access to read actions. The Registry MUST assign the default RegistryGuest role to such Registry Clients.
- A Registered User has access to all actions on Registry Objects submitted by the Registered User.
- The Registry Administrator and Registry Authority have access to all actions on all Registry Objects.

A registry MAY have a default access control policy that differs from the above semantics.

9.6.4 Root Access Control Policy

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

As described in Figure 15, an access control policy is an ExtrinsicObject that contains a pointer to a repository item. The access control policies themselves are created, updated, and deleted.

To define who may create access control policies pertaining to specified resources, it is necessary to have one or more administrative Access Control Policies. Such policies restrict Registry Users from creating access control policies to unauthorized resources. This version of the Registry specifications defines a single Root Access Control Policy that allows all actions on Access Control Policies for a resource under the following conditions:

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

9.6.5 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.

9.7 Access Control Model: XACML Binding

A registry MAY support custom access control policies based upon a normative though optional binding of the Access Control Model to [XACML].

This section defines the normative though optional binding of the abstract Access Control Model to [XACML]. This section assumes the reader is familiar with [XACML].

This binding to [XACML] enables a flexible access control mechanism that supports access control policy definition from the simples to the most sophisticated use cases.

In this binding the policyInfo repositoryItem in the abstract Access Control Model MUST be one of the following:

- A PolicySet as defined by [XACML]
- A Policy as defined by [XACML]

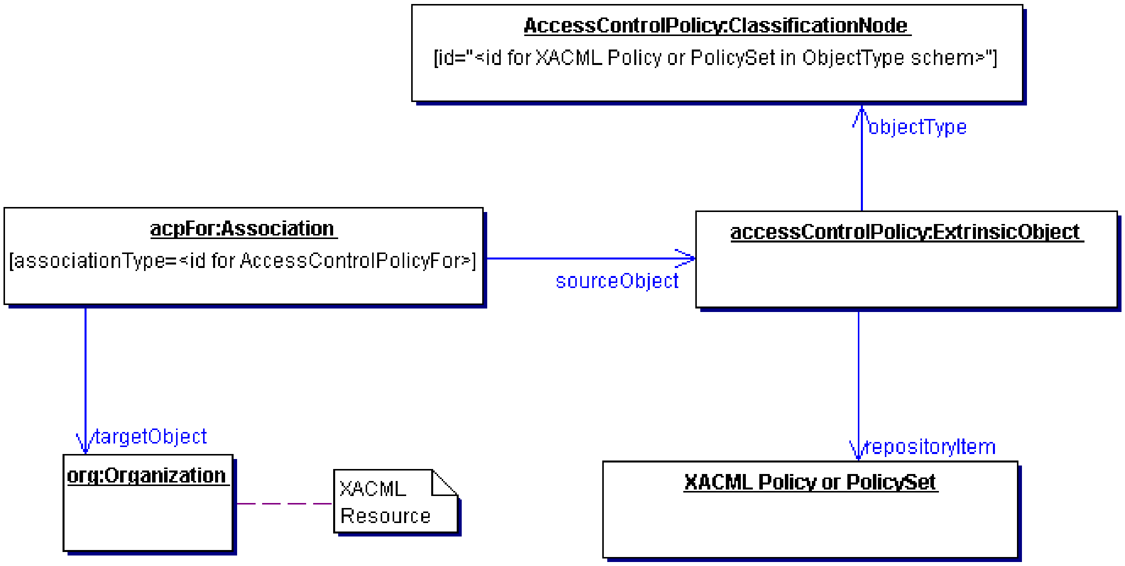

AccessControlPolicy:ClassificationNode
[id="<id for XACML Policy or PolicySet in ObjectType schem>"]
objectType
acpFor:Association
[associationType=<id for AccessControlPolicyFor>]
sourceObject
accessControlPolicy:ExtrinsicObject
targetObject
repositoryItem
org:Organization
XACML Resource
XACML Policy or PolicySet

Figure 16: Access Control Information Model: [XACML] Binding

9.7.1 Resource Binding

[XACML] defines an element called ResourceAttributeDesignator that identifies the type of resource attribute being specified in a ResourceMatch or Apply element.

The resource attributes defined by the abstract Access Control Model map to the following ResourceAttributeDesignator definitions:

Resource Attribute	AttributeId	DataType
owner	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:owner	`http://www.w3.org/2001/XMLSchema#anyURI`
selector	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:selector	`http://www.w3.org/2001/XMLSchema#string`
<attribute>	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:<attribute>	`Depends upon the specific attribute.`

Table 2: Resource Binding to [XACML]

Data Type	XACML Data Type Identifier URI	Description
Boolean	http://www.w3.org/2001/XMLSchema#boolean	
String	http://www.w3.org/2001/XMLSchema#string	Used strings of all lengths
ObjectRef	http://www.w3.org/2001/XMLSchema#anyURI	
URI	http://www.w3.org/2001/XMLSchema#anyURI	
Integer	http://www.w3.org/2001/XMLSchema#integer	
DateTime	http://www.w3.org/2001/XMLSchema#dateTime	

9.7.2 Action Binding

[XACML] defines an element called ActionAttributeDesignator that identifies the type of action being specified within in an ActionMatch or Apply element.

The actions defined by the abstract Access Control Model map to the following AttributeId and AttributeValue in the ActionMatch definitions:

Registry Action	ActionMatch.ActionAttributeDesignator.AttributeId	AttributeValue
Create	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`create`
Read	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`read`
Update	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`update`
Delete	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`delete`
Approve	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`approve`
Reference	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`reference`
Deprecate	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`deprecate`
Undeprecate	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`undeprecate`

Table 3: Action Binding to [XACML]

Action Attribute	ActionAttributeDesignator.AttributeId	DataType
id	`urn:oasis:names:tc:xacml:1.0:action:action-id`	`http://www.w3.org/2001/XMLSchema#anyURI`
reference-source	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source	http://www.w3.org/2001/XMLSchema#string
reference-source-attribute	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source-attribute	http://www.w3.org/2001/XMLSchema#string

9.7.3 Subject Binding

[XACML] defines an element called SubjectAttributeDesignator that identifies the type of subject attribute being specified in a SubjectMatch or Apply element.

The subjects defined by the abstract Access Control Model map to the following SubjectAttributeDesignator definitions:

Subject Attribute	SubjectAttributeDesignator	DataType
id	`urn:oasis:names:tc:xacml:1.0:subject:subject-id`	`http://www.w3.org/2001/XMLSchema#anyURI`
roles	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:roles	http://www.w3.org/2001/XMLSchema#string
groups	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:groups	http://www.w3.org/2001/XMLSchema#string
<attribute>	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:<attribute>	`As defined by attribute definition. Can be any attribute of the User instance for the subject.`

Table 4: Subject Binding to [XACML]

9.7.4 Function classification-node-compare

It is often necessary to test whether a resource matches a specific objectType or its sub-types. A client MAY use the special XACML function named *classification-node-compare* to perform such comparisons.

A registry MUST support a special XACML function named *classification-node-compare* whose canonical id is *urn:oasis:names:tc:ebxml-regrep:rim:acp:function:classification-node-compare*. A client MAY use this function within XACML Access control Policies to perform ClassificationNode comparisons in a taxonomy-aware manner. The following example shows how a ResourceMatch may be specified within an XACML Acccess Control Policy to perform such comparisons.

```
<!-- match ExtrinsicObject -->
```

```
<ResourceMatch
MatchId="urn:oasis:names:tc:ebxml-regrep:rim:acp:function:classification-
node-compare">
  <!—Specify the id for canonical ClassificationNode for ExtrinsicObject
objectType-->
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject
  </AttributeValue>

  <!—Specify the objectType of resource to compare with objectType
ExtrinsicObject -->
    <ResourceAttributeDesignator DataType =
"http://www.w3.org/2001/XMLSchema#string"
  AttributeId = "urn:oasis:names:tc:ebxml-
regrep:rim:acp:resource:objectType"/>
</ResourceMatch>
```

9.7.5 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.

9.7.6 Example: Default Access Control Policy

The following Policy defines the default access control policy. This Policy MUST implicitly apply to any resource that does not have an explicit Access Control Policy defined.
It consists of 3 rules, which in plain English are described as follows:

- Any subject can perform read action on any resource
- A subject may perform any action on a resource for which they are the owner.
- A subject with role of RegistryAdministrator may perform any action on any resource.

The non-normative listing of the default Access Control Policy follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides"
PolicySetId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:policy:default-
access-control-policy" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-schema-
policy-01.xsd">
  <Description>This PolicySet defines the default Access Control Policy
for all registry resources.</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-read"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
    <Target>
```

```
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-anyone-to-read">
      <Description>Any Subject can perform read action on any
resource.</Description>
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
             <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-reference"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-anyone-to-reference">
      <Description>Any Subject can perform reference action on any
resource as long as it is not deprecated.</Description>
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">reference</AttributeVa
lue>
              <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
      <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
equal">
          <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
            <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:status"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </Apply>
          <!-- Compare with the id for deprecated status -->
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names:tc:ebx
ml-regrep:StatusType:Deprecated</AttributeValue>
        </Apply>
      </Condition>
    </Rule>
  </Policy>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-owner-all"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-owner-all">
      <Description>A Subject with role of ContenOwner can perform any
action on resources owned by them.</Description>
      <Target>
        <Subjects>
          <AnySubject/>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <AnyAction/>
        </Actions>
      </Target>
      <Condition
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
one-and-only">
          <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
one-and-only">
```

```
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:owner"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </Apply>
      </Condition>
    </Rule>
  </Policy>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-all"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-registryadministrator-all">
      <Description>A Subject with role of RegistryAdministrator can
perform any action on any resource.</Description>
      <Target>
        <Subjects>
          <Subject>
            <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/urn:oasis:names:tc:eb
xml-
regrep:classificationScheme:SubjectRole/RegistryAdministrator</AttributeV
alue>
              <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <AnyAction/>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
```

9.7.7 Example: Custom Access Control Policy

The following Policy defines a custom access control policy to restrict *read access* to a resource to specified user or role. It also grants update access to specified role.
It consists of 3 rules, which in plain English are described as follows:

1. A subject may perform any action on a resource for which they are the owner. This reuses a Policy by reference from the default Access Control PolicySet.
2. A subject with the role of RegistryAdministrator may perform any action on any resource. This reuses a Policy by reference from the default Access Control PolicySet.

3. A subject with specified id may perform read actions on the resource. This restricts read access to the specified subject.
4. A subject with role of Manager may perform update actions on the resource. This relaxes update access restrictions to the specified subject.

The listing of the custom Access Control Policy follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides"
PolicySetId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:restricted-access-control-policyset"
xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-schema-
policy-01.xsd">
  <Description>This PolicySet restricts the default Access Control Policy
to limit read access to specified subjects.</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicyIdReference>urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-owner-all</PolicyIdReference>
  <PolicyIdReference>urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-
all</PolicyIdReference>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:permit-delete-access-control-policy"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-
policy-01.xsd">
    <Description>Allow Subject with specifed id to perform delete action
on any resource.</Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-delete-rule">
      <Description>Allow Subject with specifed id to perform delete
action on any resource.</Description>
      <Target>
        <Subjects>
          <Subject>
            <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:freebxml:registry:
predefinedusers:farrukh</AttributeValue>
```

```
              <SubjectAttributeDesignator 
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" 
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch 
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue 
DataType="http://www.w3.org/2001/XMLSchema#string">delete</AttributeValue
>
              <ActionAttributeDesignator 
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" 
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:permit-update-access-control-policy" 
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy" 
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-
policy-01.xsd">
    <Description>Allow Subjects with ProjectLead role to perform update 
action on any resource.</Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-update-rule">
      <Description>Allow Subjects with ProjectLead role to perform read 
action on any resource.</Description>
      <Target>
        <Subjects>
          <Subject>
            <SubjectMatch 
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue 
DataType="http://www.w3.org/2001/XMLSchema#string">/urn:oasis:names:tc:eb
xml-
regrep:classificationScheme:SubjectRole/ProjectMember/ProjectLead</Attrib
uteValue>
              <SubjectAttributeDesignator 
AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles" 
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch 
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue 
DataType="http://www.w3.org/2001/XMLSchema#string">update</AttributeValue
>
              <ActionAttributeDesignator 
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" 
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
```

9.7.8 Example: Package Membership Access Control

The following Policy defines an access control policy for controlling who can add members to a RegistryPackge. It makes use of the Reference action.

It consists of 3 rules, which in plain English are described as follows:

1. Any subject can perform read action on any resource. Referenced from default access control policy.
2. A subject may perform any action on a resource for which they are the owner. Referenced from default access control policy.
3. A subject with role of RegistryAdministrator may perform any action on any resource. Referenced from default access control policy
4. A subjects with role ProjectLead may perform addmember action on any resource associated with this ACP.

The following is a non-normative example listing of this custom Access Control Policy:

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides" 
PolicySetId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:folderACP1" 
xmlns="urn:oasis:names:tc:xacml:1.0:policy" 
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-
policy-01.xsd">
  <Description>This PolicySet restricts adding members to RegistryPackage 
resource to Role ProjectLead</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicyIdReference>urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-
read</PolicyIdReference>
```

```
  <PolicyIdReference>urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-owner-all</PolicyIdReference>
  <PolicyIdReference>urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-
all</PolicyIdReference>
  <Policy PolicyId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:policy:permit-projectLead-addMember"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-
policy-01.xsd">
    <Description>Allow Subjects with ProjectLead role to add members to
any resource associated with this ACP.</Description>
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:rule:ruleid:permit-projectLead-addMember-rule">
      <Description>Allow Subjects with ProjectLead role to add members to
any resource.</Description>
      <Target>
        <Subjects>
          <Subject>
            <!-- Match role ProjectLead -->
            <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/urn:oasis:names:tc:eb
xml-
regrep:classificationScheme:SubjectRole/ProjectMember/ProjectLead</Attrib
uteValue>
              <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <AnyResource/>
        </Resources>
        <Actions>
          <Action>
            <!-- Match "reference" action -->
            <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">reference</AttributeVa
lue>
              <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
      <!--
           Match condition where all the following are true:
```

```
           1. reference is being made via the attribute sourceObject
(from an Association instance)
           2. The associationType attribute of the Association matches
the id for associationType HasMameber

           Above is equivalent to saying Match any HasMember associations
where the resource
           (the RegistryPackage) is the sourceObject.
      -->
      <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">SourceObject</Attribut
eValue>
          <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:action:reference-source-attribute"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </Apply>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names:tc:ebx
ml-regrep:AssociationType:HasMember</AttributeValue>
          <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:action:reference-source-attribute-
filter:associationType"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </Apply>
        </Apply>
      </Condition>
    </Rule>
  </Policy>
</PolicySet>
```

9.7.9 Resolving Policy References

An XACML PolicySet MAY reference XACML Policy objects defined outside the repository item containing the XACML PolicySet. A registry implementation MUST be able to resolve such references. To resolve such references efficiently a registry 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 registry to resolve policy references efficiently.

A registry 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 registry 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.

9.7.10 ebXML Registry as a XACML Policy Store

So far we have defined how ebXML registries MAY use [XACML] to define Access Control Policies to control access to RegistryObject and RepositoryItem resources.

An important side effect of the normative binding of the Access Control Model to [XACML] is that enterprises MAY also use ebXML Registry as a [XACML] Policy store to manage Policies for protecting resources outside the registry.

In this use case, enterprises may submit [XACML] Policies and PolicySets as ExtrinsicObject-RepositoryItem pairs. These Policies may be accessed or referenced by their URL as defined by the HTTP binding of the ebXML Registry Services interface in [ebRS].

9.8 Access Control Model: Custom Binding

A registry MAY support bindings to policies describes in formats other than [XACML]. The use of such policies sacrifices interoperability and is therefore discouraged. In such cases the RepositoryItem for the policy information MAY be in any format supported by the registry in an implementation specific manner.

10 References

10.1 Normative References

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt.
[ebRS]	ebXML Registry Services Specification http://www.oasis-open.org/committees/regrep/documents/3.0/specs/regrep-rs-3.0-cd-01.pdf
[UUID]	DCE 128 bit Universal Unique Identifier http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
[RFC 3066]	H. Alvestrand, ed. *RFC 3066: Tags for the Identification of Languages* 1995. http://www.ietf.org/rfc/rfc3066.txt
[XPATH]	XML Path Language (XPath) Version 1.0 http://www.w3.org/TR/xpath
[XACML]	OASIS eXtensible Access Control Markup Language (XACML) Version 1.0 http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-01.pdf
[NCName]	Namespaces in XML 19990114 http://www.w3.org/TR/REC-xml-names/#NT-NCName

10.2 Informative References

[ISO]	ISO 11179 Information Model http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument
[UML]	Unified Modeling Language http://www.uml.org http://www.omg.org/cgi-bin/doc?formal/03-03-01

A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS ebXML Registry Technical Committee, whose voting members at the time of publication are listed as contributors on the title page of this document.

Finally, the editors wish to acknowledge the following people for their contributions of material used as input to the OASIS ebXML Registry specifications:

Name	Affiliation
Aziz Abouelfoutouh	Government of Canada
Ed Buchinski	Government of Canada
Asuman Dogac	Middle East Technical University, Ankara Turkey
Michael Kass	NIST
Richard Lessard	Government of Canada
Evan Wallace	NIST
David Webber	Individual

B. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

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