ebXML Registry Information Model

ebXML Registry Working Group

Working Draft 12/27/2000 4:40 AM

This version: Version 0.52

1 Status of this Document

This document specifies an ebXML 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/project_teams/registry/private/registryInfoModelv0.52.pdf

Latest version:
http://www.ebxml.org/project_teams/registry/private/registryInfoModelv0.52.pdf

Previous version:
http://www.ebxml.org/project_teams/registry/private/registryInfoModelv0.51.pdf
2 ebXML participants

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.

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
Bob Miller - GXS
Kunio Mizoguchi - Electronic Commerce Promotion Council of Japan
Ron Monzillo – Sun Microsystems
Joel Munter - Intel
Farrukh Najmi - Sun Microsystems
Scott Nieman - Norstan Consulting
Frank Olken – Lawrence Berkeley National Laboratory
Micheal Park - eSum Technologies
Bruce Peat - eProcess Solutions
Waqar Sadiq - Vitria
Krishna Sankar - CISCO
Kim Tae Soo - Government of Korea
Nikola Stojanovic -
David Webber - XML Global
Yutaka Yoshida - Sun Microsystems
Prasad Yendluri - webmethods
Peter Z. Zhoo - Knowledge For the new Millennium
Table of Contents

1 STATUS OF THIS DOCUMENT .................................................................1

2 EBXML PARTICIPANTS ...........................................................................2

3 INTRODUCTION ........................................................................................6

  3.1 SUMMARY OF CONTENTS OF DOCUMENT ..........................................6
  3.2 DOCUMENT VERSION HISTORY .........................................................6
  3.3 GENERAL CONVENTIONS ...............................................................6
  3.4 AUDIENCE .......................................................................................7
  3.5 RELATED DOCUMENTS .....................................................................7

4 DESIGN OBJECTIVES ................................................................................7

  4.1 GOALS ...........................................................................................7
  4.2 CAVEATS AND ASSUMPTIONS .........................................................8

5 SYSTEM OVERVIEW .................................................................................8

  5.1 ROLE OF EBXML REGISTRY .............................................................8
  5.2 REGISTRY SERVICES .......................................................................8
  5.3 WHAT THE REGISTRY INFORMATION MODEL DOES .........................8
  5.4 HOW THE REGISTRY INFORMATION MODEL WORKS .......................8
  5.5 WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED ...9

6 REGISTRY INFORMATION MODEL: PUBLIC VIEW ....................................9

  6.1 MANAGEDOBJECT ..........................................................................10
  6.2 ASSOCIATION ................................................................................10
  6.3 EXTERNALLINK ..............................................................................10
  6.4 CLASSIFICATIONNODE .................................................................10
  6.5 CLASSIFICATION ..........................................................................10
  6.6 PACKAGE ......................................................................................10
  6.7 AUDITABLEEVENT .........................................................................10
  6.8 POSTALADDRESS .........................................................................11
  6.9 CONTACT ......................................................................................11
  6.10 ORGANIZATION ...........................................................................11

7 REGISTRY INFORMATION MODEL: DETAIL VIEW ...................................11

  7.1 INTERFACE OBJECT .......................................................................12
  7.2 INTERFACE VERSIONABLE ............................................................13
  7.3 INTERFACE MANAGEDOBJECT .....................................................14
  7.4 INTERFACE EXTRINSICOBJECT EXTRINSICOBJECT .........................15
  7.4.1 Pre-Defined Extrinsic Object Types .............................................16
  7.5 INTERFACE INTRINSICOBJECT INTRINSICOBJECT ..........................17
Table of Figures

135 Table of Figures
136 Figure 1: Information Model Public View.................................................................9
137 Figure 2 ..........................................................................................................................12
138 Figure 3 ..........................................................................................................................12
139 Figure 4: Information Model Inheritance View..........................................................12
140 Figure 5: Example of Managed Object Association ..................................................23
141 Figure 6: Example showing a Classification Tree......................................................26
142 Figure 7: Information Model Classification View.......................................................27
143 Figure 8: Classification Instance Diagram .................................................................27
144 Figure 9: Context Sensitive Classification................................................................29
145 Figure 10: Information Model: Security View.............................................................32

Table of Tables

146 Table of Tables
147 Table 1: Pre-defined Classification Schemes...............................................................30

148

149
3 Introduction

3.1 Summary of Contents of Document
This document specifies the information model for the ebXML Registry.

A separate document, *ebXML Registry Services Specification* [5], describes how
to build Registry Services that provide access to the information content in the
 ebXML Registry.

3.2 Document Version History
- Version 0.1: Initial version
- Version 0.2: Changes based on review. Simplified OASIS classification
  model. Eliminated ambiguity between managed object content (the content)
  and ManagedObject the metadata about the content. Changes to Object
  class hierarchy and attributes.
- Version 0.3: Changes based on review dated 9/29/2000. Changed “managed
  object” to “managed object content”. Made Package a ManagedObject. Made
  ManagedObject to Submission association a many to many association.
  Logged post-Tokyo issues raised.
- Version 0.41: Changes based on Tokyo face-to-face meeting. Added context
  sensitive classifications, reformatted document to conform to ebXML
  document standard.
- Version 0.5: Major re-write. Cleanup of base information model. Added
  IntrinsicObject and ExtrinsicObject. Changed ExternalObject to ExternalLink.
  Added security information model.
- Version 0.51: Incorporated the information model aspects of the security
  specification.

3.3 General Conventions
- UML diagrams are used as a way to concisely describe concepts. They are
  not intended to convey any specific implementation or methodology
  requirements.
- Interfaces are often used in UML diagrams. They are used instead of classes
  with attributes to provide an abstract definition without implying any specific
  implementation. Specifically they do not imply that objects in the registry will
  be access directly via these interfaces. Objects in the registry are accessed
  via registry services interface described in [5].
- The term “managed object content” is used to refer to actual registry content
  (e.g. a DTD, as opposed to metadata)
The term “ManagedObject” is used to refer to an object that provides metadata about content instance (managed object content).

The information model does not contain any elements that are the actual content of the Registry (managed object content). All elements 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].

### 3.4 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

### 3.5 Related Documents

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

- **a)** *ebXML Registry Business Domain Model* [4] - defines requirements for ebXML Registry Services
- **b)** *ebXML Registry Services Specification* [5] - defines the actual registry services based on this information model
- **c)** *Trading-Partner Specification* [8] (under development) - defines how profiles can be defined for a party and how two parties’ profiles may be used to define a party agreement
- **e)** *Core Components specification* [7] (??? called Mary Kay to get name)

### 4 Design Objectives

#### 4.1 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 [2] and the ISO 11179 [3] Registry models
- Align with relevant works in progress within other ebXML working groups
4.2 Caveats and Assumptions

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

5 System Overview

5.1 Role of ebXML Registry

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

5.2 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 [5]. This document does not provide details on these services but may occasionally refer to them.

5.3 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 classes.

The Registry information model:

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

5.4 How the Registry Information Model Works

Implementers of the ebXML Registry may use the information model to determine which classes to include in their registry implementation and what attributes and methods these classes may have. They may also use it to determine what sort of database schema their registry implementation may need.
5.5 Where the Registry Information Model May Be Implemented

The Registry Information Model may be implemented within an ebXML registry in the form of a relational database schema, object database schema or some other physical schema. It may also be implemented as interfaces and classes within a registry implementation.

6 Registry Information Model: Public View

This chapter provides a high level public view of the most visible objects in the registry.

Figure 1 shows the public view of the objects in the Registry and their relationships as a UML class diagram. It does not show inheritance, class attributes or class methods.

The reader is again reminded that the information model is modeling metadata and not actual content.

Figure 1: Information Model Public View
6.1 ManagedObject

The central object in the information model is a ManagedObject. An instance of ManagedObject exists for each content instance submitted to registry. Instances of ManagedObject class provide metadata about a managed object content in the registry. The actual managed object content (e.g. a DTD) is not contained in an instance of the ManagedObject class.

6.2 Association

Association instances are used to define many-to-many associations between objects in the information model. Associations are described in detail in chapter 22.

6.3 ExternalLink

ExternalLink instances model a named URI to content that may reside outside the registry. ManagedObject may be associated with any number of ExternalLinks to annotate a ManagedObject with external links to external content.

Consider the case where a Submitting Organization submits a managed object content (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 ManagedObject. The user may click on such links and navigate to an external web page referenced by the link.

6.4 ClassificationNode

ClassificationNode instances are used to define tree structures where each node in the tree is a ClassificationNode. Classification trees constructed with ClassificationNodes are used to define classification schemes or ontologies. ClassificationNode is described in detail in chapter 25.

6.5 Classification

Classification instances are used to classify managed object content by associating their ManagedObject instance with a ClassificationNode within a classification scheme. Classification is described in detail in chapter 25.

6.6 Package

Logically related ManagedObjects may be grouped into a Package. It is anticipated that Registry Services will allow operations to be performed on an entire package of objects in the future.

6.7 AuditableEvent

AuditableEvent is used to provide an audit trail for ManagedObjects.
6.8 PostalAddress
PostalAddress is a simple reusable entity class that defines attributes of a postal Address.

6.9 Contact
Contact is a simple reusable entity class that defines attributes of a contact person.

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

7 Registry Information Model: Detail View
This chapter covers the information model classes in more detail than the Public View. The detail view introduces some additional classes within the model that were not described in the public view of the information model.

Figure 4 shows the inheritance or “is a” relationships between the classes in the information model. Note that it does not show the relationships since they have already been shown in Figure 1. Class attributes and class methods are also not shown in Figure 2. Detailed description of methods and attributes of most interfaces and classes will be displayed in tabular form following the description of each class in the model.

The interface Association will be covered in detail separately in chapter 10. The interfaces Classification and ClassificationNode will be covered in detail separately in chapter 11.

The reader is again reminded that the information model is modeling metadata and not actual content.
7.1 Interface Object

All Known Subinterfaces:
Association, Classification, ClassificationNode, ExternalLink,
ExtrinsicObject, IntrinsicObject, ManagedObject, Organization, Package,
Submission
Object provides a common base interface for almost all objects in the information model. Information model classes whose instances have a unique identity and an independent life cycle are descendents of the Object class.

Note that Contact and Address are not descendents of the Object class because their instances do not have an independent existence and unique identity. They are always a part of some other class's instance (e.g. Organization has an Address).

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getDescription()</td>
<td>Gets the context independent textual description for this object.</td>
</tr>
<tr>
<td>String getGUID()</td>
<td>Gets the globally unique ID for this object.</td>
</tr>
<tr>
<td>void setDescription(String description)</td>
<td>Sets the context independent textual description for this object.</td>
</tr>
<tr>
<td>void setGUID(String GUID)</td>
<td>Sets the globally unique ID for this object.</td>
</tr>
</tbody>
</table>

### 7.2 Interface `Versionable`

**All Known Subinterfaces:**

- Association, Classification, ClassificationNode, ExternalLink
- ExtrinsicObject, IntrinsicObject, ManagedObject, Organization, Package

The Versionable interface defines the behavior common to classes that are capable of creating versions of their instances. At present all ManagedObject classes are required to implement the Versionable interface.

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getMajorVersion()</td>
<td>Gets the major revision number for this version of the Versionable object.</td>
</tr>
<tr>
<td>int getMinorVersion()</td>
<td>Gets the minor revision number for this version of the Versionable object.</td>
</tr>
</tbody>
</table>
7.3 Interface ManagedObject

All Superinterfaces:
Object, Versionable

All Known Subinterfaces:
Association, Classification, ClassificationNode, ExternalLink, ExtrinsicObject, IntrinsicObject, Organization, Package

The ManagedObject class models a common base class for all submitted content whose life cycle is managed by the registry. Content submitted to the registry is further specialized by ExtrinsicObject and IntrinsicObject sub-classes of ManagedObject.

Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>getAssociations()</td>
</tr>
<tr>
<td>Collection</td>
<td>getAuditTrail()</td>
</tr>
<tr>
<td>Collection</td>
<td>getClassifications()</td>
</tr>
<tr>
<td>Collection</td>
<td>getExternalLinks()</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
</tr>
<tr>
<td>Package</td>
<td>getPackage()</td>
</tr>
<tr>
<td>int</td>
<td>getStatus()</td>
</tr>
<tr>
<td>void</td>
<td>setName(String name)</td>
</tr>
</tbody>
</table>

void setMajorVersion(int majorVersion)

Getts 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.
repository.

```java
void setStatus(int status)
```

Sets the life cycle status of the ManagedObject within the registry.

Methods inherited from interface Object

- `getDescription`, `getGUID`, `setDescription`, `setGUID`

Methods inherited from interface Versionable


### 7.4 Interface ExtrinsicObject

ExtrinsicObject models all submitted content whose type is not intrinsically known to the registry.

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 modeled by ExtrinsicObject include party profiles, business process descriptions, schemas, etc.

#### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String getMimeType()</code></td>
<td>Gets the mime type associated with the ManagedObject.</td>
</tr>
<tr>
<td><code>int getObjectType()</code></td>
<td>Gets the pre-defined object type that best describes the ManagedObject.</td>
</tr>
<tr>
<td><code>URI getURI()</code></td>
<td>Gets the URI to the actual object in repository</td>
</tr>
<tr>
<td><code>boolean isOpaque()</code></td>
<td>Determines whether the ExtrinsicObject is opaque (not readable) by the registry. In some situations, a Submitting Organization may submit content that is encrypted and not even readable by the registry. This attribute allows the registry to know whether this is the case.</td>
</tr>
</tbody>
</table>
void `setMimeType` (String mimeType)
Sets the mime type associated with the ManagedObject.

void `setObjectType` (int type)
Sets the pre-defined object type that best describes the ManagedObject.

void `setOpaque` (boolean isOpaque)
Sets whether the ExtrinsicObject is opaque (not readable) by the registry.

void `setURI` (URI uri)
Sets the URI to the actual object in repository.

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

### 7.4.1 Pre-Defined Extrinsic Object Types

The following table lists pre-defined types of ExtrinsicObjects.

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int <code>OBJECT_TYPE_PARTY_AGREEMENT</code></td>
</tr>
<tr>
<td>These are XML documents that represent a technical agreement between two parties on how they plan to communicate with each other in a specific B2B protocol.</td>
</tr>
</tbody>
</table>

| static int `OBJECT_TYPE_PARTY_PROFILE` |
| These are XML documents that provide information about a party interested in participating in B2B interaction. |

| static int `OBJECT_TYPE_PROCESS` |
| A process description document. |

| static int `OBJECT_TYPE_ROLE` |
| An XML description of a Role in a Collaboration Party Profile |

| static int `OBJECT_TYPE_SERVICE_INTERFACE` |
| An XML description of a service interface |

| static int `OBJECT_TYPE_SOFTWARE_COMPONENT` |
| A software component (e.g. EJB, class library etc.) |

| static int `OBJECT_TYPE_TRANSPORT` |
| An XML description of a transport configuration Collaboration Party Profile |

| static int `OBJECT_TYPE_UML_MODEL` |
| A UML Model |

| static int `OBJECT_TYPE_UNKNOWN` |
| An unspecified objectType. |

| static int `OBJECT_TYPE_XML_SCHEMA` |
A schema (DTD, XML Schema, Relax, etc.) for an XML document

### 7.5 Interface **IntrinsicObject**

**IntrinsicObject**

**All Superinterfaces:**
- ManagedObject, Object, Versionable

**All Known Subinterfaces:**
- Association, Classification, ClassificationNode, ExternalLink, Organization, Package

IntrinsicObject provides a common base class for modeling all submitted content whose type is known to the Registry. In fact, these types are defined by the ebXML registry 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.

### 7.6 Interface **ExternalLink**

**ExternalLink**

**All Superinterfaces:**
- IntrinsicObject, ManagedObject, Object, Versionable

ExternalLink instances model a named URI to content that may reside outside the registry.

Consider the case where a Submitting Organization submits a managed object content (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.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI getURI()</td>
<td>Gets URI to the external content.</td>
</tr>
<tr>
<td>void setURI(URI uri)</td>
<td>Sets URI to the external content.</td>
</tr>
</tbody>
</table>

Note that methods inherited from the base interfaces of this interface are not shown.
8 Registry Audit Trail

This chapter describes the information model elements that support the audit trail capability of the registry. There are several classes in this chapter that are entity classes that are used as wrappers to model a set of related attributes. These entity classes do not have any associated behavior. They are analogous to the “struct” construct in the C programming language.

The getAuditTrail() method of a ManagedObject returns an ordered Collection of AuditableEvents. These AuditableEvent constitute the audit trail for the ManagedObject. AuditableEvents include a timestamp for the event. Each AuditableEvent has an AuditableIdentity that identifies the specific user that performed an action that resulted in an AuditableEvent. Each AuditableIdentity has an Organization which is usually the submitting Organization.

8.1 Interface AuditableEvent

All Superinterfaces:

Object

AuditableEvent instances provide a long-term record of events that effect a change of state in a ManagedObject. A ManagedObject is associated with an ordered Collection of AuditableEvent instances that provide a complete audit trail for that Object.

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 ManagedObject. For example a client request could Create, Update, Deprecate or Delete a ManagedObject. No AuditableEvent is created for requests that do not alter the state of a ManagedObject. Specifically, read-only requests do not generate an AuditableEvent. No AuditableEvent is generated for a ManagedObject when it is classified, assigned to a Package or associated with another Object.

See Also:

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int EVENT_TYPE_CREATED</td>
</tr>
<tr>
<td>An event that created a ManagedObject</td>
</tr>
<tr>
<td>static int EVENT_TYPE_DELETED</td>
</tr>
<tr>
<td>An event that deleted a ManagedObject</td>
</tr>
<tr>
<td>static int EVENT_TYPE_DEPRECATED</td>
</tr>
<tr>
<td>An event that deprecated a ManagedObject</td>
</tr>
</tbody>
</table>
static int EVENT_TYPE_UPDATED
   An event that updated the state of a ManagedObject
static int EVENT_TYPE_VERSIONED
   An event that versioned a ManagedObject

Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuditableIdentity</td>
<td>getAuditableIdentity()</td>
<td>Gets the AuditableIdentity that sent the request that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>effected this event.</td>
</tr>
<tr>
<td>int</td>
<td>getEventType()</td>
<td>The type of this event</td>
</tr>
<tr>
<td>Timestamp</td>
<td>getTimestamp()</td>
<td>Gets the Timestamp for when this event occurred.</td>
</tr>
</tbody>
</table>

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

8.2 Interface AuditableIdentity

All Superinterfaces:

   Object

AuditableIdentity instances are used in an AuditableEvents to keep track of the identity of the requestor that sent the request that generated the AuditableEvent.

Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>getName()</td>
<td>Get the name of this AuditableIdentity.</td>
</tr>
<tr>
<td>Organization</td>
<td>getOrganization()</td>
<td>Gets the Submitting Organization that sent the request that effected this change.</td>
</tr>
</tbody>
</table>

8.3 Interface Organization

All Superinterfaces:

   IntrinsicObject,ManagedObject,Object,Versionable

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

Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostalAddress</td>
<td>getAddress</td>
<td>Gets the Address for this Organization.</td>
</tr>
<tr>
<td>Contact</td>
<td>getContact</td>
<td>Gets the primary Contact for this Organization.</td>
</tr>
<tr>
<td>String</td>
<td>getFax</td>
<td>Gets the FAX number for this Organization.</td>
</tr>
<tr>
<td>Organization</td>
<td>getParent</td>
<td>Gets the parent Organization for this Organization.</td>
</tr>
<tr>
<td>String</td>
<td>getTelephone</td>
<td>Gets the main telephone number for this Organization.</td>
</tr>
</tbody>
</table>

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

8.4 Class Contact

Contact is a simple reusable entity class that defines attributes of a contact person.

Field Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostalAddress</td>
<td>address</td>
<td>The postal address for this Contact.</td>
</tr>
<tr>
<td>String</td>
<td>email</td>
<td>The email address for this Contact.</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>fax</td>
<td>The FAX number for this Contact.</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>mobilePhone</td>
<td>The mobile telephone number for this Contact.</td>
</tr>
<tr>
<td>PersonName</td>
<td>name</td>
<td>Name of contact person</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>pager</td>
<td>The pager telephone number for this Contact.</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>telephone</td>
<td>The default (land line) telephone number for this Contact.</td>
</tr>
</tbody>
</table>
8.5 Class PostalAddress

Address is a simple reusable entity class that defines attributes of a postal Address.

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>city</td>
<td>String</td>
<td>The city</td>
</tr>
<tr>
<td>country</td>
<td>String</td>
<td>The country</td>
</tr>
<tr>
<td>postalCode</td>
<td>String</td>
<td>The postal or zip code</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>The state</td>
</tr>
<tr>
<td>street</td>
<td>String</td>
<td>The street</td>
</tr>
</tbody>
</table>

8.6 Class TelephoneNumber

A simple reusable entity class that defines attributes of a telephone number.

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>areaCode</td>
<td>String</td>
<td>Area code</td>
</tr>
<tr>
<td>countryCode</td>
<td>String</td>
<td>country code</td>
</tr>
<tr>
<td>extension</td>
<td>String</td>
<td>internal extension if any</td>
</tr>
<tr>
<td>number</td>
<td>String</td>
<td>The telephone number suffix not including the country or area code.</td>
</tr>
<tr>
<td>url</td>
<td>String</td>
<td>A URL that can dial this number electronically</td>
</tr>
</tbody>
</table>
8.7 Class PersonName

A simple entity class for a person’s name.

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>firstName</td>
<td>The first name for this Contact.</td>
</tr>
<tr>
<td>String</td>
<td>lastName</td>
<td>The last name (surname) for this Contact.</td>
</tr>
<tr>
<td>String</td>
<td>middleName</td>
<td>The middle name for this Contact.</td>
</tr>
</tbody>
</table>

9 Managed Object Naming

A managed object content has a name that may or may not be unique within the Registry.

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

10 Association of Managed Objects

A managed object content may be associated with 0 or more objects. The information model defines an Association class. An instance of the Association class represents an association between a ManagedObject and another Object. An example of such an association is between a PartyProfile and a PartyAgreement associated with that PartyProfile as shown in Figure 5.
10.1 Interface Association

All Superinterfaces:
- IntrinsicObject, ManagedObject, Object, Versionable

All Known Subinterfaces:
- Classification

Association instances are used to define many-to-many associations between objects in the information model.

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

### Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>ASSOCIATION_TYPE_CLASSIFIED_BY</th>
<th>Defines that the source object is classified by the target object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_CONTAINED_BY</td>
<td>Defines that source object is contained by the target object.</td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_CONTAINS</td>
<td>Defines that source object contains the target object.</td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_EXTENDS</td>
<td>Defines that source object inherits from or specializes the target object.</td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE.Implements</td>
<td>Defines that source object implements the behaviour defined by the target object.</td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_INSTANCE_OF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that source object is an instance of target object</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_RELATED_TO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that source object is an instance of target object.</td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_SUPERSEDED_BY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that the source object is superseded by the target object.</td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_SUPERSEDES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that the source object supersedes the target object.</td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_USED_BY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that the source object is used by the target object in some manner.</td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td>ASSOCIATION_TYPE_USES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines that the source object uses the target object in some manner.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th></th>
<th>Defines that source object is an instance of target object</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>getAssociationType()</td>
</tr>
<tr>
<td></td>
<td>Gets the predefined association type for this Association.</td>
</tr>
<tr>
<td>Object</td>
<td>getSourceObject()</td>
</tr>
<tr>
<td></td>
<td>Gets the Object that is the source of this Association.</td>
</tr>
<tr>
<td>String</td>
<td>getSourceRole()</td>
</tr>
<tr>
<td></td>
<td>Gets the name of the role played by the source Object in this Association.</td>
</tr>
<tr>
<td>Object</td>
<td>getTargetObject()</td>
</tr>
<tr>
<td></td>
<td>Gets the Object that is the target of this Association.</td>
</tr>
<tr>
<td>String</td>
<td>getTargetRole()</td>
</tr>
<tr>
<td></td>
<td>Gets the name of the role played by the target Object in this Association.</td>
</tr>
<tr>
<td>boolean</td>
<td>isBidirectional()</td>
</tr>
<tr>
<td></td>
<td>Determine whether this Association is bi-directional.</td>
</tr>
<tr>
<td>void</td>
<td>setAssociationType(int associationType)</td>
</tr>
<tr>
<td></td>
<td>Sets the predefined association type for this Association.</td>
</tr>
<tr>
<td>void</td>
<td>setBidirectional(boolean bidirectional)</td>
</tr>
<tr>
<td></td>
<td>Set whether this Association is bi-directional.</td>
</tr>
<tr>
<td>void</td>
<td>setSourceRole(String sourceRole)</td>
</tr>
<tr>
<td></td>
<td>Sets the name of the role played by the source Object in this Association.</td>
</tr>
<tr>
<td>void</td>
<td>setTargetRole(String targetRole)</td>
</tr>
<tr>
<td></td>
<td>Sets the name of the role played by the destination Object in this Association.</td>
</tr>
</tbody>
</table>
11 Classification of Managed Objects

This section describes the how the information model supports classification of managed object contents. It is a simplified version of the OASIS classification model [2].

A managed object content may be classified in many ways. For example the same Party managed object content may be classified by the industry it is in, by the products it sells and by the geographical location it is in.

A general classification scheme can be viewed as a classification tree. In the example shown below in Figure 4, ManagedObjects representing PartyProfiles are shown as shaded boxes. Each PartyProfile represents an automobile manufacturer. Each PartyProfile is classified by the ClassificationNode named Automotive under the root ClassificationNode named Industry. Furthermore, the US Automobile manufacturers are classified by the US ClassificationNode under the Geography ClassificationNode. Similarly, a European automobile manufacturer is classified by the Europe ClassificationNode under the Geography ClassificationNode.

The example shows how a ManagedObject may be classified by multiple classification schemes. A classification scheme is defined by a ClassificationNode that is the root of a classification tree (e.g. Industry, Geography).

---

1 Superset of OASIS. Some examples are borrowed from OASIS.
Figure 6: Example showing a Classification Tree

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

In order to support a general classification scheme that can support single level as well as multi-level classifications, the information model defines the classes and relationships shown in Figure 7.
A Classification is a specialized form of an Association. Figure 8 shows an example of a Party object that is classified by the Industry that it belongs to.

Figure 8: Classification Instance Diagram

11.1 Interface `ClassificationNode` ClassificationNode

**All Superinterfaces:**
- `IntrinsicObject`, `ManagedObject`, `Object`, `Versionable`

ClassificationNode instances are used to define tree structures where each node in the tree is a ClassificationNode. Such classification trees constructed with ClassificationNodes are used to define classification schemes or ontologies.

**See Also:**
- `Classification`

**Method Summary**
### ClassificationNode

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getParent()</code></td>
<td>Gets the parent ClassificationNode for this ClassificationNode.</td>
</tr>
<tr>
<td><code>setParent(ClassificationNode parent)</code></td>
<td>Sets the parent ClassificationNode for this ClassificationNode.</td>
</tr>
</tbody>
</table>

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

In Figure 6, several instances of ClassificationNode are defined (all light colored boxes). A ClassificationNode has zero or 1 ClassificationNode for its parent and zero or more ClassificationNodes for its immediate children. If a ClassificationNode has no parent then it is the root of a classification tree. Note that the entire classification tree is recursively defined by a single information model element ClassificationNode.

### 11.2 Interface **Classification**

**All Superinterfaces:**

- Association, IntrinsicObject, ManagedObject, Object, Versionable

Classification instances are used to classify managed object content by associating their ManagedObject instance with a ClassificationNode instance within a classification scheme.

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

Classification is a specialized form of Association from a ManagedObject to a specific ClassificationNode in the classification tree. The information model defines a Classification class as a sub-class of Association class to allow for future specialization as well as to make classification notion be obvious in the model. This also allows for more efficient implementation by providing a different extent for Classifications than from Associations.

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

### 11.2.1 Context Sensitive Classification

Consider the case depicted in Figure 9 where a PartyProfile 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 PartyProfile 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

A more complex case is where a Classification may be associated with multiple contexts. In order to support the possibility of Classification within multiple contexts, a Classification is itself classified by another Classification that binds the first Classification to a Classification node that provides the missing context.

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

- A ManagedObject to be classified by defining a Classification that associates it with a ClassificationNode in a classification tree
- A ManagedObject to be classified along multiple facets by having multiple classifications that associate it with multiple ClassificationNodes.
- A classification defined for a ManagedObject to be qualified by the context in which it is being classified
11.3 Example of Classification Schemes [9]

The following table lists some examples of possible classification schemes enabled by the information model. These schemes are based on a subset of contextual concepts identified by BP and CC in [9]. This list is meant to be illustrative not prescriptive.

<table>
<thead>
<tr>
<th>Classification Scheme (Context)</th>
<th>Usage Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industry</td>
<td>Find all Parties in Automotive industry</td>
</tr>
<tr>
<td>Process</td>
<td>Find a ServiceInterface that implements a Process</td>
</tr>
<tr>
<td>Product</td>
<td>Find a business that sells a product</td>
</tr>
<tr>
<td>Locale</td>
<td>Find a Supplier located in a region</td>
</tr>
<tr>
<td>Temporal</td>
<td>Find Supplier that can ship with 24 hours</td>
</tr>
<tr>
<td>Role</td>
<td>Find All Suppliers</td>
</tr>
<tr>
<td>Package</td>
<td>Find all DTDs in the RosettaNet DTD package</td>
</tr>
</tbody>
</table>

Table 1: Pre-defined Classification Schemes

12 Querying of Managed Objects

This chapter describes how the information model supports the querying of managed object contents based on the attributes, content, associations and classifications of managed object contents. Details of the access protocol between clients and the Registry for the purpose of object querying are described in [5]. This chapter defines at a high level the query mechanism without defining the actual query protocol and messages exchanged as part of that protocol.

12.1 Object Query Use Cases

It is recognized that there are several different use cases defining how a client may want to query and search the Registry for managed object contents.

12.1.1 Browse and Drill Down Query

This is expected to be the primary use case for querying the Registry. In this scenario a user browses the registry content using a GUI tool referred to as the Registry Browser. The user expects to initially browse the content based on the pre-defined classification schemes defined in section 11.3. The user may also use additional classification schemes that may have been defined for objects selected by the pre-defined classification scheme chosen. The user will select a managed object content and drill down to view the details of the object. Such browse and drill down query support is defined in section 12.2.
12.1.2 Ad Hoc Queries Based on Object Metadata And Content

This is an advanced form of use case for querying the Registry. In this scenario a client program may search for managed object contents based on the metadata defined as attributes in its corresponding ManagedObject as well as the managed object content itself.

12.1.3 Keyword Search Query

In this scenario a user may search for managed object contents by specifying keywords that may be used to identify the managed object contents.

12.2 Classification Based Query Support

CLASSIFICATION BASED QUERY SUPPORT IS REQUIRED FOR AN EBXML REGISTRY IMPLEMENTATION. AN IMPLEMENTATION MUST SUPPORT THE FOLLOWING CAPABILITIES FOR CLASSIFICATION BASED QUERY SUPPORT. DETAILS OF THE ACTUAL PROTOCOL AND MESSAGES EXCHANGED ARE DESCRIBED IN [5].

1. Receive a query which requests a list of root ClassificationNodes (classification schemes) defined and return the ClassificationNodes that have no parent ClassificationNode.

2. Receive a query in which a ClassificationNode is identified and return the classification tree under that node in the tree.

3. Receive a query in which a ClassificationNode is identified and return the managed object contents classified by that ClassificationNode.

13 Information Model: Security View

This chapter describes the aspects of the information model that relate to the security features of the registry. It shows the view of the objects in the Registry from a security perspective. It is showing object relationships as a UML class diagram. It is not showing class attributes or class methods that will be described in subsequent sections.
13.1 Interface AccessControlPolicy

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

The Permission object is used for authorization and access control to Objects in the registry. The Permissions for an Object are defined in an AccessControlPolicy object.

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

See Also:
- Privilege
- AccessControlPolicy

Method Summary

<table>
<thead>
<tr>
<th>Collection</th>
<th>getPermissions()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Permissions defined for this AccessControlPolicy</td>
</tr>
</tbody>
</table>

13.3 Interface Privilege

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

A requesting Principal must have all of the PrivilegeAttributes specified in a Privilege in order to gain access to a method in a protected Object. Permissions defined in the Object'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, Groups or a SecurityClearences.

See Also:
- PrivilegeAttribute
- Permission

Method Summary

<table>
<thead>
<tr>
<th>Collection</th>
<th>getPrivilegeAttributes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Privileges associated with this Permission</td>
</tr>
</tbody>
</table>
13.4 Interface PrivilegeAttribute

All Known Subinterfaces:

Group, Identity, Role, SecurityClearance

PrivilegeAttribute is a common base class 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

13.5 Interface SecurityClearance

All Superinterfaces:

PrivilegeAttribute

A SecurityClearance is PrivilegeAttribute that describes a security clearance.

13.6 Interface Role

All Superinterfaces:

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 may be allowed to write a prescription but a Nurse may not.

13.7 Interface Group

All Superinterfaces:

PrivilegeAttribute

A security Group PrivilegeAttribute. A Group is an aggregation of users that may have different roles. For example a hospital may have a Group defined for Nurses and Doctors that are participating in a specific clinical trial (e.g AspirinTrial). 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 Nurses as a rule may not be allowed to write prescriptions).

13.8 Interface Identity
PrivilegeAttribute

A security Identity PrivilegeAttribute. This is typically used to identify a person. Identity attribute may be in form of a digital certificate.

13.9 Interface 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, which has a set of PrivilegeAttributes. These PrivilegeAttributes include at least one identity, and optionally a set of role 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

<table>
<thead>
<tr>
<th>Collection</th>
<th>getGroups()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the Groups associated with this Principal.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Collection</th>
<th>getIdentities()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the Identities associated with this Principal.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Collection</th>
<th>getRoles()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the Roles associated with this Principal.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Collection</th>
<th>getSecurityClearances()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the SecurityClearances associated with this Principal.</td>
<td></td>
</tr>
</tbody>
</table>
References

[1] ebXML Glossary,

[2] OASIS Information Model

http://www.nist.gov/itl/div897/ctg/regrep/oasis-work.html

[3] ISO 11179 Information Model


http://www.ebxml.org/specdrafts/RegRepv1-0.pdf


http://www.ebxml.org/specdrafts/Busv2-0.pdf

[7] ebXML Core Components Metamodel (unable to get these so far)

[8] Trading-Partner Specification

http://www.ebxml.org/project_teams/trade_partner/private/

[9] Context table informal document from Core Components

http://www.ebxml.org/project_teams/core_components/ContextTable.doc

14 Disclaimer

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.
### 15 Contact Information

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Company</th>
<th>Address</th>
<th>City, State, Postal Code</th>
<th>Country</th>
<th>Phone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Leader</td>
<td>Scott Nieman</td>
<td>Norstan Consulting</td>
<td>5101 Shady Oak Road</td>
<td>Minnetonka, MN 55343</td>
<td>USA</td>
<td>952.352.5889</td>
<td>Scott.Nieman@Norstan</td>
</tr>
<tr>
<td>Vice Team Lead</td>
<td>Yutaka Yoshida</td>
<td>Sun Microsystems</td>
<td>901 San Antonio Road, MS UMPK17-102</td>
<td>Palo Alto, CA 94303</td>
<td>USA</td>
<td>650.786.5488</td>
<td><a href="mailto:Yutaka.Yoshida@eng.sun.com">Yutaka.Yoshida@eng.sun.com</a></td>
</tr>
<tr>
<td>Editor</td>
<td>Farrukh S. Najmi</td>
<td>Sun Microsystems</td>
<td>1 Network Dr., MS BUR02-302</td>
<td>Burlington, MA, 01803-0902</td>
<td>USA</td>
<td>781.442.0703</td>
<td><a href="mailto:najmi@east.sun.com">najmi@east.sun.com</a></td>
</tr>
</tbody>
</table>
Copyright Statement

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.