OASIS/ebXML Registry Information Model v1.10 DRAFT

OASIS/ebXML Registry ~~Project Team~~Technical Committee

~~Working Draft 3/19/2001~~30 August 2001~~27 June 2001~~8 ~~May 2001~~

~~This version: Version 0.61~~

1 Status of this Document

~~This document specifies an ebXML~~ DRAFT ~~DRAFT STANDARD for the *eBusiness* community.~~

Distribution of this document is unlimited.

~~The document formatting is based on the Internet Society's Standard RFC format.~~

This version:

~~http://www.ebxml.org/specdrafts~~project_teams/registry/private~~/RegistryInfoModel~~v1.0~~0.61.pdf~~
http://www.oasis-open.org/committees/regrep/documents/rimV1-10.pdf~~doc~~ebxml.org/specs/ebRIM.pdf~~

***Latest version*:**
http://www.oasis-open.org/committees/regrep/documents/rimV1-10.pdf~~doc~~//www.ebxml.org/specs/ebRIM.pdf

http://www.ebxml.org.org//pspecdraftsroject_teams/registry/private/RegistryInfoModelv1.0.pdf

Previous version:

http://www.ebxml.org/project_teams/registry/privatee/RegistryInfoModelv0.90600.pdf

2 OASIS/ebXML Registry Technical Committeeparticipants

We would like to recognize the following for their significant participation to the development of this document. The authors wish to acknowledge the support of the members of the Registry Project Team who contributed ideas to this specification by the group's discussion e-mail list, on conference calls and during face-to-face meetings. This document, in its current form, is a draft working document of the OASIS ebXML Registry Technical Committee. It build upon version 1.0 which has beenwas approved by the OASIS/ebXML Registry Technical Committee as DRAFT Specfication of the TC. At the time of thatis approval the following were members of the OASIS/ebXML Registry Technical Committee.

Nagwa Abdelghfour, Sun Microsystems
Nicholas Berry, Boeing
Kathryn Breininger, Boeing
Lisa Carnahan, US NIST (TC Chair)
Dan Chang, IBM
Joseph M. Chiusano, LMI
Joe Dalman, Tie Commerce
Suresh Damodaran, Sterling Commerce
Vadim Draluk, BEA
John Evdemon, Vitria Technologies
Anne Fischer, Drummond Group
Sally Fuger, AIAG
Len Gallagher, NIST
Michael Joya, XMLGlobal
Una Kearns, Documentum
Kyu-Chul Lee, Chungnam National University
Megan MacMillan, Gartner Solista
Norbert Mikula, DataChannel
Joel Munter, Intel
Farrukh Najmi, Sun Microsystems
Joel Neu, Vitria Technologies
Sanjay Patil, IONA
Neal Smith, Chevron
Nikola Stojanovic, Encoda Systems Inc.
David Webber, XMLGlobal
Prasad Yendluri, webmethods
Yutaka Yoshida, Sun Microsystems
Lisa Carnahan, – NIST
Joe Dalman, - Tie

Philippe DeSmedt, - Viquity
Sally Fuger, – AIAG
Len Gallagher, - NIST
Steve Hanna, - Sun Microsystems
Scott Hinkelman, - IBM
Michael Kass, NIST
Jong.L Kim, – Innodigital
Kyu-Chul Lee, Chungnam National University
Sangwon Lim, Korea Institute for Electronic Commerce
Bob Miller, - GXS
Kunio Mizoguchi, - Electronic Commerce Promotion Council of Japan
Dale Moberg, – Sterling Commerce
Ron Monzillo, – Sun Microsystems
JP Morgenthal, – eThink Systems, Inc.
Joel Munter, - Intel
Farrukh Najmi, - Sun Microsystems
Scott Nieman, - Norstan Consulting
Frank Olken, – Lawrence Berkeley National Laboratory
Michael Park, - eSum Technologies
Bruce Peat, - eProcess Solutions
Mike Rowley, – Excelon Corporation
Waqar Sadiq, - Vitria
Krishna Sankar, - Cisco Systems Inc.
Kim Tae Soo, - Government of Korea
Nikola Stojanovic, - Encoda Systems, Inc.
David Webber, – XML Global
Yutaka Yoshida, - Sun Microsystems
Prasad Yendluri, - webmethods
Peter Z. Zhoo, - Knowledge For the new Millennium

Table of Contents

Table of Figures

Table of Tables

Introduction

3.12.1 Summary of Contents of Document

This document specifies the information model for the ebXML *Registry*.

A separate document, ebXML Registry Services Specification [ebRS], describes how to build *Registry Services* that provide access to the information content in the ebXML *Registry*.

3.22.2 General Conventions

- o*UML* diagrams are used as a way to concisely describe concepts. They are not intended to convey any specific *iImplementation* or methodology requirements.

oInterfaces are often used in *UML* diagrams. They are used instead of *Cc*lasses with attributes to provide an abstract definition without implying any specific *iImplementation*. Specifically, they do not imply that objects in the *Registry* will be accessed directly via these interfaces. Objects in the *Registry* are accessed via interfaces described in the ebXML Registry Services Specification. . Each get method in every interface has an explicit indication of the attribute name that the get method maps to. For example getName method maps to an attribute named name .

- o *UML* diagrams are used as a way to concisely describe concepts. They are not intended to convey any specific *Implementation* or methodology requirements.

The term *"repository item"* is used to refer to an object that has been submitted to a Registry for storage and safekeeping (e.g. an XML document or a DTD). Every repository item is described by a RegistryEntryRegistryObject instanceThe term "repository item" is used to refer to actual *Registry* content (e.g. a *DTD*, as opposed to metadata about the *DTD*). It is important to note that the information model is not modeling actual repository items.

- o

The term "RegistryEntryRegistryObject" is used to refer to an object that provides metadata about a content *instance* (repository item).

- o The term "RegistryObject" is also used to refer to the name of the most base class in the information model.

interface in the information model to avoid the confusion with the common term "object".

However, when the term "object" is used to refer to a *class* or an interface in the information model, it may also mean RegistryObject because almost all classes are descendants of RegistryObject.

- The information model does not containdeal with any *elements* that are the actual content of the Registry (repository item)repository. All *eElements* of the information model represent metadata about the content and not the content itself.

Software practitioners MAY use this document in combination with other ebXML specification documents when creating ebXML compliant software.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

2.2.1 Naming Conventions

In order to enforce a consistent capitalization and naming convention in this document, "Upper Camel Case" *(UCC)* and "Lower Camel Case" *(LCC)* Capitalization styles are used in the following conventions

- Element name is in *UCC* convention
- (example: <UpperCamelCaseElement/>).
- Attribute name is in *LCC* convention
- (example: <UpperCamelCaseElement lowerCamelCaseAttribute="Whatever"/>).
- *Class*, Interface names use UCC convention
- (examples: ClassificationNode, Versionable).
- Method name uses LCC convention
- (example: getName(), setName())

Also, *Capitalized Italics* words are defined in the ebXML Glossary [ebGLOSS].

3.32.3 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*

3.42.4 Related Documents

The following specifications provide some background and related information to the reader:

a)*ebXML Registry Business Domain Model* [BDM] - defines requirements for ebXML Registry Services

b)a) ebXML Registry Services Specification [ebRS] - defines the actual *Registry Ssservices* based on this information model

c)b) ebXML Collaboration-Protocol Profile and Agreement Specification [ebCPPA] (under development) - defines how profiles can be defined for a *pParty* and how two *pParties'* profiles maymay be used to define a *pParty* agreement

d)c) ebXML Business Process Specification Schema [BPMebBPSS]

d) ebXML Technical Architecture Specification [ebTA]

43 Design Objectives

4.13.1 Goals

The goals of this version of the specification are to:

- Communicate what information is in the *Registry* and how that information is organized
- Leverage as much as possible the work done in the *OASIS* [OAS] and the *ISO* 11179 [ISO] Registry models
- Align with relevant works in progress within other ebXML working groups
- Be able to evolve to support future ebXML *Registry* requirements
- Be compatible with other ebXML specifications

4.2Caveats and Assumptions

The Registry Information Model specification is first in a series of phased deliverables. Later versions of the document will include additional functionality planned for current and future development.

54 System Overview

5.14.1 Role of ebXML *Registry*

The *Registry* provides a stable store where content information submitted by a *Submitting Organization* is made persistent. Such content information is used to facilitate ebXML-based *Bbusiness* to *Bbusiness* (B2B) partnerships and transactions. Submitted content may be *XML* schema and documents, process descriptions, *Ccore Ccomponents*, context descriptions, *UML* models, information about parties and even software components.

5.24.2 Registry Services

A set of *Registry Services* that provide access to *Registry* content to clients of the *Registry* is defined in the ebXML Registry Services Specification [ebRS]. This document does not provide details on these services but may occasionally refer to them.

5.34.3 What the Registry Information Model Does

The Registry Information Model provides a blueprint or high-level schema for the ebXML *Registry*. Its primary value is for implementers of ebXML *Registries*. It provides these implementers with information on the type of metadata that is stored in the *Registry* as well as the relationships among metadata *Cclasses*.

The Registry information model:

- Defines what types of objects are stored in the *Registry*
- Defines how stored objects are organized in the *Registry*
- Is based on ebXML metamodels from various working groups

5.44.4 How the Registry Information Model Works

Implementers of the ebXML *Registry* maymay use the information model to determine which *Cclasses* to include in their *Registry* *iImplementation* and what attributes and methods these *Cclasses* maymay have. They maymay also use it to determine what sort of database schema their *Registry* *iImplementation* maymay need.

```
[Note]The information model is meant to be
     illustrative and does not prescribe any
     specific iImplementation choices.
```

5.54.5 Where the Registry Information Model May Be Implemented

The Registry Information Model maymay be implemented within an ebXML *Registry* in the form of a relational database schema, object database schema or some other physical schema. It maymay also be implemented as interfaces and *Cclasses* within a *Registry* *iImplementation*.

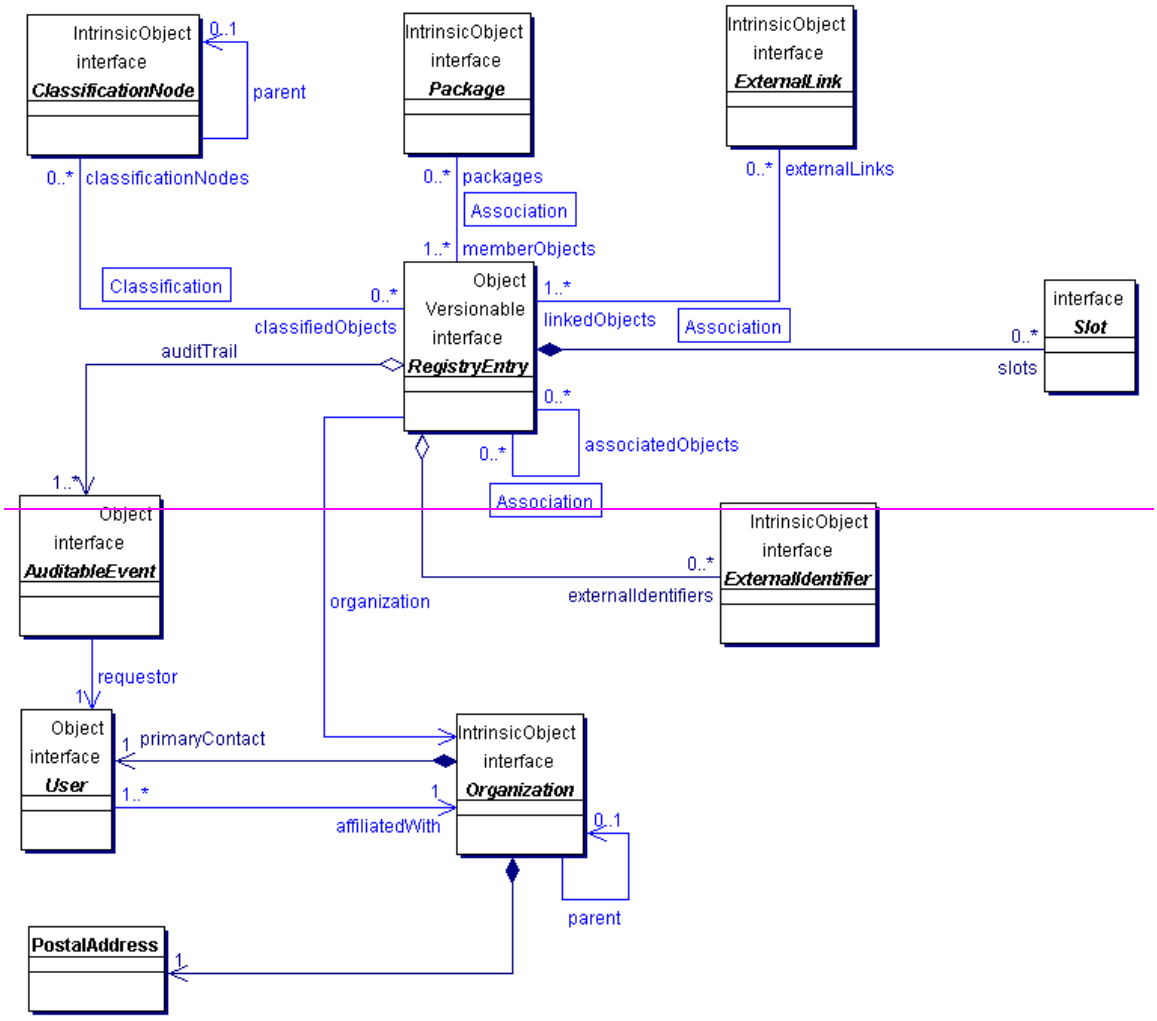
5.64.6 Conformance toas an ebXML *Registry*

If an *iImplementation* claims *cConformance* to this specification then it supports all required information model *Cclasses* and interfaces, their attributes and their semantic definitions that are visible through the ebXML *Registry Services*.

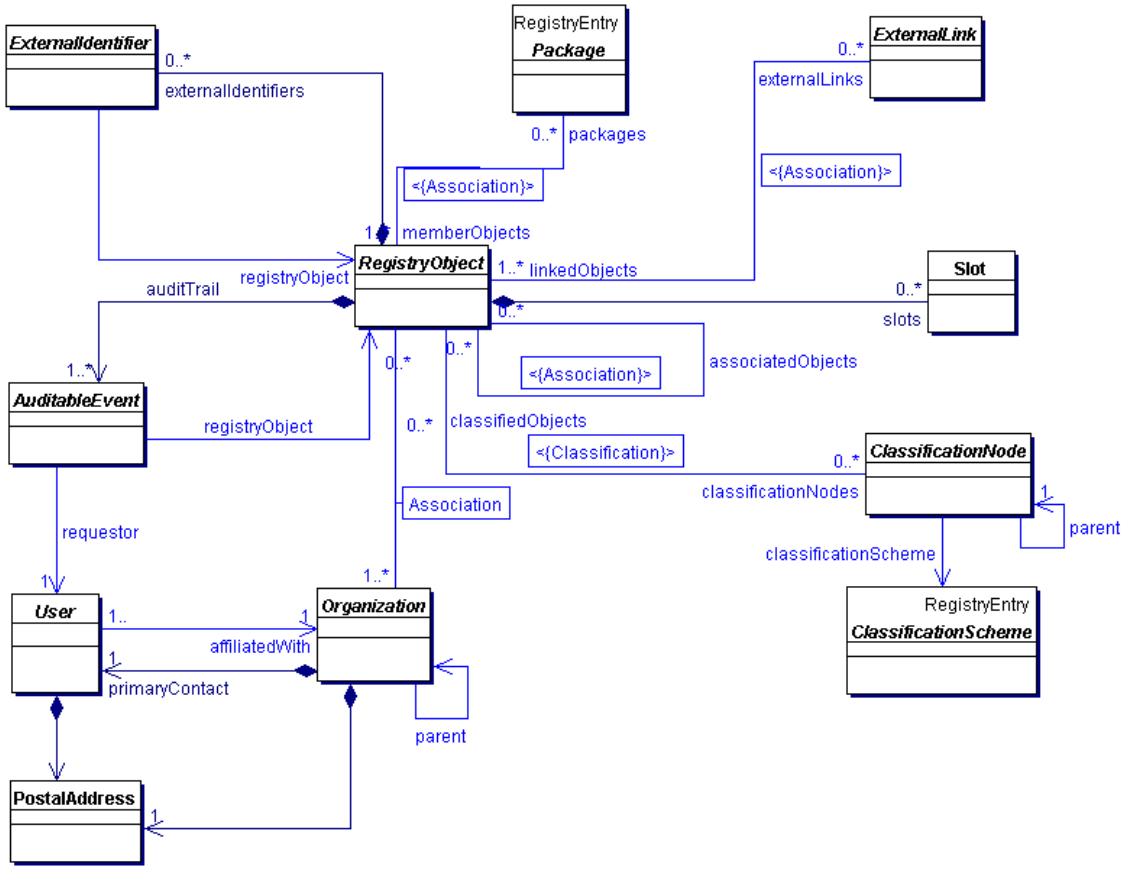
65 Registry Information Model: High LevelPublic Public View

This chaptersection provides a high level public view of the most visible objects in the *Registry*.

Figure 1Figure 1Figure 1Figure 1Figure 1 shows the high level public view of the objects in the *Registry* and their relationships as a *UML Cclass Ddiagram*. It does not show *iInheritance*, *Cclass* attributes or *Cclass* methods.
The reader is again reminded that the information model is not modeling actual repository items.



601



602

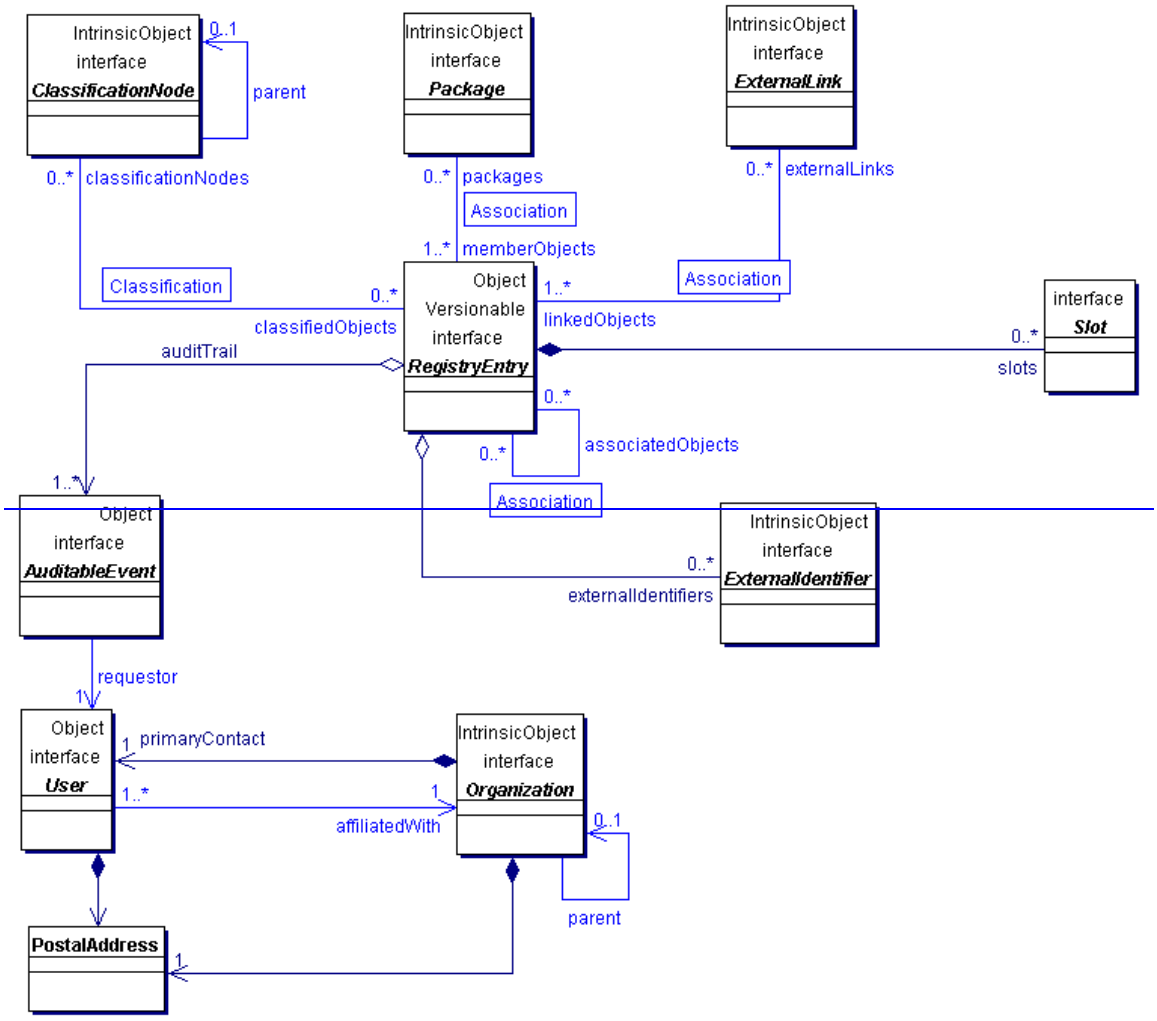


Figure 11: Information Model High Level Public View

6.15.1 RegistryObjectEntry

The RegistryObject class is an abstract base class used by most classes in the model. It provides minimal metadata for registry objects. It also provides methods for accessing related objects that provide additional dynamic metadata for the registry object.The central object in the information model is a RegistryEntry. An *iInstance* of RegistryEntry exists for each content *iInstance* submitted to the *Registry*. *Instances* of the RegistryEntry *Cclass* provide metadata about a repository item in the *Registry*. The actual repository item (e.g. a *DTD*) is not contained in an *iInstance* of the RegistryEntry *Cclass*. Note that most *Cclasses* in the information model are specialized sub-classes of RegistryEntry. Each RegistryEntry is related to exactly one repository item, however, in the future revision of this document, it may be related to multiple repository items.

6.25.2 Slot

Slot iInstancesinstances provide a dynamic way to add arbitrary attributes to RegistryEntryRegistryObject iInstancesinstances. This ability to add attributes dynamically to RegistryEntryRegistryObject iInstancesinstances enables extensibility within the Registry Information Model. For example, if a company wants to add a "copyright" attribute to each RegistryObject instance that it submits, it can do so by adding a slot with name "copyright" and value containing the copyrights statement.

6.35.3 Association

Association iInstancesinstances are RegistryEntriesRegistryObject instances that are used to define many-to-many associations between objects in the information model. Associations are described in detail in chaptersection 910.

6.45.4 ExternalIdentifier

ExternalIdentifier iInstancesinstances provide additional identifier information to a RegistryEntryRegistryObject instance, such as DUNS number, Social Security Number, or an alias name of the organization.

6.55.5 ExternalLink

ExternalLink iInstancesinstances are RegistryEntriesRegistryObject instances that model a named URI to content that is not managed by the *Registry*. Unlike managed content, such external content may change or be deleted at any time without the knowledge of the rRegistry. A RegistryEntryRegistryObject instance may be associated with any number of ExternalLinks.

Consider the case where a *Submitting Organization* submits a repository item (e.g. a *DTD*) and wants to associate some external content to that object (e.g. the *Submitting Organization*'s home page). The ExternalLink enables this capability. A potential use of the ExternalLink capability may be in a GUI tool that displays the ExternalLinks to a RegistryEntryRegistryObject. The user may click on such links and navigate to an external web page referenced by the link.

5.6 ClassificationScheme

A ClassificationScheme instance is a RegistryObject instance that represents a structured way to classify or categorize RegistryObject instances. A very common example of a classification scheme in science is the *Classification of living things* where living things are categorized in under a tree like structure. Another example is the Dewey Decimal system used in libraries to categorize books and other publications. ClassificationScheme is described in detail in section 10.

6.65.7 ClassificationNode

ClassificationNode iInstancesinstances are RegistryEntriesRegistryObject instances that are used to define tree structures under a ClassificationScheme, where each node in the tree is a ClassificationNode and the root is the ClassificationScheme. *Classification* trees constructed with ClassificationNodes are used to define cClassification schemes or ontologies. ClassificationNode is described in detail in chaptersection 1011.

6.75.8 ClClassification

Classification iiInstances are RegistryObjectEntries that are used to classify repository items by associating their RegistryEntryother RegistryObject iiInstances with a ClassificationNode within a cClassificationS scheme. Classification is described in detail in chaptersection 1011.

6.85.9 Package

Package iiInstances are RegistryEntry instancesies that group logically related RegistryEntriesRegistryObject instances together. One use of a Package is to allow operations to be performed on an entire pPackage of objects. For example all objects belonging to a Package may be deleted in a single request.

6.95.10 AuditableEvent

AuditableEvent iInstancesinstances are ObjectsRegistryObject instances that are used to provide an audit trail for RegistryEntriesRegistryObject instances. AuditableEvent is described in detail in chaptersection 78.

6.105.11 User

User iInstancesinstances are ObjectsRegistryObject instances that are used to provide information about registered users within the rRegistry. User objects are used in audit trail for RegistryEntriesRegistryObject instances. User is described in detail in chaptersection 78.

6.115.12 PostalAddress

PostalAddress is a simple reusable eEntity Cclass that defines attributes of a postal address.

6.125.13 Organization

Organization iInstancesinstances are RegistryEntriesRegistryObject instances that provide information on organizations such as a *Submitting Organization*. Each Organization iiInstance may have a reference to a parent Organization.

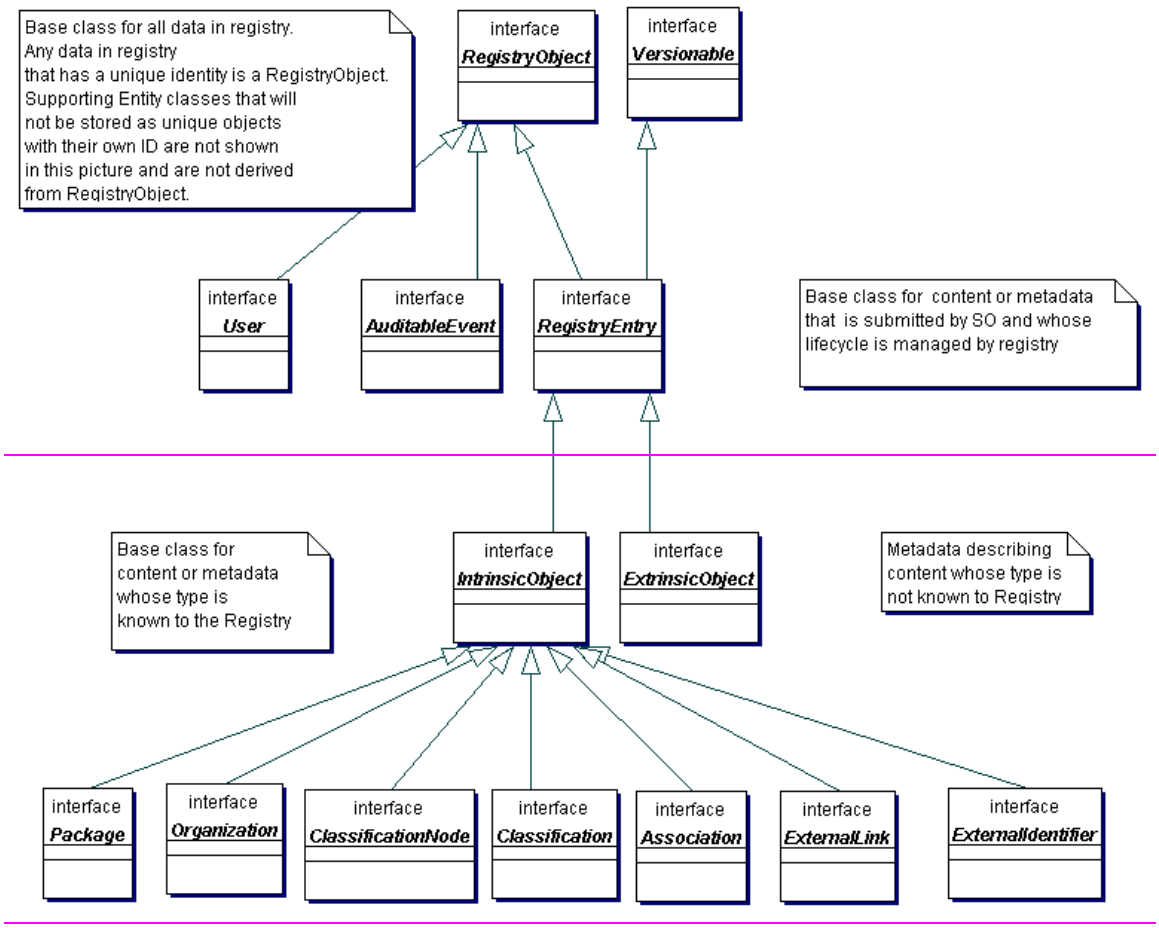
76 Registry Information Model: Detail View

This chaptersection covers the information model *Cclasses* in more detail than the Public View. The detail view introduces some additional *Cclasses* within the model that were not described in the public view of the information model.

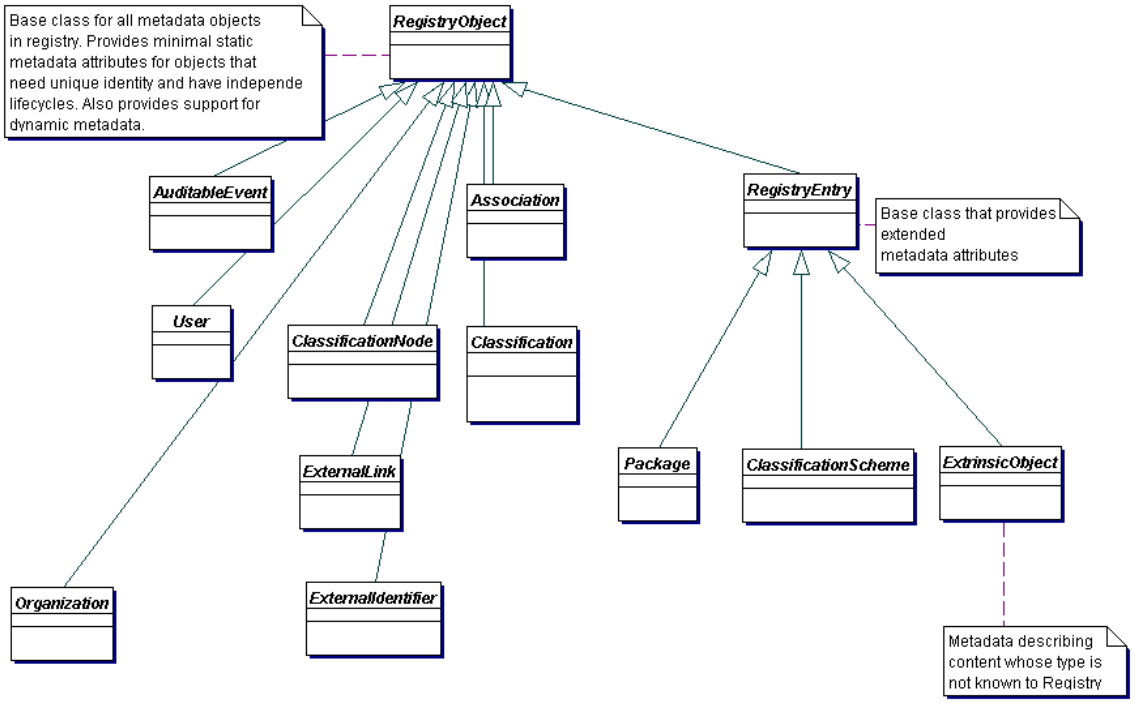
Figure 2Figure 2Figure 2Figure 2Figure 2 shows the *iInheritance* or "is a" relationships between the *Cclasses* in the information model. Note that it does not show the other types of relationships, such as "has a" relationships, since they have already been shown in a previous figure. *Class* attributes and *class* methods are also not shown. Detailed description of methods and attributes of most interfaces and *Cclasses* will be displayed in tabular form following the description of each *Cclass* in the model.

The interfaceclass Association will be covered in detail separately in chaptersection 910. The interfacesclasses Classification and ClassificationNode will be covered in detail separately in chaptersection 1011.

The reader is again reminded that the information model is not modeling actual repository items.



707



708

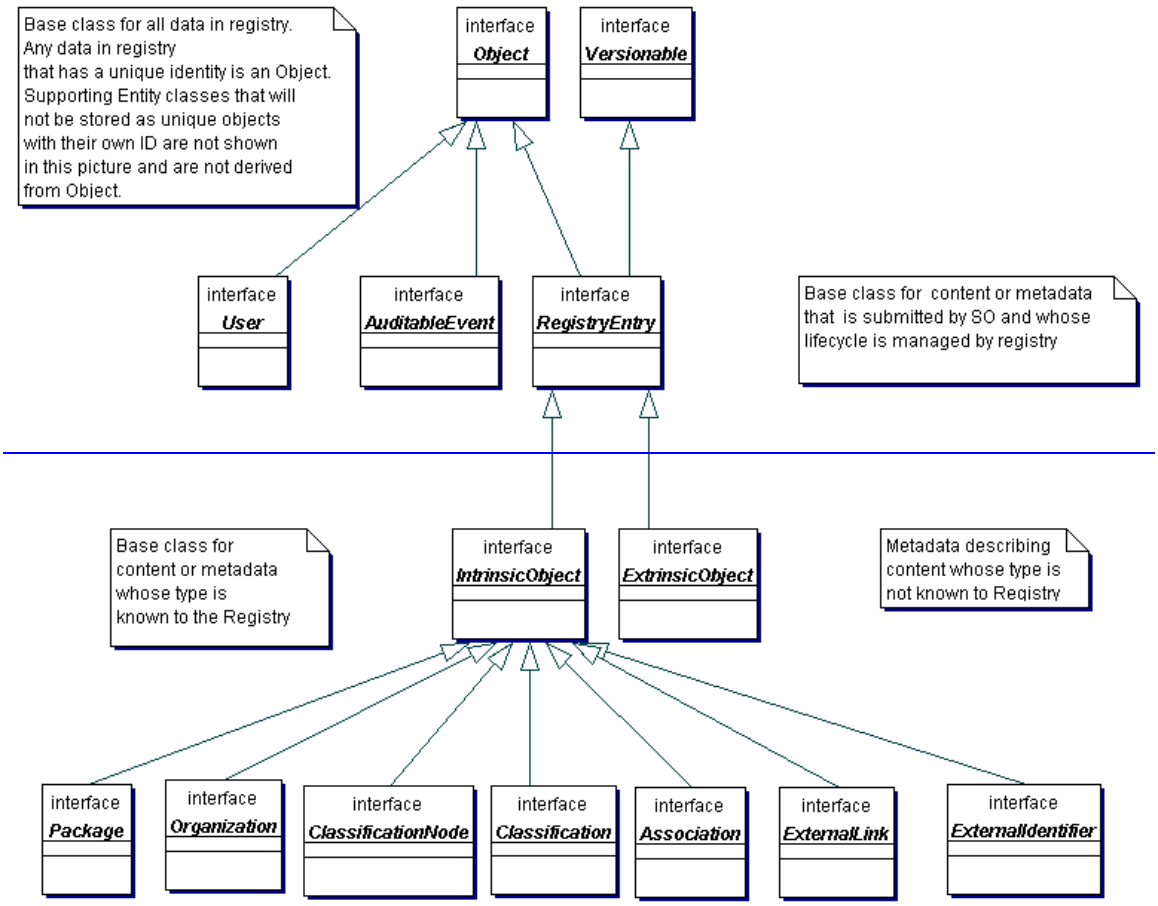


Figure 222: Information Model *Inheritance* View

6.1 Attribute and Methods of Information Model Classes

Information model classes are defined primarily in terms of the attributes they carry. These attributes provide state information on instances of these classes. Implementations of a registry often map class attributes to attributes in an XML store or columns in a relational store.

Information model classes may also have methods defined for them. These methods provide additional behavior for the class they are defined within. Methods are currently used in mapping to SQL stored procedures in the SQL query capability defined in [ebRS].

Since the model supports inheritance between classes, it is usually the case that a class in the model inherits attributes and methods from its base classes, in addition to defining its own specialized attributes and methods.

6.2 Data Types

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

Data Type	Primitive Type	Description	Length
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
ShortName	String	A short text string	64 characters
LongName	String	A long text string	128 characters
FreeFormText	String	A very long text string for free-form text	256 characters
UUID	String	DCE 128 Bit Universally unique Ids used for referencing another object	64 characters
URI	String	Used for URL and URN values	256 characters
Integer		Used for integer values	4 bytes
Timestamp		Used for a time stamp value such as Date	

7.16.3 InterfaceClass ObjectRegistryObject

All Known Subinterfaces:

Direct Known Subclasses:

Association, AuditableEvent, Classification, ClassificationNode, ExternalIdentifier, ExternalLink, Organization, RegistryEntry, UserAssociation, Classification, ClassificationNode, ExternalLink, ExtrinsicObject, IntrinsicObject, RegistryEntry, Organization, Package, User, AuditableEvent, ExternalIdentifier, Submission

---

ObjectRegistryObject provides a common base interfaceclass for almost all objects in the information model. Information model *cClasses* whose *iInstances*instances have a unique identity and an independent life cycle are descendants of the ObjectRegistryObject *cClass*.

Note that Slot and PostalAddress are not descendants of the ObjectRegistryObject *cClass* because their *iInstances*instances do not have an independent existence and unique identity. They are always a part of some other *cClass*'s *iInstance* (e.g. Organization has a PostalAddress).

6.3.1 Attribute Summary

The following is the first of many tables that summarize the attributes of a class. The columns in the table are described as follows:

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	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

Attribute	Data Type	Required	Default Value	Specified By	Mutable	
accessControlPolicy	UUID	No		Registry	No	
description	FreeFormText	No		Client	Yes	
id	UUID	Yes		Client or registry	No	
name	LongName	No		Client	Yes	
objectType	LongName	Yes		Registry	No	

6.3.2 Attribute accesControlPolicy

Each RegistryObject instance has an AccessControlPolicy instance associated with it. An AccessControlPolicy instance defines the *Security Model* associated with the RegistryObject in terms of “who is permitted to do what” with that RegistryObject.

6.3.3 Attribute description

Each RegistryObject instance may have textual description in a human readable and user-friendly manner.

6.3.4 Attribute id

Each RegistryObject instance must have a universally unique ID. Registry objects use the id of other RegistryObject instances for the purpose of referencing those objects.

Note that some classes in the information model do not have a need for a unique id. Such classes do not inherit from RegistryObject class. Examples include Entity classes such as TelephoneNumber, PostalAddress and PersonName.

The id attribute of various derived classes of RegistryObject fall into two categories:

1. UUID based Id
2. Attribute based Id

6.3.4.1 UUID based Id

Most classes derived from RegistryObject have an id that is a Universally Unique ID as defined by [UUID]. Such UUID based Id attributes may be specified by the client. If the UUID based ID is not specified, it must be generated by the registry when a new RegistryObject instance is first submitted to the registry.

6.3.4.2 Attribute based Id

A few classes derived from RegistryObject have an Id that is not a UUID but is instead composed of multiple attributes of that object. This is very similar to the concept of a multi-column primary key in relational databases, or multi-attribute key instances in XML Schema.

Examples of classes that use attribute based Id are Classification, Association and ExternalIdentifier. The reason these objects do not use UUIDs and instead use attribute based Id is that they do not have an independent lifecycle separate from their primary RegistryObject.

Attribute based Ids are not UUIDs and therefore are not constrained by the 64 bit limit of the UUID data type. Instead they can be of arbitrary length.

6.3.5 Attribute name

Each RegistryObject instance may have human readable name. The name does not need to be unique with respect other RegistryObject instances.

6.3.6 Attribute objectType

Each RegistryObject instance has an objectType. The objectType for almost all objects in the information model is the name of their class. For example the objectType for a Classification is “Classification”. The only exception to this rule is that the objectType for an ExtrinsicObject instance is user defined and indicates the type of repository item associated with the ExtrinsicObject.

6.3.6.1 Pre-defined Object Types

The following table lists pre-defined object types. Note that for an ExtrinsicObject there are many types defined based on the type of repository item the ExtrinsicObject catalogs. In addition there are object types defined for IntrinsicObject sub-classes that may have concrete instances.

These pre-defined object types are defined as a *ClassificationScheme*. While the scheme may easily be extended a *Registry* MUST support the object types listed below.

Name	description
Unknown	An ExtrinsicObject that catalogues content whose type is unspecified or unknown.
CPA	An ExtrinsicObject of this type catalogues an *XML* document *Collaboration Protocol Agreement* (*CPA*) representing a technical agreement between two parties on how they plan to communicate with each other using a specific protocol.
CPP	An ExtrinsicObject of this type catalogues an document called *Collaboration Protocol Profile* (*CPP*) that provides information about a *Party* participating in a *Business* transaction.
Process	An ExtrinsicObject of this type catalogues a process description document.
Role	An ExtrinsicObject of this type catalogues an *XML* description of a *Role* in a *Collaboration Protocol Profile* (*CPP*).
ServiceInterface	An ExtrinsicObject of this type catalogues an *XML* description of a service interface as defined by [ebCPP].
SoftwareComponent	An ExtrinsicObject of this type catalogues a software component (e.g., an EJB or *Class* library).
Transport	An ExtrinsicObject of this type catalogues an *XML* description of a transport configuration as defined by [ebCPP].
UMLModel	An ExtrinsicObject of this type catalogues a *UML* model.
XMLSchema	An ExtrinsicObject of this type catalogues an *XML* schema (*DTD*, *XML* Schema, RELAX grammar, etc.).
Package	A Package object
ExternalLink	An ExternalLink object
ExternalIdentifier	An ExternalIdentifier object
Association	An Association object
Classification	A Classification object
ClassificationNode	A ClassificationNode object
AuditableEvent	An AuditableEvent object
User	A User object

Organization	An Organization object

814

815 ### 6.3.7 Method Summary

816 In addition to its attributes, the RegistryObject class also defines the following
817 methods. These methods are used to navigate relationship links from a
818 RegistryObject instance to other objects.
819

Method Summary for RegistryObject	
Collection[1]	**getAssociatedObjects**( ) Gets the collection of RegistryObject instances associated with this object.
Collection	**getAssociations**( ) Gets all Associations where this object is the source of the Association.
Collection	**getAuditTrail**( ) Gets the complete audit trail of all requests that effected a state change in this object as an ordered Collection of AuditableEvent objects.
Collection	**getClassificationNodes**( ) Gets the ClassificationNodes that classify this object.
Collection	**getClassifications**( ) Gets the Classification that classify this object.
Collection	**getExternalIdentifiers**( ) Gets the collection of ExternalIdentifiers associated with this object.
Collection	**getExternalLinks**( ) Gets the ExternalLinks associated with this object.
Collection	**getOrganizations**(String type) Gets the Organizations associated with this object. If a non-null type is specified it is used as a filter to match only specified type of organizations as indicated by the associationType attribute in the Association instance linking the object to the Organization.
Collection	**getPackages**( ) Gets the Packages that this object is a member of.
Collection	**getSlots**( ) Gets the Slots associated with this object.

[1] A Collection represents a collection of multiple RegistryObject instances

820
821
822
823

7.2Interface Versionable

All Known Subinterfaces:

Association, Classification, ClassificationNode, ExternalLink, ExtrinsicObject, IntrinsicObject, RegistryEntry, Organization, Package, ExternalIdentifier

The Versionable interface defines the behavior common to *cClasses* that are capable of creating versions of their *i*Instances. At present all RegistryEntry *cClasses* are requiredREQUIRED to implement the Versionable interface.

Method Summary of Versionable	
int	**getMajorVersion**() Gets the major revision number for this version of the Versionable object. Maps to attribute named `majorVersion`.
int	**getMinorVersion**() Gets the minor revision number for this version of the Versionable object. Maps to attribute named `minorVersion`.
void	**setMajorVersion**(int majorVersion) SetGets the major revision number for this version of the Versionable object.
void	**setMinorVersion**(int minorVersion) Sets the minor revision number for this version of the Versionable object.

7.36.4 InterfaceClass RegistryEntry

Super Classes:

RegistryObject

All Superinterfaces:

ObjectRegistryObject, Versionable

Direct Known Subclasses:

ClassificationScheme, ExtrinsicObject, Package**All Known Subinterfaces:**

Association, Classification, ClassificationNode, ExternalLink, ExtrinsicObject, IntrinsicObject, Organization, Package, ExternalIdentifier

RegistryEntry is a common base *cClass* for classes in the information model that require additional metadata beyond the minimal metadata provided by

RegistryObject class. The additional metadata is described by the attributes of the RegistryEntry class below.

6.4.1 Attribute Summaryall metadata describing submitted content whose life cycle is managed by the rRegistry. Metadata describing content submitted to the rRegistry is further specialized by the ExtrinsicObject and IntrinsicObject subclasses of RegistryEntry.

Attribute	Data Type	Required	Default Value	Specified By	Mutable
expiration	Timestamp	No		Client	Yes
majorVersion	Integer	Yes	1	Registry	Yes
minorVersion	Integer	Yes	0	Registry	Yes
stability	LongName	No		Client	Yes
status[2]	LongName	Yes		Registry	Yes
userVersion	ShortName	No		Client	Yes

Note that attributes inherited by RegistryEntry class from the RegistryObject class are not shown in the table above.

6.4.2 Attribute expiration

Each RegistrEntry instance may have an expirationDate. This attribute defines a time limit upon the stability guarantee provided by the stability attribute. Once the expirationDate has been reached the stability attribute in effect becomes STABILITY_DYNAMIC implying that content can change at any time and in any manner. A null value implies that there is no expiration on stability attribute.

6.4.3 Attribute majorVersion

Each RegistrEntry instance must have a major revision number for the current version of the RegistryEntry instance. This number is assigned by the registry when the object is created. This number may be updated by the registry when an object is updated.

6.4.4 Attribute minorVersion

Each RegistrEntry instance must have a minor revision number for the current version of the RegistryEntry instance. This number is assigned by the registry

[2] Was Integer in RIM 1.0 for some reason.

when the object is created. This number may be updated by the registry when an object is updated.

6.4.5 Attribute stability

Each RegistrEntry instance may have a stability indicator. The stability indicator is provided by the submitter as a guarantee of the level of stability for the content.

6.4.5.1 Pre-defined RegistryEntry Stability Enumerations

The following table lists pre-defined choices for RegistryEntry stability attribute. These pre-defined stability types are defined as a *Classification* scheme. While the scheme may easily be extended, a *Registry* MAY support the stability types listed below.

Name	Description
Dynamic	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed arbitrarily by submitter at any time.
DynamicCompatible	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed in a backward compatible way by submitter at any time.
Static	Stability of a RegistryEntry that indicates that the content is static and will not be changed by submitter.

6.4.6 Attribute status

Each RegistryEntry instance must have a life cycle status indicator. The status is assigned by the registry.

6.4.6.1 Pre-defined RegistryObject Status Types

The following table lists pre-defined choices for RegistryObject status attribute. These pre-defined status types are defined as a *Classification* scheme. While the scheme may easily be extended, a *Registry* MUST support the status types listed below.

Name	Description
Submitted	Status of a RegistryObject that catalogues content that has been submitted to the *Registry*.
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.
Withdrawn	Status of a RegistryObject that catalogues content that has been withdrawn from the *Registry*.

6.4.7 Attribute userVersion

Each RegistrEntry instance may have a userVersion. The userVersion is similar to the majorVersion-minorVersion tuple. They both provide an indication of the version of the object. The majorVersion-minorVersion tuple is provided by the registry while userVersion provides a user specified version for the object.

7.3.1 Pre-defined RegistryEntry Status Types

The following table lists pre-defined choices for RegistryEntry status attribute. These pre-defined status types are defined as a *Classification* scheme. While the scheme may easily be extended, a *rRegistry* must MUST support the status types listed below.

Name	Description
Submitted	Status of a RegistryEntry that catalogues content that has been submitted to the *Registry*.
Approved	Status of a RegistryEntry that catalogues content that has been submitted to the *Registry* and has been subsequently approved.
Deprecated	Status of a RegistryEntry that catalogues content that has been submitted to the *Registry* and has been subsequently deprecated.
Withdrawn	Status of a RegistryEntry that catalogues content that has been withdrawn from the *Registry*.

7.3.2 Pre-dDefined Object Types

The following table lists pre-defined object types. Note that for an ExtrinsicObject there are many types defined based on the type of repository item the ExtrinsicObject catalogs. In addition there there are object types defined for IntrinsicObject sub-classes that may have concrete *i*Instances.

These pre-defined object types are defined as a *Classification* scheme. While the scheme may easily be extended a *rRegistry* mustMUST support the object types listed below.

name	description
Unknown	An ExtrinsicObject that catalogues content whose type is unspecified or unknown.
CPA	An ExtrinsicObject of this type catalogues an *XML* document *Collaboration Protocol Agreement* (*CPA*) representing a technical agreement between two parties on how they plan to communicate with each other using a specific protocol.
CPP	An ExtrinsicObject of this type catalogues an *XML* document called *Collaboration Protocol Profile* (*CPP*) that provides information about a *pParty* participating in a *Bbusiness* transaction.
Process	An ExtrinsicObject of this type catalogues a process description document.
Role	An ExtrinsicObject of this type catalogues an *XML* description of a *Role* in a *Collaboration Protocol Profile* (*CPP*).
ServiceInterface	An ExtrinsicObject of this type catalogues an *XML* description of a service interface as defined by [CPAebCPP].
SoftwareComponent	An ExtrinsicObject of this type catalogues a software component (e.g., an EJB or *cClass* library).
Transport	An ExtrinsicObject of this type catalogues an *XML* description of a transport configuration as defined by [CPAebCPP].
UMLModel	An ExtrinsicObject of this type catalogues a *UML* model.
XMLSchema	An ExtrinsicObject of this type catalogues an *XML* schema (*DTD*, *XML* Schema, RELAX grammar, etc.).
Package	A Package object
ExternalLink	An ExternalLink object
ExternalIdentifier	An ExternalIdentifier object
Association	An Association object
Classification	A Classification object
ClassificationNode	A ClassificationNode object
AuditableEvent	An AuditableEvent object
User	A User object
Organization	An Organization object

917 -

7.3.3 Pre-defined RegistryEntry Stability Enumerations

The following table lists pre-defined choices for RegistryEntry stability attribute. These pre-defined stability types are defined as a *Classification* scheme. While the scheme may easily be extended, a *rRegistry* mustMAY support the stability types listed below.

Name	Description
Dynamic	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed arbitrarily by submitter at any time.
DynamicCompatible	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed in a backward compatible way by submitter at any time.
Static	Stability of a RegistryEntry that indicates that the content is static and will not be changed by submitter.

7.46.5 InterfaceClass Slot

Slot *i*Instancesinstances provide a dynamic way to add arbitrary attributes to RegistryEntryRegistryObject *i*Instancesinstances. This ability to add attributes dynamically to RegistryEntryRegistryObject *i*Instancesinstances enables extensibility within the iRegistry Information mModel.

In this model, aA RegistryEntryRegistryObject may have 0 or more Slots. A slot is composed of a name, a slotType and a collection of values.

6.5.1 Attribute SummaryThe name of slot is locally unique within the RegistryEntry *iInstance*. Similarly, the value of a Slot is locally unique within a slot *iInstance*. Since a Slot represent an extensible attribute whose value may be a collection, therefore a Slot is allowed to have a collection of values rather than a single value. The slotType attribute may optionally specify a type or category for the slot.

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

6.5.2 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*.

6.5.3 Attribute slotType

Each Slot instance may have a slotType that allows different slots to be grouped together.

6.5.4 Attribute values

A Slot instance must have a Collection of values. Since a Slot represent an extensible attribute whose value may be a collection, therefore a Slot is allowed to have a collection of values rather than a single value.

7.56.6 InterfaceClass ExtrinsicObject

All Super interfaceClasses:

RegistryEntry, ObjectRegistryObject, Versionable

ExtrinsicObjects provide metadata that describes submitted content whose type is not intrinsically known to the rRegistry and therefore mustMUST be described by means of additional attributes (e.g., mime type).

Since the registry can contain arbitrary content without intrinsic knowledge about that content, ExtrinsicObjects require special metadata attributes to provide some knowledge about the object (e.g. mime type).

Examples of content described by ExtrinsicObject include *Collaboration Protocol Profiles* (*CPP*), *Bbusiness Pprocess* descriptions, and schemas.

6.6.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
contentURI	URI	Yes		Registry	No
isOpaque	Boolean	No	false	Client	No
mimeType	LongName	Yes		Client	No

Note that attributes inherited from RegistryEntry and RegistryObject are not shown in the table above.

6.6.2 Attribute contentURI

Each ExtrinsicObject instance must have a contentURI attribute. The contentURI is a URI to the repository item that is catalogued by this ExtrinsicObject instance. The contentURI is assigned by the Registry and must be resolvable by the registry.

6.6.3 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*.

6.6.4 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.

~~Note that methods inherited from the base interfaces of this interface are not shown.~~

~~7.6Interface IntrinsicObject~~

~~**All Superinterfaces:**~~

~~RegistryEntry, ObjectRegistryObject, Versionable~~

~~**All Known Subinterfaces:**~~

~~Association, Classification, ClassificationNode, ExternalLink, Organization, Package, ExternalIdentifier~~

~~IntrinsicObject serve as a common base *cClass* for derived *cClasses* that catalogue submitted content whose type is known to the *Registry* and defined by the ebXML *rRegistry* specifications.~~

~~This interface currently does not define any attributes or methods. Note that methods inherited from the base interfaces of this interface are not shown.~~

7.76.7 InterfaceClass Package

All Super Classesinterfaces:

IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

Packages allow for grouping of Llogically related registry entriesRegistryEntriesRegistryObject instances may be grouped into a Packageeven if individual member objects belong to different Submitting Organizations. It is anticipated that *Registry Services* will allow operations to be performed on an entire p*Package* of objects in the future.

6.7.1 Attribute Summary

The Package class defines no new attributes other than those that are inherited from RegistryEntry and RegistryObject base classes. The inherited attributes are not shown here.

6.7.2 Method Summary

In addition to its attributes, the Package class also defines the following methods.

Method Summary of Package	
Collection	**getMemberObjects**() Get the collection of RegistryEntriesRegistryObject instances that are members of this Package. Maps to attribute named `memberObjects`.

7.86.8 InterfaceClass ExternalIdentifier

All Super Classesinterfaces:

IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

ExternalIdentifier *i*Instancesinstances provide the additional identifier information to RegistryEntryRegistryObject such as DUNS number, Social Security Number, or an alias name of the organization. The attribute *name* inherited from ObjectRegistryObject is used to contain the identification scheme (Social Security Number, etc), and the attribute *value* contains the actual information. Each RegistryEntryRegistryObject may contain have 0 or more association(s) with ExternalIdentifier instances.

6.8.1 Attribute Summary

Attribute	Data Type	Required	Default	Specified	Mutable

			Value	By	
registryObject	UUID	Yes		Client	No
value	ShortName	Yes		Client	Yes

See Also:

Note that attributesmethods inherited from the base interfacesclasses of this interfaceclass are not shown.

6.8.2 Attribute registryObject

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

6.8.3 Attribute value

Each ExternalIdentifier instance must have a value attribute which provides the identifier value for this ExternalIdentifier (e.g. social security number).

6.8.4 Inherited Attribute id

The id attribute for an ExternalIdentifier is an attribute based id composed of the value of the registryObject and the name attributes in that order, where each attribute value is separated by a ':'.

The pattern is as follows:
urn:uuid:<RegistryObject id>:<name>

An example is as follows:

urn:uuid:a2345678-1234-1234-123456789012:Social Security Number

6.8.5 Inherited Attribute name

An ExternalIdentifier instance for a RegistryObject instance must have a unique name among all other ExternalIdentifier instances for that RegistryObject instance.

7.96.9 InterfaceClass ExternalLink

All Super Classesinterfaces:
IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

ExternalLinks use URIs to associate content in the rRegistry with content that may reside outside the rRegistry. For example, an organization submitting a *DTD* could use an ExternalLink to associate the *DTD* with the organization's home page.

6.9.1 Attribute Summary

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

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

6.9.2 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.

6.9.3 Method Summary

In addition to its attributes, the ExternalLink class also defines the following methods.

Method Summary of ExternalLink	
Collection	**getLinkedObjects**() Gets the collection of objectRegistryObjects that are linked by this ExternalLink to content outside the registryuse this external link. Maps to attribute named linkedObjects.

Note that methods inherited from the base interfacesclasses of this interfaceclass are not shown.

87 Registry Audit Trail

This chaptersection describes the information model *eElements* that support the audit trail capability of the *Registry*. Several *cClasses* in this chaptersection are *eEntity cClasses* that are used as wrappers to model a set of related attributes. These *eEntity cClasses* do not have any associated behavior. They are analogous to the “struct” construct in the C programming language.

The getAuditTrail() method of a RegistryEntryRegistryObject returns an ordered Collection of AuditableEvents. These AuditableEvents constitute the audit trail for the RegistryEntryRegistryObject. AuditableEvents include a timestamp for the *eEvent*. Each AuditableEvent has a reference to a User identifying the specific user that performed an action that resulted in an AuditableEvent. Each User is affiliated with an Organization, which is usually the *sSubmitting Organization*.

8.17.1 InterfaceClass AuditableEvent

All Superinterfac Classes:

ObjectRegistryObject

AuditableEvent iInstancesinstances provide a long-term record of eEvents that effect a change of state in a RegistryEntryRegistryObject. A RegistryEntryRegistryObject is associated with an ordered Collection of AuditableEvent iInstancesinstances that provide a complete audit trail for that ObjectRegistryObject.

AuditableEvents are usually a result of a client-initiated request. AuditableEvent iInstancesinstances are generated by the *Registry Sservice* to log such eEvents.

Often such eEvents effect a change in the life cycle of a RegistryEntryRegistryObject. For example a client request could Create, Update, Deprecate or Delete a RegistryEntryRegistryObject. No AuditableEvent is created for requests that do not alter the state of a RegistryEntryRegistryObject. Specifically, read-only requests do not generate an AuditableEvent. No AuditableEvent is generated for a RegistryEntryRegistryObject when it is classified, assigned to a Package or associated with another ObjectRegistryObject.

7.1.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
eventType	LongName	Yes		Registry	No
registryObject	UUID	Yes		Registry	No
timestamp	Timestamp	Yes		Registry	No
user	UUID	Yes		Registry	No

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

7.1.2 Attribute eventType

Each AuditableEvent must have an eventType attribute which identifies the type of event recorded by the AuditableEvent.

8.1.17.1.2.1 Pred-defined Auditable Event Types

The following table lists pre-defined auditable eevent types. These pre-defined eevent types are defined as a pre-defined *ClassificationSchemetion scheme* with

name "EventType". While thise scheme may easily be extended, a rRegistry mustMUST support the eevent types listed below.

Name	description
Created	An eEvent that created a RegistryEntryRegistryObject.
Deleted	An eEvent that deleted a RegistryEntryRegistryObject.
Deprecated	An eEvent that deprecated a RegistryEntryRegistryObject.
Updated	An eEvent that updated the state of a RegistryEntryRegistryObject.
Versioned	An eEvent that versioned a RegistryEntryRegistryObject.

7.1.3 Attribute RegistryObject

Each AuditableEvent must have a registryObject attribute that identifies the RegistryObject instance that was affected by this event.

7.1.4 Attribute timestamp

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

7.1.5 Attribute user

Each AuditableEvent must have a timestamp attribute that identifies the User that sent the request that generated this event affecting the RegistryObject instance.

Method Summary of AuditableEvent

Note that methods inherited from the base interfaces of this interface are not shown.

8.27.2 InterfaceClass User

All Super interfacesClasses:

ObjectRegistryObject

User iInstancesinstances are used in an AuditableEvent to keep track of the identity of the requestor that sent the request that generated the AuditableEvent.

7.2.1 Attribute Summary

Attribute	Data Type	Required	Default	Specified	Mutable

			Value	By	
address	PostalAddress	Yes		Client	Yes
email	LongName	Yes		Client	Yes
organization	UUID	Yes		Client	No
personName	PersonName	Yes		Client	No
telephoneNumbers	Collection of TelephoneNumber	Yes		Client	Yes
url	URI	No		Client	Yes

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

7.2.2 Attribute address

Each User instance must have an address attribute that provides the postal address for that user.

7.2.3 Attribute email

Each User instance must have an email attribute that provides the email address for that user.

7.2.4 Attribute organization

Each User instance must have an organization attribute that references the Organization instance for the organization that the user is affiliated with.

7.2.5 Attribute personName

Each User instance must have a personName attribute that provides the human name for that user.

7.2.6 Attribute telephoneNumbers

Each User instance must have a telephoneNumbers attribute that contains the Collection of TelephoneNumber instances for each telephone number defined for that user.

7.2.7 Attribute url

Each User instance may have a url attribute that provides the URL address for the web page associated with that user.

8.37.3 InterfaceClass Organization

All Super CinterfacesIasses:

IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

Organization *iInstances*instances provide information on organizations such as a *Submitting Organization*. Each Organization *iInstance* may have a reference to a parent Organization.

7.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
address	PostalAddress	Yes		Client	Yes
parent	UUID	No		Client	Yes
primaryContact	UUID	Yes		Client	No
telephoneNumbers	Collection of TelephoneNumber	Yes		Client	Yes

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

7.3.2 Attribute address

Each Organization instance must have an address attribute that provides the postal address for that organization.

7.3.3 Attribute parent

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

7.3.4 Attribute primaryContact

Each Organization instance must have a primaryContact attribute that references the User instance for the user that is the primary contact for that organization.

7.3.5 Attribute telephoneNumbers

Each Organization instance must have a telephoneNumbers attribute that contains the Collection of TelephoneNumber instances for each telephone number defined for that organization.In addition it may have a contact attribute defining the primary contact within the organization. An Organization also has an address attribute.

See Also:

Note that methods inherited from the base interfaces of this interface are not shown.

7.4 *Class* PostalAddress

Class PostalAddress

PostalAddress is a simple reusable *eEntity cClass* that defines attributes of a postal address.

7.4.1 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	No
state	ShortName	No		Client	Yes
street	ShortName	No		Client	Yes

7.4.2 Attribute city

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

7.4.3 Attribute country

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

7.4.4 Attribute postalCode

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

7.4.5 Attribute state

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

7.4.6 Attribute street

Each PostalAddress may have a street attribute identifying the street address for that address.

7.4.7 Method Summary

In addition to its attributes, the PostalAddress class also defines the following methods.

Method Summary of ExternalLink	
Collection	**getSlots()** Gets the collection of Slots for this object. Each PostalAddress may have multiple Slot instances where a Slot is a dynamically defined attribute. The use of Slots allows the client to extend PostalAddress class by defining additional dynamic attributes using slots to handle locale specific needs.

8.57.5 *Class* TelephoneNumber

A simple reusable *eEntity cClass* that defines attributes of a telephone number.

7.5.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
areaCode	String4	No		Client	Yes
countryCode	String4	No		Client	Yes
extension	String8	No		Client	Yes
number	String8	No		Client	Yes
phoneType	LongName	No		Client	Yes
url	URI	No		Client	Yes

7.5.2 Attribute areaCode

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

7.5.3 Attribute countryCode

Each TelephoneNumber instance may have an countryCode attribute that provides the country code for that telephone number.

7.5.4 Attribute extension

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

7.5.5 Attribute number

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

7.5.6 Attribute phoneType

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

7.5.7 Attribute url

Each TelephoneNumber instance may have a url attribute that provides the url, if any, associated with that telephone number. It is an anticipated that it will be possible to dial telephone numbers via URLs sometime in the future. Do we need this or should we remove it??-

8.67.6 *Class* PersonName

A simple *eEntity eClass* for a person's name.

7.6.1 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

7.6.2 Attribute firstName

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

7.6.3 Attribute lastName

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

7.6.4 Attribute middleName

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

---

98 Registry-ObjectEntry Naming

A RegistryEntryRegistryObject has a name that may or may not be unique within the *Registry*.

In addition a RegistryEntryRegistryObject may have any number of context sensitive alternate names that are valid only in the context of a particular *cClassification* scheme. Alternate contextual naming will be addressed in a later version of the Registry Information Model.

<FSN: We should remove this chapter??>

109 Association of Registry-Entryies

A RegistryEntryRegistryObject may be associated with 0 or more objectRegistryObjects. The information model defines an Association *cClass*. An *iInstance* of the Association *cClass* represents an association between a source RegistryEntryRegistryObject and another target ObjectRegistryObject. An example of such an association is between an ExtrinsicObject instances that catalogues a new *Collaboration Protocol Profile* (*CPP*) and another ExtrinsicObject instance that catalogues an older *Collaboration Protocol Profile* where the newer *CPP* supersedes the older *CPP* as shown in Figure 3Figure 3Figure 3Figure 3Figure 3.

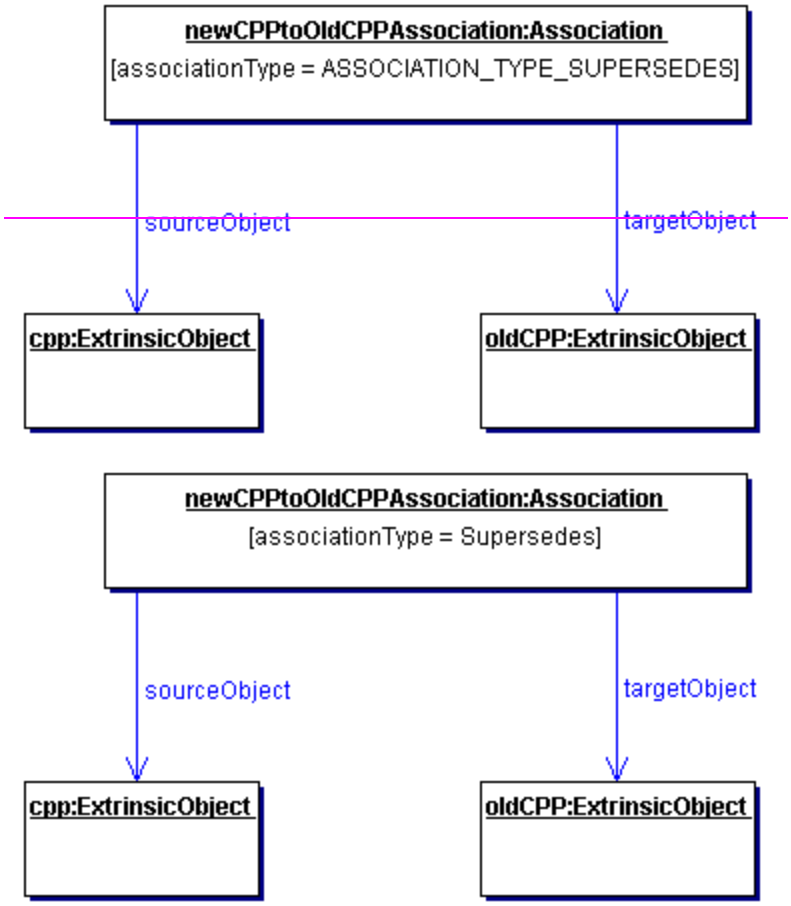


Figure 333: Example of Registry Entry Association

10.19.1 InterfaceClass *Association*

All Super interfacesClasses:

IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

All Known Subinterfaces:

Classification

Association iInstancesinstances are used to define many-to-many associations between objectRegistryObjects in the information model.

An *iInstance* of the Association cClass represents an association between two ObjectRegistryObjects.

9.1.1 Attribute Summary

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

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

9.1.2 Attribute associationType

Each Association must have an associationType attribute that identifies the type of that association. This MUST be the name attribute of an association type as defined by 1.1.1.

9.1.2.1 Pre-defined Association Types

The following table lists pre-defined association types. These pre-defined association types are defined as a *Classification* scheme. While the scheme may easily be extended a *Registry* MUST support the association types listed below.

name	description
RelatedTo	Defines that source RegistryObject is related to target RegistryObject.
HasMember	Defines that the source Package object has the target RegistryObject object as a member. Reserved for use in Packaging of RegistryEntries.
ExternallyLinks	Defines that the source ExternalLink object externally links the target RegistryObject object. Reserved for use in associating ExternalLinks with RegistryEntries.
ExternallyIdentifies	Defines that the source ExternalIdentifier object identifies the target RegistryObject object. Reserved for use in associating ExternalIdentifiers with RegistryEntries.
ContainedBy	Defines that source RegistryObject is contained by the target RegistryObject.
Contains	Defines that source RegistryObject contains the target RegistryObject.
Extends	Defines that source RegistryObject inherits from or specializes the target RegistryObject.
Implements	Defines that source RegistryObject implements the functionality defined by the target RegistryObject.
InstanceOf	Defines that source RegistryObject is an *Instance* of

	target RegistryObject.
SupersededBy	Defines that the source RegistryObject is superseded by the target RegistryObject.
Supersedes	Defines that the source RegistryObject supersedes the target RegistryObject.
UsedBy	Defines that the source RegistryObject is used by the target RegistryObject in some manner.
Uses	Defines that the source RegistryObject uses the target RegistryObject in some manner.
ReplacedBy	Defines that the source RegistryObject is replaced by the target RegistryObject in some manner.
Replaces	Defines that the source RegistryObject replaces the target RegistryObject in some manner.

9.1.3 Attribute sourceObject

Each Association must have a sourceObject attribute that references the RegistryObject instance that is the source or owner of that association.

9.1.4 Attribute targetObject

Each Association must have an targetObject attribute that references the RegistryObject instance that is the target of that association.

9.1.5 Inherited Attribute id

The id attribute for an Association is an attribute based id composed of the value of the sourceObject, targetObject and associationType attributes in that order, where each attribute value is separated by a ':'.

The pattern is as follows:
urn:uuid:< sourceObject id>:< targetObject id>:<associationType>

An example is as follows:

urn:uuid:a2345678-1234-1234-123456789012: a2345678-1234-1234-123456789013:Implements

~~10.1.1Pre-defined Association Types~~

~~The following table lists pre-defined association types. These pre-defined association types are defined as a *Classification* scheme. While the scheme may~~

easily be extended a *rRegistry* mustMUST support the association types listed below.

name	description
RelatedTo	Defines that source objectRegistryObject is an related toinstance of target objectRegistryObject.
PHasMemberackages	Defines that the source Package object haspackages the target RegistryEntry object as a member. Reserved for use in Packaging of Registry Entries.
ExternallyLinks	Defines that the source ExternalLink object externally links the target RegistryEntry object. Reserved for use in associating ExternalLinks with Registry Entries.
ExternallyIdentifies	Defines that the source ExternalIdentifier object identifies the target RegistryEntry object. Reserved for use in associating ExternalIdentifiers with Registry Entries.
ContainedBy	Defines that source objectRegistryObject is contained by the target objectRegistryObject.
Contains	Defines that source objectRegistryObject contains the target objectRegistryObject.
Extends	Defines that source objectRegistryObject inherits from or specializes the target objectRegistryObject.
Implements	Defines that source objectRegistryObject implements the functionality defined by the target objectRegistryObject.
InstanceOf	Defines that source objectRegistryObject is an *iInstance* of target objectRegistryObject.
SupersccededBy	Defines that the source objectRegistryObject is superseded by the target objectRegistryObject.
Supersccedes	Defines that the source objectRegistryObject supersedes the target objectRegistryObject.
UsedBy	Defines that the source objectRegistryObject is used by the target objectRegistryObject in some manner.
Uses	Defines that the source objectRegistryObject uses the target objectRegistryObject in some manner.
ReplacedBy	Defines that the source objectRegistryObject is replaced by the target objectRegistryObject in some manner.
Replaces	Defines that the source objectRegistryObject replaces the target objectRegistryObject in some manner.

–

In some association types, such as Extends and

~~Implements, although the association is between RegistryObjects, the actual relationship specified by that type is between repository items pointed by RegistryObjects.~~

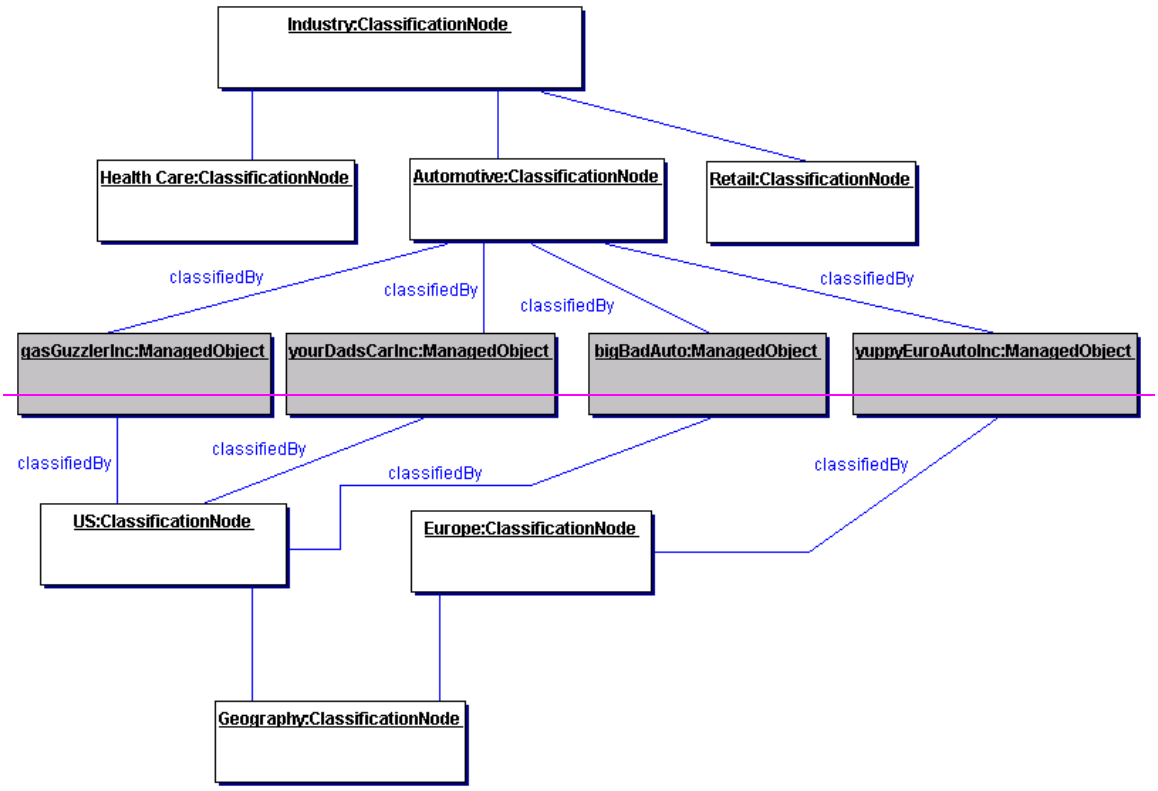
1110*Classification* of Registry ObjectEntryies

This section describes the how the information model supports a*C*lassification of RegistryEntryRegistryObjecties. It is a simplified version of the *OASIS* classification model [OAS].

A RegistryEntryRegistryObject may be classified in many ways. For example the RegistryEntryRegistryObject 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 a*C*lassification scheme can be viewed as a a*C*lassification tree. In the example shown in Figure 4Figure 4Figure 4Figure 4Figure 4, RegistryEntriesRegistryObject 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 root ClassificationNodeClassificationScheme instance with named "Industry". Furthermore, the US Automobile manufacturers are classified by the US ClassificationNode under the Geography ClassificationScheme with name "Geography"Node. Similarly, a European automobile manufacturer is classified by the "Europe" ClassificationNode under the Geography ClassificationScheme with name "Geography"Node.

The example shows how a RegistryEntryRegistryObject may be classified by multiple cClassificationNode instances under multiple ClassificationSscheme instances. A a*Classification* scheme is defined by a ClassificationNode that is the root of a a*Classification* tree (e.g. Industry, Geography).



1403

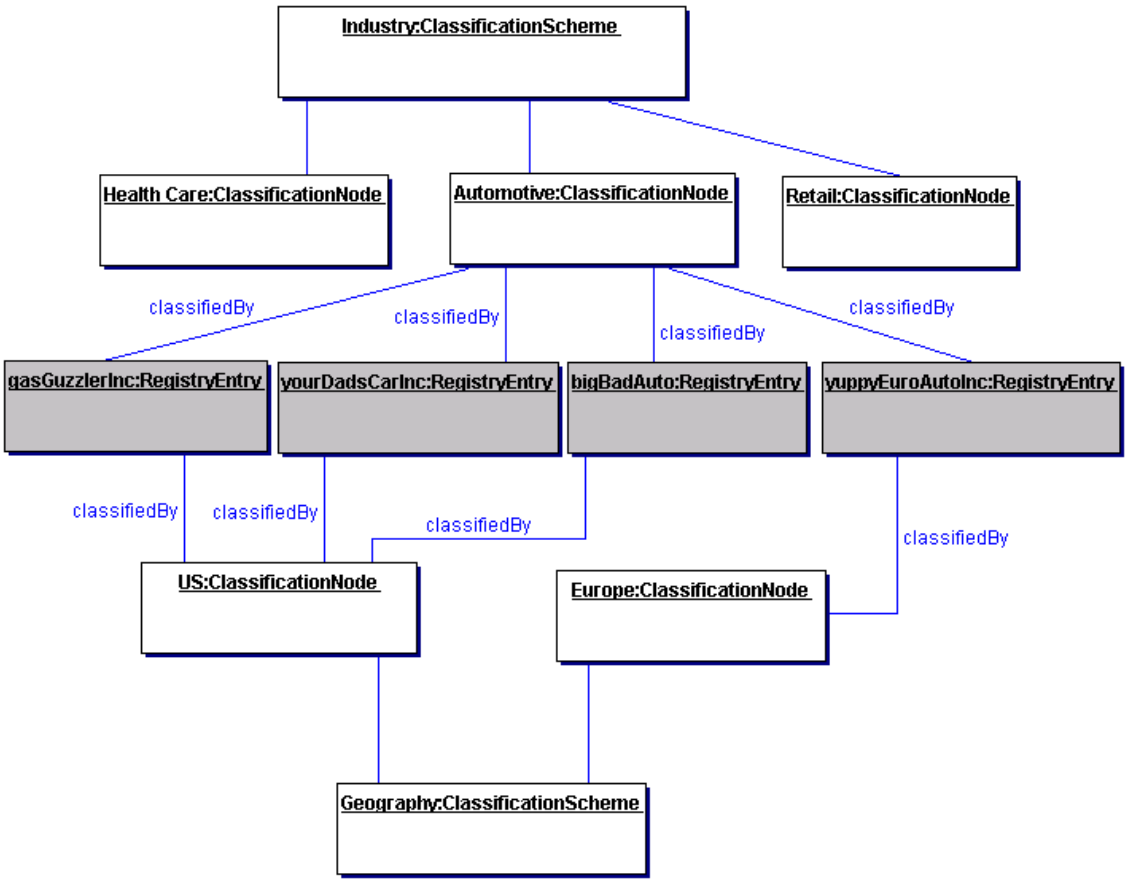


Figure 44: Example showing a *Classification* Tree

```
[Note]It is important to point out that the dark
     nodes (gasGuzzlerInc, yourDadsCarInc etc.) are
     not part of the cClassification tree. The leaf
     nodes of the cClassification tree are Health
     Care, Automotive, Retail, US and Europe. The
     dark nodes are associated with the
     cClassification tree via a Classification
     iInstance that is not shown in the picture
```

In order to support a general *cClassification* scheme that can support single level as well as multi-level *cClassifications*, the information model defines the *cClasses* and relationships shown in Figure 5Figure 5Figure 5Figure 5Figure 5.

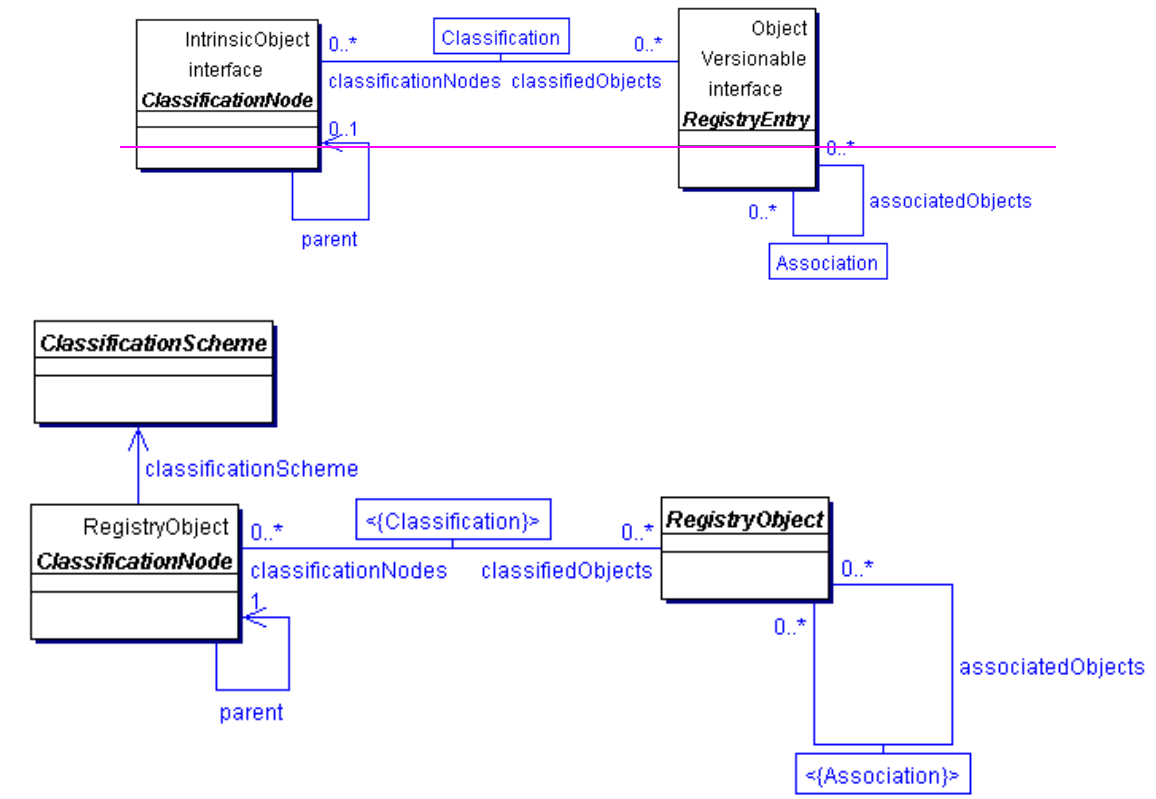


Figure 555: Information Model *Classification* View

A Classification is somewhat like a specialized form of an Association. Figure 6Figure 6Figure 6Figure 6Figure 6 shows an example of an ExtrinsicObject *iInstance* for a *Collaboration Protocol Profile* (*CPP*) object that is classified by a ClassificationNode representing the Industry that it belongs to.

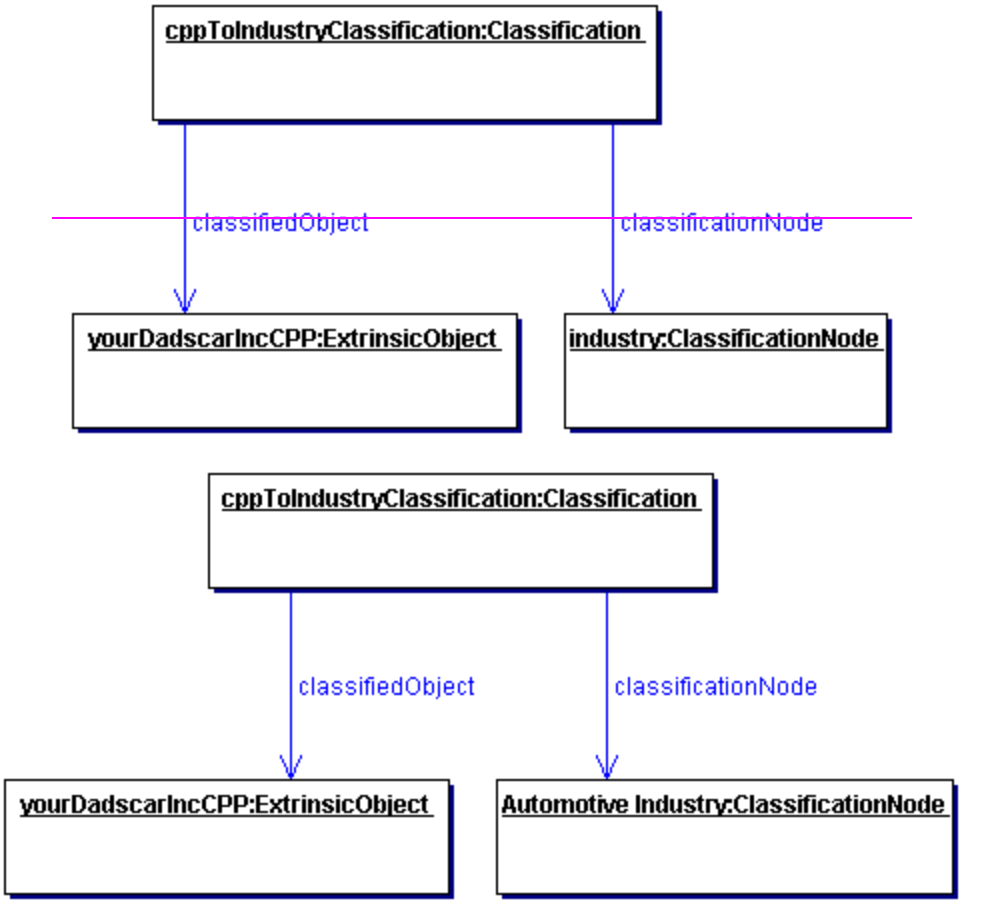


Figure ~~666~~: Classification *Instance* Diagram

10.1 Class ClassificationScheme

A ClassificationScheme instance defines the root of a classification hierarchy where the nodes of the tree are composed of ClassificationNodes instances. Currently the ClassificationScheme class does not define any attributes other than the attributes inherited from RegistryObject and RegistryEntry base classes.

<FSN: Do we need any attributes or methods for this class?? Also do we need to change the name of GetRootClassificationNodesRequest/Response to GetClassificationSchemeRequest/Response in the RS 1.1??>

~~11.1~~10.2 ~~Interface~~Class ClassificationNode

Base ~~All Superinterfaces~~classes:

~~IntrinsicObject, RegistryEntry,~~ ~~Object~~RegistryObject~~, Versionable~~

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

See Also:

Classification

10.2.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
parent	UUID	Yes		Client	No
code	ShortName	No		Client	No

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

10.2.2 Attribute parent

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

10.2.3 Attribute code

Each ClassificationNode may have a code attrubite. The code attribute contains a code within a standard coding scheme as described in section 10.5.

10.2.4 Method Summary

In addition to its attributes, the Package class also defines the following methods.

Method Summary of ClassificationNode	
ClassificationScheme	**getClassificationScheme**() Get the ClassificationScheme that this this ClassificationNode belongs to.
Collection	**getClassifiedObjects**() Get the collection of RegistryObjectsEntries classified by this ClassificationNode. Maps to attribute named classifiedObjects.
String	**getPath**() Gets the path from the ClassificationSchemeroot ancestor of this ClassificationNode. The path conforms to the [XPATH] expression syntax (e.g "/Geography/Asia/Japan"). Maps to attribute named path.

Note that methods inherited from the base interfacesclasses of this interfaceclass are not shown.

In Figure 4Figure 4Figure 4Figure 4Figure 4, several *i*Instancesinstances of ClassificationNode are defined (all light colored boxes). A ClassificationNode has exactlyzero or one ClassificationNodes for its parent and zero or more ClassificationNodes for its immediate children. The parent of a ClassificationNodes may be another ClassificationNodes or a ClassificationScheme in case of first level ClassificationNodes.If a ClassificationNode has no parent then it is the root of a *c*Classification tree. Note that the entire *c*Classification tree is recursively defined by a single information model *e*Element ClassificationNode.

11.210.3 InterfaceClass Classification

Base CAll Superinterfaceslasses:

IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

---

Classification *i*Instancesinstances are used to classify repository item by associating their RegistryEntryRegistryObject *i*Instance with a ClassificationNode *i*Instance within a *c*Classification scheme.

In Figure 4Figure 4Figure 4Figure 4Figure 4, Classification *i*Instancesinstances are not explicitly shown but are implied as associations between the RegistryEntriesRegistryObject instances (shaded leaf node) and the associated ClassificationNode.

10.3.1 Attribute Summary

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classificationNode	UUID	Yes		Client	No
classifiedObject	UUID	Yes		Client	No

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

10.3.2 Attribute classificationNode

Each Classification instance must have a classificationNode attribute that references the ClassificationNode instance that is used to classify a RegistryObject specified by the classifiedObject attribute. This is similar to the targetObject attribute in an Association instance.

10.3.3 Attribute classifiedObject

Each Classification instance must have a classifiedObject attribute that references the RegistryObject instance that is classified by this Classification. This is similar to the sourceObject attribute in an Association instance.

10.3.4 Inherited Attribute id

The id attribute for a Classificiation is an attribute based id composed of the value of the classifiedObject and the classificationNode attributes in that order, where each attribute value is separated by a ':'.

The pattern is as follows:
urn:uuid:<classifiedObject id>:< classificationNode id>

An example is as follows:

urn:uuid:a2345678-1234-1234-123456789012: a2345678-1234-1234-123456789013
~~Note that methods inherited from the base interfaces of this interface are not shown.~~

~~11.2.1~~10.3.5 Context Sensitive *Classification*

Consider the case depicted in Figure 7~~Figure 7~~~~Figure 7~~Figure 7~~Figure 7~~ where a *Collaboration Protocol Profile* for ACME Inc. is classified by the Japan ClassificationNode under the Geography ~~c~~*Classification* scheme. In the absence of the context for this ~~c~~*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.

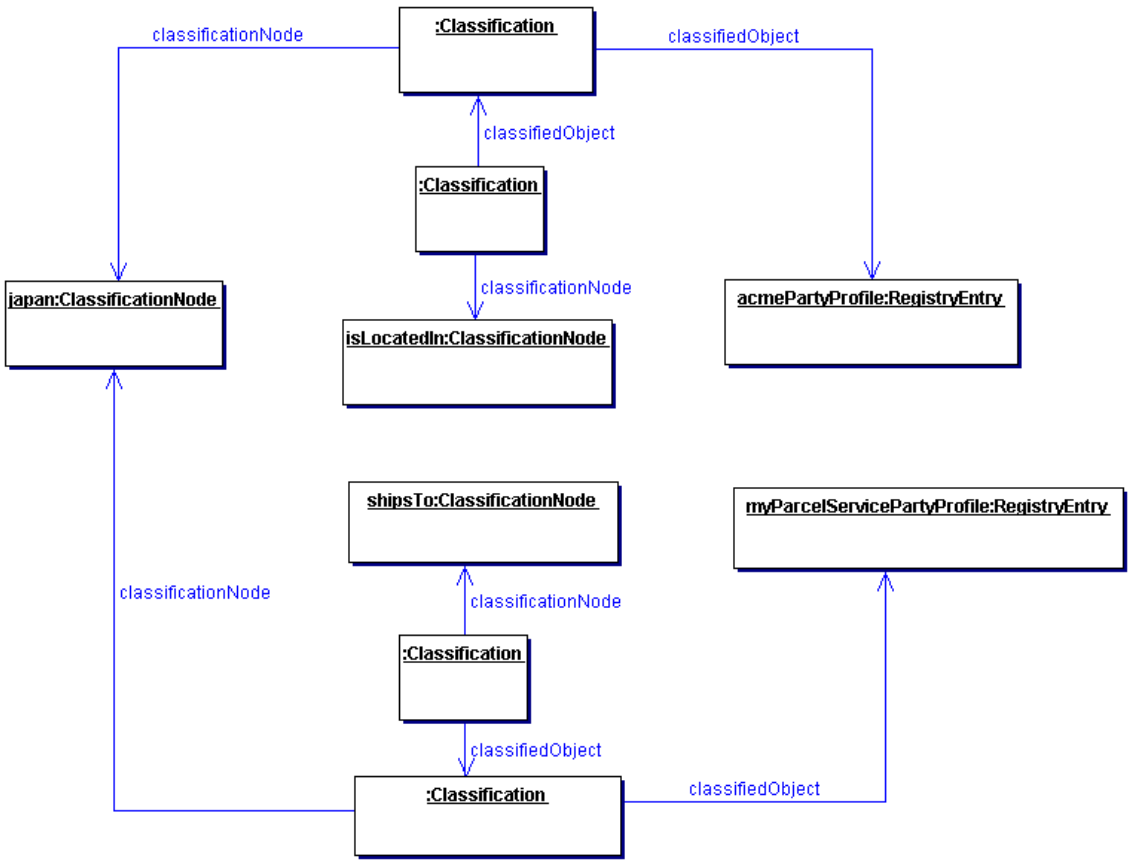


Figure 77: Context Sensitive *Classification*

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 *cClassification* schemes in the information model allows:

- A RegistryEntryRegistryObject to be classified by defining a Classification that associates it with a ClassificationNode in a *cClassification* tree.
- A RegistryEntryRegistryObject to be classified along multiple facets by having multiple *cClassifications* that associate it with multiple ClassificationNodes.
- A *cClassification* defined for a RegistryEntryRegistryObject to be qualified by the contexts in which it is being classified.

11.310.4 Example of *Classification* Schemes

The following table lists some examples of possible *cClassification* schemes 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 (Context)**	**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 *Bbusiness* 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 *rRole* of "Seller"	

Table 1: Sample *Classification* Schemes

11.410.5 Standardized Taxonomy Support

Standardized taxonomies also referred to as ontologies, classification schemes, or or coding schemes exist in various industries to provide a structured coded vocabulary. The ebXML *rRegistry* does not define support for specific taxonomies. Instead it provides a general capability to link RegistryEntrieItems to codes defined by various taxonomies.

The information model provides two alternatives for using standardized taxonomies for *cClassification* of RegistryEntrieItems.

11.4.110.5.1 Full-featured Taxonomy Based *Classification*

The information model provides a full-featured taxonomy based *cClassification* alternative based on Classification, ClassificationScheme and ClassificationNode *iInstances*instances. This alternative requires that a standard taxonomy be imported into the *Registry* as a *cClassification* tree consisting of ClassificationNode *iInstances*instances rooted under a ClassificationScheme instance. This specification does not prescribe the transformation tools necessary to convert standard taxonomies into ebXML *Registry cClassification* trees. However, the transformation mustMUST ensure that:

1. o The name attribute of the root ClassificationNodeClassificationScheme instance is the *name* of the standard taxonomy (e.g. NAICS, ICD-9, SNOMED).
2. o All codes in the standard taxonomy are preserved in the *code* attribute of a ClassificationNode.

3. o The intended structure of the standard taxonomy is preserved in the ClassificationNode tree, thus allowing polymorphic browse and drill down discovery. This means that whenis searching for entries classified by Asia, a client will find entries classified by descendants of Asia (e.g. Japan and Korea).

11.4.210.5.2 Light Weight Taxonomy Based *Classification*

<FSN: This section will be reworked based on the classification sub-team proposal??>

The information model also provides a lightweight alternative for classifying RegistryEntryRegistryObject iInstancesinstances by codes defined by standard taxonomies, where the submitter does not wish to import an entire taxonomy as a native *cClassification* scheme.

In this alternative the submitter adds one or more taxonomy related Slots to the RegistryEntryRegistryObject for a submitted repository item. Each Slot's name identifies a standardized taxonomy while the Slot's value is the code within the specified taxonomy. Such taxonomy related Sslots mustMUST be defined with a slotType of `Classification`.

For example if a RegistryEntryRegistryObject has a Slot with name "NAICS", a slotType of "Classification" and a value "51113" it implies that the RegistryEntryRegistryObject is classified by the code for "Book Publishers" in the NAICS taxonomy. Note that in this example, there is no need to import the entire NAICS taxonomy, nor is there any need to create iInstancesinstances of ClassificationNode or Classification.

The following points are noteworthy in this light weight *cClassification* alternative:

- ?o Validation of the name and the value of the Classification" is responsibility of the SO and not of the ebXML *Registry* itself.
- ?o Discovery is based on exact match on slot name and slot value rather than the flexible "browse and drill down discovery" available to the heavy weight *cClassification* alternative.

1211Information Model: Security View

This chaptersection describes the aspects of the information model that relate to the security features of the *Registry*.

<FSN: Thos chapter will be updated based on output from the security sub-team?? It is therefore not updated in format to the new format for RIM 1.1>

Figure 8Figure 8Figure 8Figure 8Figure 8 shows the view of the objects in the *Registry* from a security perspective. It shows object relationships as a *UML cClass* diagram. It does not show *cClass* attributes or *cClass* methods that will be described in subsequent sections. It is meant to be illustrative not prescriptive.

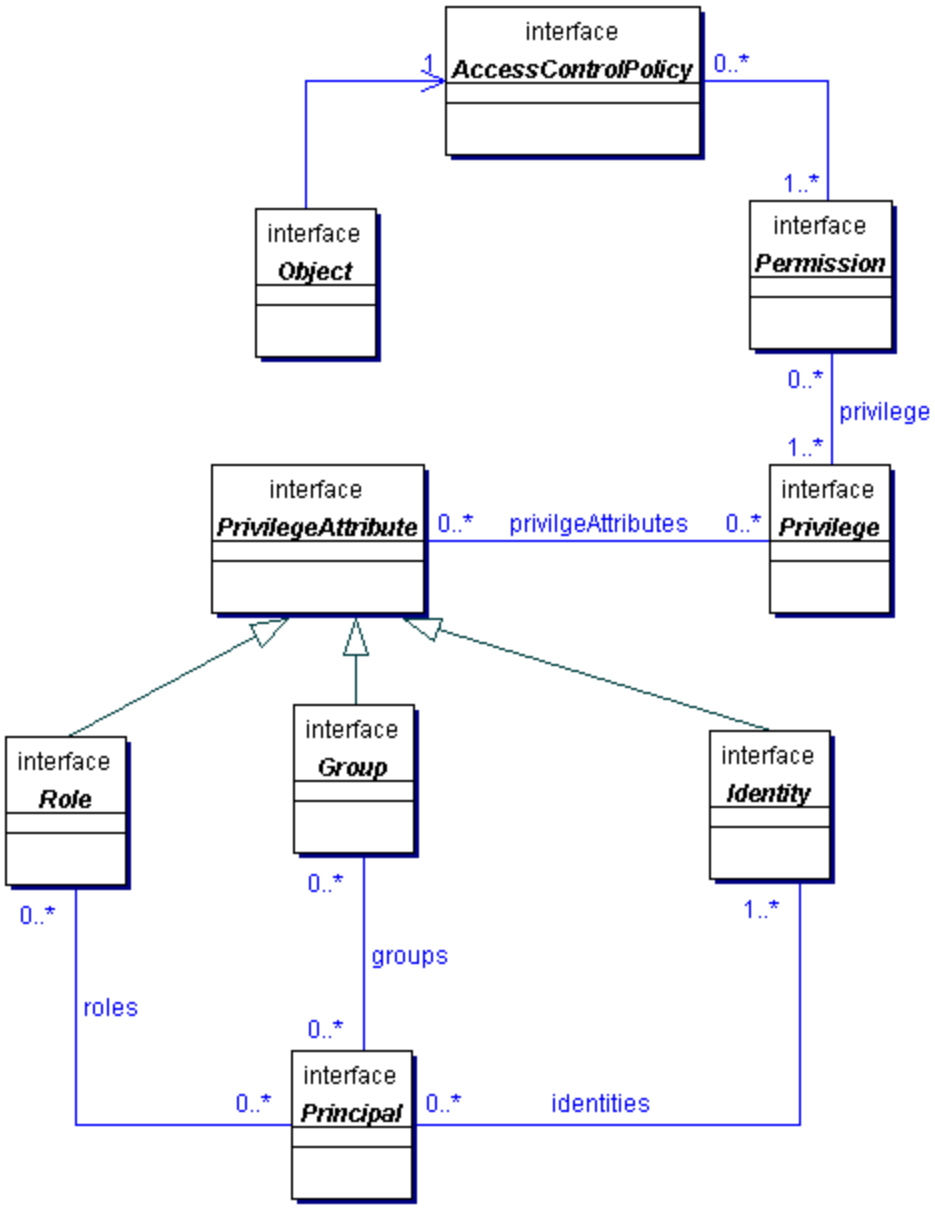


Figure 888: Information Model: Security View

12.111.1 InterfaceClass AccessControlPolicy

Every ObjectRegistryObject is associated with exactly one AccessControlPolicy which defines the policy rules that govern access to operations or methods performed on that ObjectRegistryObject. Such policy rules are defined as a collection of Permissions.

Method Summary of AccessControlPolicy	
Collection	**getPermissions**() Gets the Permissions defined for this AccessControlPolicy. Maps to attribute named `permissions`.

12.211.2 InterfaceClass Permission

The Permission object is used for authorization and access control to ObjectRegistryObjects in the *Registry*. The Permissions for an ObjectRegistryObject are defined in an AccessControlPolicy object.

A Permission object authorizes access to a method in an ObjectRegistryObject if the requesting Principal has any of the Privileges defined in the Permission.

See Also:

`Privilege`, `AccessControlPolicy`

Method Summary of Permission	
String	**getMethodName**() Gets the method name that is accessible to a Principal with specified Privilege by this Permission. Maps to attribute named `methodName`.
Collection	**getPrivileges**() Gets the Privileges associated with this Permission. Maps to attribute named `privileges`.

12.311.3 InterfaceClass Privilege

A Privilege object contains zero or more PrivilegeAttributes. A PrivilegeAttribute can be a Group, a Role, or an Identity.

A requesting Principal mustMUST have all of the PrivilegeAttributes specified in a Privilege in order to gain access to a method in a protected ObjectRegistryObject. Permissions defined in the ObjectRegistryObject's AccessControlPolicy define the Privileges that can authorize access to specific methods.

This mechanism enables the flexibility to have object access control policies that are based on any combination of Roles, Identities or Groups.

See Also:

`PrivilegeAttribute`, `Permission`

Method Summary of Privilege	
`Collection`	**`getPrivilegeAttributes`**`()` Gets the PrivilegeAttributes associated with this Privilege. Maps to attribute named `privilegeAttributes`.

12.411.4 InterfaceClass PrivilegeAttribute

All Known Subinterfacesclasses:

Group, Identity, Role

PrivilegeAttribute is a common base cClass for all types of security attributes that are used to grant specific access control privileges to a Principal. A Principal may have several different types of PrivilegeAttributes. Specific combination of PrivilegeAttributes may be defined as a Privilege object.

See Also:

`Principal`, `Privilege`

12.511.5 InterfaceClass Role

All Superinterfacesclasses:

PrivilegeAttribute

A security Role PrivilegeAttribute. For example a hospital may have *Roles* such as Nurse, Doctor, Administrator etc. Roles are used to grant Privileges to Principals. For example a Doctor rRole may be allowed to write a prescription but a Nurse rRole may not.

12.611.6 InterfaceClass Group

All Superinterfacesclasses:

PrivilegeAttribute

A security Group PrivilegeAttribute. A Group is an aggregation of users that may have different rRoles. For example a hospital may have a Group defined for Nurses and Doctors that are participating in a specific clinical trial (e.g. AspirinTrial group). Groups are used to grant Privileges to Principals. For example the members of the AspirinTrial group may be allowed to write a prescription for Aspirin (even though Nurse rRole as a rule may not be allowed to write prescriptions).

12.711.7 InterfaceClass Identity

All Superinterfacesclasses:

PrivilegeAttribute

A security Identity PrivilegeAttribute. This is typically used to identify a person, an organization, or software service. Identity attribute may be in the form of a digital certificate.

12.811.8 InterfaceClass Principal

Principal is a completely generic term used by the security community to include both people and software systems. The Principal object is an entity that has a set of PrivilegeAttributes. These PrivilegeAttributes include at least one identity, and optionally a set of rrole memberships, group memberships or security clearances. A principal is used to authenticate a requestor and to authorize the requested action based on the PrivilegeAttributes associated with the Principal.

See Also:

`PrivilegeAttributes`, `Privilege`, `Permission`

Method Summary of Principal	
`Collection`	**`getGroups`**`()` Gets the Groups associated with this Principal. Maps to attribute named `groups`.
`Collection`	**`getIdentities`**`()` Gets the Identities associated with this Principal. Maps to attribute named `identities`.
`Collection`	**`getRoles`**`()` Gets the Roles associated with this Principal. Maps to attribute named `roles`.

1312References

[ebGLOSSGLS] ebXML Glossary,
http://www.ebxml.org/documents/199909/terms_of_reference.htm

[ebTA] ebXML Technical Architecture Specification
http://www.ebxml.org/specdrafts/ebXML_TA_v1.0.4.pdf

[OAS] OASIS Information Model
http://xsun.sdct.itl.nist.govwww.nist.gov/itl/div897/ctg/regrep/OasisRegrepSpec.pdfoasis-work.html

[ISO] ISO 11179 Information Model
http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba7852566210054419d7/b83fc7816a6064c68525690e0065f913?OpenDocument

[BRA97] IETF (Internet Engineering Task Force). RFC 2119: Key words for use in RFCs to Indicate Requirement Levels
http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2119.htmlDM] Registry and Repository: Business Domain Model
http://www.ebxml.org/specdrafts/RegRepv1-0.pdf

[ebRS] ebXML Registry Services Specification
http://www.ebxml.org/specdrafts/ebXML_RS_v1.0.pdf
http://www.ebxml.org/project_teams/registry/private/RegistryServicesSpecificationv0.83.pdf

[BPMebBPSS] ebXML Business Process Metamodel Specification Schema
http://www.ebxml.org/specdrafts/Busv2-0.pdf

[ebCPPCPA] ebXML Collaboration-Protocol Profile and Agreement Trading-Partner Specification
http://www.ebxml.org/specfrafts/project_teams/trade_partner/private/

[CTB] Context table informal document from Core Components
http://www.ebxml.org/project_teams/core_components/ContextTable.doc

[UUID] DCE 128 bit Universal Unique Identifier
http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
http://www.opengroup.org/publications/catalog/c706.htmttp://www.w3.org/TR/REC-xml

[XPATH] XML Path Language (XPath) Version 1.0
http://www.w3.org/TR/xpath

1413Disclaimer

The views and specification expressed in this document are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this design.

1514Contact Information

Team Leader

Name:	Lisa CarnahanScott Nieman
Company:	NISTNorstan Consulting
Street:	100 Bureau Drive STOP 8970 5101 Shady Oak Road
City, State, Postal Code:	Gaithersburg, MD 20899-8970 Minnetonka, MN 55343
Country:	USA
Phone:	301-975-3362 952.352.5889
Email:	lisa.carnahan@nist.govScott.Nieman@Norstan

EditorVice Team Lead

Name:	Sally FugerYutaka Yoshida
Company:	<Need Sally's contact info??>Sun Microsystems
Street:	901 San Antonio Road, MS UMPK17-102
City, State, Postal Code:	Palo Alto, CA 94303
Country:	USA
Phone:	650.786.5488
Email:	Yutaka.Yoshida@eng.sun.com

sfuger@aiag.org

Technical Editor

Name:	Farrukh S. Najmi
Company:	Sun Microsystems
Street:	1 Network Dr., MS BUR02-302
City, State, Postal Code:	Burlington, MA, 01803-0902
Country:	USA
Phone:	781.442.0703
Email:	najmi@east.sun.com

Copyright Statement

Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved
Copyright © ebXML 2000 & 2001 All Rights Reserved.

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 MAY not be modified in any way, such as by removing the copyright notice or references to ebXML, UN/CEFACT, or OASIS, except as required to translate it into languages other than English.

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

This document and the information contained herein is provided on an "AS IS" basis and ebXML 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.
Copyright © ebXML 2000. All Rights Reserved.

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 may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process 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 ebXML or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and ebXML 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.~~