OASIS Registry/Repository Technical Specification

Working Draft 1.1 December 20, 2000

Abstract

This specification represents the collective efforts of the Registry and Repository Technical Committee of OASIS, the Organization for the Advancement of Structured Information Standards. It specifies a registry/repository information model and a registry services interface to a collection of registered objects, including but not limited to XML documents and schemas. The information model uses UML diagrams and written semantic rules to specify logical structures that serve as the basis of definition for an XML-based registry services interface. The information model is used for definitional purposes only; conformance to this specification depends solely on correct implementation of some designated subset of the registry services interface.

The registry services interface consists of request services to create new registry information or to modify or supplement existing registry entries. It also consists of query and retrieval services to search registry content and retrieve selected registry information, or to retrieve registered objects via object references or locators. The registry services interface supports browsing by arbitrary electronic agents as well as interoperation among conforming implementations. This document deals primarily with the registry, although some scenarios and requirements for the repository are included.

This document is a draft proposal under development by the Oasis Registry/Repository Technical Committee. Its purpose is to solicit additional input and to convey the current state of the Oasis Registry/Repository Information Model and Technical Specification.

Status of this Document

This document represents a work in progress upon which no reliance should be made. Its temporary accessibility, until more permanent accessibility is established at the OASIS web site, is via the following URL:

ftp://xsun.sdct.itl.nist.gov/regrep/OasisRegrepSpec.pdf

Comments are welcome and should be submitted to:

Lisa Carnahan
Chair, Oasis Registry and Repository
lisa.carnahan@nist.gov

Information on joining the OASIS Registry and Repository Technical Committee can be found at URL http://www.oasis-open.org

Document Version History

- Version 0.1, June 1999. Initial version, Terry Allen, Editor.

- Version 0.1.5, July 1999. Editor added material on RFC 2169 and RFC 2483, with a suggestion that we use RDF. Added reference to Editor's essay in taxonomy. Added "User Preferences Among Registries and Repositories" section. Added section on typology s.v. "Interface", warning not to link to interface. Added section on usage of registry documents.

- Version 0.2, August 1999. Editor added references to IETF WEBDAV and DASL I-D and much content from face-to-face meeting. Later added remark about XML Schema in a repository.

- Version 0.3, October 1999. Editor updated, reduced scope, included references to DTDs and sample documents.

- Version 0.4, November 1999. Editor clarified what is normative and nonnormative in submission section.

- Version 0.9, April 2000. Based on member comments, Editor revised sections, moved some functional requirements into this document, folded in request by identifier, and added design principles.

- Version 0.95, July 2000. NIST assumes temporary editorship of document. Many changes resulted from contributions and discussions at the June 15-16, 2000 Oasis Reg/Rep meeting in Paris, France. The objective and principles were changed, the design principles and scenarios were revised, the Registry section was replaced by an Information Model, and the Submission Semantics section was replaced by XML definitions and protocols based on the Information Model.

- Version 1.0, November 2000. Changes based on discussions at Oasis Reg/Rep face-to-face meeting held October 18-19, 2000 at Sun Microsystems, Inc. offices in Menlo Park, CA, USA. Decision to base the Information Model on UML notation rather than Entity-Attribute-Relationship notation.

- Version 1.1, December 2000. Modifications based on discussions at the Oasis Reg/Rep face-to-face meeting held December 5, 2000, in association with the XML 2000 conference in Washington, DC, and a follow-on teleconference held December 15, 2000. New Sections "Registry Query" and "Registry Filters"added to the document as result of discussions, but without a formal vote. Recursive enhancements to "GetRegisteredObject" added to document as result of discussions, but without a formal vote.

Contributors

The chairman of the OASIS Registry and Repository Technical Committee is Lisa Carnahan of the U.S. National Institute of Standards and Technology (NIST). Norbert Mikula of DataChannel is the OASIS Chief Technical Officer.

Voting members of the Technical Committee, at the time of this version, are:

Nagwa Abdelghfour (Sun Microsystems),
Lisa Carnahan (U.S. NIST),
Dan Chang (IBM),
Robin Cover (ISOGEN),
Úna Kearns (Documentum),
Megan MacMillan (Gartner Solista),
Norbert Mikula (DataChannel),
Yutaka Yoshida (Sun Microsystems), and
Jaime Walker (Boeing).

Other individuals who have contributed to the development of the specification include:

Terry Allen (Commerce One),
Murray Altheim (Sun Microsystems),
Bryan Caporlette (Sequoia Software),
Ron Daniel (Metacode),
Len Gallagher, (U.S. NIST),
Eduardo Gutentag (Sun Microsystems),
Michael Mealling (Network Solutions),
Ron Schuldt (UDEF),
Priscilla Walmsley (XMLSolutions). and
Norm Walsh (Arbortext)

Notational Conventions

This specification uses Unified Modeling Language (UML) and written semantic rules to specify the OASIS Registry/Repository Information model. It uses XML 1.0 and additional written semantic rules to specify the effect of SubmitRequest DTD's on registry content and to define the syntax and results of GetRegisterdObject, GetRegistryEntry, and RegistryQuery DTD's.

Some options for RegistryQuery are specified using syntax and semantics from ISO/IEC 9075 - Database Language SQL, W3C XML Query, or ODMG Object Query Language (OQL).

Editor's Notes

1. The Introduction sections were copied directly from Introduction sections of earlier versions of the OASIS specification. These sections were written months ago, and no attempt was made to update the wording for this version. Some wording in these paragraphs is inconsistent with decisions made at subsequent meetings, especially regarding Subsection 1.5, entitled "In Scope But Not Specified". It was copied directly from Section 5 of version 0.9.

2. Section 2 originated at the Oasis Reg/Rep face-to-face meeting held October 18-19, 2000 at Sun Microsystems, Inc. offices in Menlo Park, CA, USA. Slight modifications were made at the December 5, 2000 face-to-face meeting in Washington, DC.

3. Sections 3 and 4 have changed only minimally from version 1.0 to 1.1, based on the December 5, 2000, face-to-face in Washington, DC.

4. Section 5.1 "ObjectType" and Section 5.2 "FileType" were modified based on discussions at the December 5 face-to-face. Some Enumeration Domains in Section 5 are still very unstable, especially 5.2 "FileType", 5.6 "PropertyRights", 5.8 "NameContext", 5.11 "RelatedRole", and 5.12 "RoleCategory".

5. Section 6 changed only minimally from version 1.0- to 1.1, but there were some typo corrections that impact the previously distributed XML ELEMENT and ENTITY definitions.

6. Sections 7.1 and 7.2 have changed considerably from version 1.0 to version 1.1. GetRegisteredObject and GetRegistryEntry now take a RegistryEntryQuery as input rather than the previous single AssignedURN. The recursion rules for GetRegisteredObject, previously unfinished, have been filled in. Filtering options have been added to GeetRegistryEntry.

7. Section 7.4 is completely new in version 1.1 as the result of discussions during the December 15 teleconference. It also spawned a completely new Section of the document, "Register Filters", now labeled as Section 9. The previous Section 9, "Conformance" is now Section 10. No vote was taken to formally approve the content of Section 7.4 or Section 9. The chair will conduct an email ballot over the next few weeks.

8. The remainder of Section 7, including the DTD's for "SubmitRequest", "ClassificationScheme", and "RegistryPackage", have not changed from version 1.0 to version 1.1.

9. Section 8, "Request Elements", has not changed from version 1.0 to version 1.1, but it still has not been explicitly voted upon. It will also be the subject of follow-on email balloting.

10. Subsection 8.24, "Request by unique identifier", was copied directly from version 0.9 with no attempt to update any of the wording. There is a very close relationship between these requests and the GetRegisteredObject and GetRegistryEntry services presented in Sections 7.1 and 7.2. The syntax of Section 8.24, if desirable, could be mapped to the semantics of those two services.

11. Section 9, "Registry Filters", is completely new, based on preliminary discussions during the December 15 teleconference. It is VERY TENTATIVE and should only be considered as a placeholder for XML syntax for logical predicates over each of the classes in the Registry/Repository Information Model. It has not been acted upon yet by Oasis TC membership.

12. Section 10, "Conformance", is the former Section 9, renumbered because of the addition of "Registry Filters". It has been in the document through two versions, but has not yet been formally voted upon by the TC membership. It will be evaluated along with the other new sections in an upcoming TC email ballot. The Conformance options could be refined based on the new facilties added in Sections 7.1, 7.2, and 7.4. The subsection on "Normative Policy Requirements" at the end of the Conformance section was copied directly from Section 9 of version 0.9 from Summer 2000, with no attempt to update the wording to reflect recent decisions.

13. Section 11, Terminology and Relevant Specs, was copied directly from an earlier version of the specification with no attempt to do any updates of the wording. It should be re-written to properly reference other specifications and point to external definitions.

14. Annex 1, "SQL Representations" is completely new. It was added as support for the RegistryEntrySQL query added as an option for RegistryEntryQuery in Section 7.4.1. The intent is that SQL, OQL, and XML Query support "always" be optional as for as this specification is concerned. However, the definitions are added in an Annex so that implementations wishing to support such queries will do so using the same syntax.

15. There was agreement at the October face-to-face meeting to add wording to the document in a couple of other areas, but precise text has not yet been proposed. Topics include consideration of privileges, roles, authentication, distinctions between searching and browsing, and authentication tokens ala UDDI.

Table of Contents

1. Introduction

As XML comes into use on the Web, DTDs, schemas, style sheets, and reuseable public text will be referred to by identifier, rather than being packaged with actual documents. It is critically necessary to be able to retrieve the referred-to entities, and in the Web context, it is preferrable to be able to do this automatically. And it is vital for users to be able to locate DTDs and schemas for the document types they want to create by consulting an interface to metadata about those DTDs and schemas.

1.1 Objectives and Deliverables

The objective of the OASIS Registry and Repository Technical Committee is to develop a specification for interoperable registries and repositories for SGML- and XML-related entities, including but not limited to DTDs and schemas, with an interface that enables searching on the contents of a repository of those entities, and to construct a prototype registry and repository. The registry and repository are to be designed to interoperate and cooperate with other registries and repositories compliant with this specification. The prototype is intended to serve as a model for an extensible and distributed network of registries and repositories; the specification is viewed as the primary deliverable.

1.2 Design Principles

The following design principles have been agreed to:

- The Registry Technical Specification shall employ existing standards and specifications where possible, avoiding specifications that are not stable.
- The normative part of the Registry Technical Specification shall be as small as reasonable.
- The normative part of the Registry Technical Specification shall be complete enough that registries and repositories conformant to it can interoperate in an extensible and distributed network.
- The normative part of the Registry Technical Specification shall be extensible; in particular, it shall be possible to extend the registration information schema or DTD without inhibiting interoperability among registries. (This point is called out because the registration information schema or DTD is likely to be a normative part of this specification).
- The Registry Technical Specification shall be vendor-neutral.
- The Registry Technical Specification shall use XML by preference for encoding of information and documents.
- The Registry Technical Specification shall assume the use of HTTP.
- The registry and repository shall be scaleable.

1.3 Expected Scenarios

These scenarios (or use cases, if you believe that use cases aren't scenarios) involve both users retrieving something from the repository and contributors registering something in the registry, which may involve depositing something in the repository. The OASIS Registry and Repository Technical Committee does not intend to specify solutions to all of thees scenarios, especially those that involve services built on top of registries. The OASIS Registry and Repository Technical Committee intends to add scenarios to this list only when they affect the data model proposed for the registry and repository.

1.3.1 Obtaining a DTD automatically

A user or user agent retrieves an XML-related entity such as a DTD automatically over the Web, as a result of some use of it in an XML context.

Motivation. Unless everything needed for parsing and displaying a document under all circumstances is packaged with the document itself, the document must refer to something (DTD, style sheet, public text) by identifier. It is necessary to be able to retrieve the referred-to entity, and in the Web context, it is preferrable to be able to do this automatically.

Example A. A user is sent a document the DOCTYPE declaration of which refers to a DTD by unique identifier (URN, PI, or FPI). His parser tells him it can't find the DTD, so he goes out and retrieves it manually from a repository (he doesn't need the registry interface because he has a unique ID but he does need to know where to find the repository).

Example B. A user clicks on a link to the stockmarket news and his browser receives an XML document the DOCTYPE declaration of which refers to a DTD by unique identifier; his browser, which has no copy locally, retrieves it automatically from the repository.

1.3.2 Depositing an XML-related entity

A creator of a resource deposits it, possibly along with related data, for service to the public, at some range of accessibility from archival (retrieval rate could be slow) to utility (retrieval rate must be fast, large number of connections must be supported, round-the-clock uptime with failover, etc.).

Motivation. Many creators of resources lack the facilities to serve them reliably; even those that can do so may not wish to deal with the burden.

Example A. An IETF working group decides that a DTD that is part of their specification, but which the IETF has no facilities to serve, must be available from a public Web server with high bandwidth, and doesn't want to have to maintain the server. It sends the DTD to a registry and repository services, which serves it, as in the first scenario.

Example B. A nonprofit publisher wishes its resources to be available for inspection and display. It deposits the resources in a repository and provides appropriate metadata for the repository's registry. The owner of the repository undertakes to make them available (but not with a high guaranteed quality of service).

Example C. Rosetta Net, a (real life) consortium of hardware vendors and suppliers, develops UML models, DTDs, and sets of text values used in their content, all expected to be in heavy demand, the text values to change frequently. It deposits the UML models, DTDs, and the initial set of text values in a repository, contracts for a regular update schedule and the highest available quality of service, and the repository undertakes to serve them, update them as agreed, push updates to subscribers, and maintain high quality of service for retrieval requests. Rosetta Net doesn't need a registry interface for this purpose because everything is to happen automatically, but it provides appropriate registry metadata so that the DTDs can be browsed and searched.

Example D. The Air Transport Association, which maintains important DTDs but make them available only to its members, wishes to offload the work of supplying those DTDs. It deposits the DTDs in a repository, contracts for service as in Example C, and in addition arranges that the DTDs are listed in the registry interface but are available only when an appropriate credential is presented in connection with a request for them. (This is an application of access control.)

1.3.3 Registering a resource without deposit

The owner of a resource, or another repository, registers the resource in the registry, but does not deposit the resource itself.

Motivation. Registries can interoperate to increase useability, but the actual storage location of a resource alone must not restrict the content of a registry.

Example A. A special-purpose registry wishes to makes its content visible in another registry, while maintaining that content in its own repository. It submits appropriate registry documents to the registry, including a pointer to its repository, and agrees with the registry that it will supply timely update information and that the registry will update its records and interface in a timely manner.

1.3.4 Browsing or searching for a DTD

A user ready to compose an XML document searches for a DTD that covers the subject of the document.

Motivation. Every day in newsgroups and e-mail discussion lists such as comp.text.sgml, comp.text.xml, and xml-dev people ask whether there is a DTD for some subject area or functional purpose. The number of such queries will grow if XML is widely adopted. Somehow they have to be answered if wheel reinvention is to be minimized.

Example A. A user is about to write his resume, and wants to use XML. He goes to a registry and looks in a subject hierarchy (or taxonomy) to find a resume DTD (this is browsing, not searching). The subject hierarchy interface displays three appropriate listings, he chooses among them on the basis of their descriptions, downloads the DTD he chose from the repository, manually adds it to his SO catalog, and sets to work with vi and SP.

Example B. A user is about to write his resume, and wants to use XML. He goes to a registry and uses its search engine to find a resume DTD (this is searching, not browsing). The search interface returns three hits, he chooses among them on the basis of their descriptions, downloads from the repository the DTD he chose, and loads it into his XML writing tool. The interface also provides a time-to-live value, showing him how long he can expect his resume DTD to be served by the repository.

Example C. A homeowner is about to advertise his house for sale, and opens his verboprocessor. He says "take a memo: real estate for sale" and the verboprocessor automatically contacts a registry to find an appropriate XML DTD for the homeowner's jurisdiction. He dictates the text of his ad, and the verboprocessor sends it to all real estate listing services it can locate. (In this scenario the verboprocessor uses a registry to find something in a repository and queries the repository with more than one query parameter.)

Example D. An XML application designer needs a component to represent the list of names of French provinces, so he consults a registry. The registry interface indicates that the list is available as a tab-delimited list in ASCII, as an XML schema datatype declaration, and as a parameter entity declaration in DTD syntax. He chooses the parameter entity declaration format by clicking something in the interface, and the repository returns it.

1.4 Basic functional requirements

On the basis of the registry and repository scenarios, the following functional requirements have been identified:

- **Registration**. A registry must support registration of the contents of the repository (and potentially other repositories) using standardized administrative metadata.

- **Classification**. Metadata must support application of both controlled vocabulary (for taxonomic view) and uncontrolled vocabulary (for searching) for subject matter of registered entities.

- **Service**. The registry must return a registered entity in response to a request by URN, URL, PI, and, or, FPI. That is, a user shall be able to request an XML-related entity by PI, FPI, URL, or URN (note that some entities may have multiple unique identifiers) and get the entity as the response. (This requirement can be thought of as applying to the repository, but it is convenient to imagine all requests being channelled through the registry.)

- **Metadata**. The registry must return metadata about registered entity in response to a request in a specified format that uses a unique identifier.

- **Revision**. It must be possible for the SO to request (and obtain) revision of information it provided, or that the RA assigned (or that is provided as added value) while not changing the registered entity.

1.5 In scope but not specified

The OASIS Registry and Repository Technical Committee has omitted to specify how to provide certain functionality that it is agreed is needed:

- **Submission**. Submissions to a registry may be made by an application-to- application process or through a human-manipulable interface. Certain semantics related to the purpose of the submission should be specified,

but it is not useful to specify the design of a human-manipulable interface. Further, while it may prove to be useful to specify a common method of application-to-application submission, the method of packaging a submission package should not be specified until current work in the IETF on XML packaging has borne fruit. Consequently, this specification does not prescribe any method for submitting items to a registry. Certain semantics related to submission are specified (below). This area will require further work in the future.

- **Service Description**. While it is considered desireable that any registry provide a way to obtain information about the registration authorities it supports, and other services, the OASIS Registry and Repository Technical Committee has not specified a method of doing so.

- **Interoperability**. Work on interoperability has been postponed to projected second phase of work in order to concentrate on the specification of the registry itself.

- **PIs and FPIs**. Support of the requirement to return registered entities in response to a request by PI or FPI has not been provided. It is envisioned that these unique identifiers should be conveyed as URNs, but construction of a URN name space for this purpose has not been essayed.

2. Information Model Overview

An OASIS registry/repository is a software system that consists of two components, a registry and a repository. A registry/repository is concerned with registered objects and registry entries. A *registered object* is something important that an author or producer wants to have visible to the world so that it can be discovered and used by a client or customer. A *registry entry* is relevant descriptive information about a registered object. Such information is called *metadata*. Registry entries are stored in a registry and registered objects are stored in a repository. A repository has only an elementary access method that allows retrieval of registered objects by object reference. In general, a registry entry will provide that object reference for some registered object.

A registry can operate independently, or it can be paired with a repository. In either case, the implementation must support a *registry services* interface. This interface can be used by abstract agents to assist a human or some other software process to register new objects, provide appropriate metadata for those objects, browse or query registry content, filter out irrelevant references, and retrieve selected registry entries or registered objects.

A registry/repository has administrative obligations, so it must be concerned with more than just the metadata for a specific registered item. It is also concerned with organizations and contacts within those organizations that submit or maintain submissions made to a registry. It is important that a registry be able to identify who has the privileges necessary to add or modify registry content and to maintain a log of all changes made.

2.1 Registry/Repository objects

Using UML notation, a registry/repository can be visualized as in Figure 1. The RegRepObject class represents the persistent objects of a registry/repository implementation. Each instance of this class has a unique identifier locally assigned by that implementation. Initially, this identifier is not visible to the public or as part of the registry services interface. It may be used locally to support associations among classes. Later, it may become the basis of a globally unique identifier that can be used for distributed processing among cooperating implementations. The five classes identified as subtypes of the RegRepObject class are the only classes that require independent object identifiers. All other classes presented in this specification are dependent on one of these five; whether or not these other classes have independent object identifiers is an implementation decision that has no effect on the semantics of the registry services interface.

The RegisteredObject class identifies all registered objects that are managed by the local registry/repository implementation. Some requests to register an object will be accompanied by the object itself and some will not. If the object to be registered is submitted to the registry/repository for storage and safe-keeping, then it will become a persistent instance of this class.

The RegistryEntry class identifies the metadata for a registered object. Each instance includes an object URL that can be used to locate the corresponding registered object. It also includes both a common name and a global name. The common name is intended for use by humans and is not necessarily unique except in some local human context. The global name, called the assignedURN, is used as an alias for the local object identifier since that identifier may not be visible to users. It is intended for use both by humans and by software systems and is unique within all registries that claim conformance to the registry/repository specification. The assignedURN can be used by registry services to Get the metadata for a registered object (see Section 7.2), or to Get the object itself (see Section 7.1). Each RegistryEntry instance also includes additional attributes to define the registered object's persistence and mutability, its administrative and payment status, and its submitting and responsible organizations. The RegistryEntry class will be linked to a number of other dependent classes. These dependent classes will maintain the classifications, associations, and other metadata associated with the registered object identified by a registry entry (see Figures 2 and 3).

The Organization class identifies all organizations that have some relationship to a registered object. This includes a submitting organization (SO), a responsible organization (RO), and a registration authority (RA). An organization wishing to become a submitting organization must first make an application to a recognized registration authority via a registry services request for that purpose. The end result is that a submitting organization and responsible contacts within that organization will be known to the registration authority before follow-on submissions from that organization will be accepted. Each organization will be assigned a globally unique orgURN that can be used as an alias for its local object identifier. An organization will be the basic authentication unit for accepting registry requests that modify the content of a registry or repository. Organizations can be subdivided as appropriate by parent/child relationships to achieve an appropriate granularity for authentication.

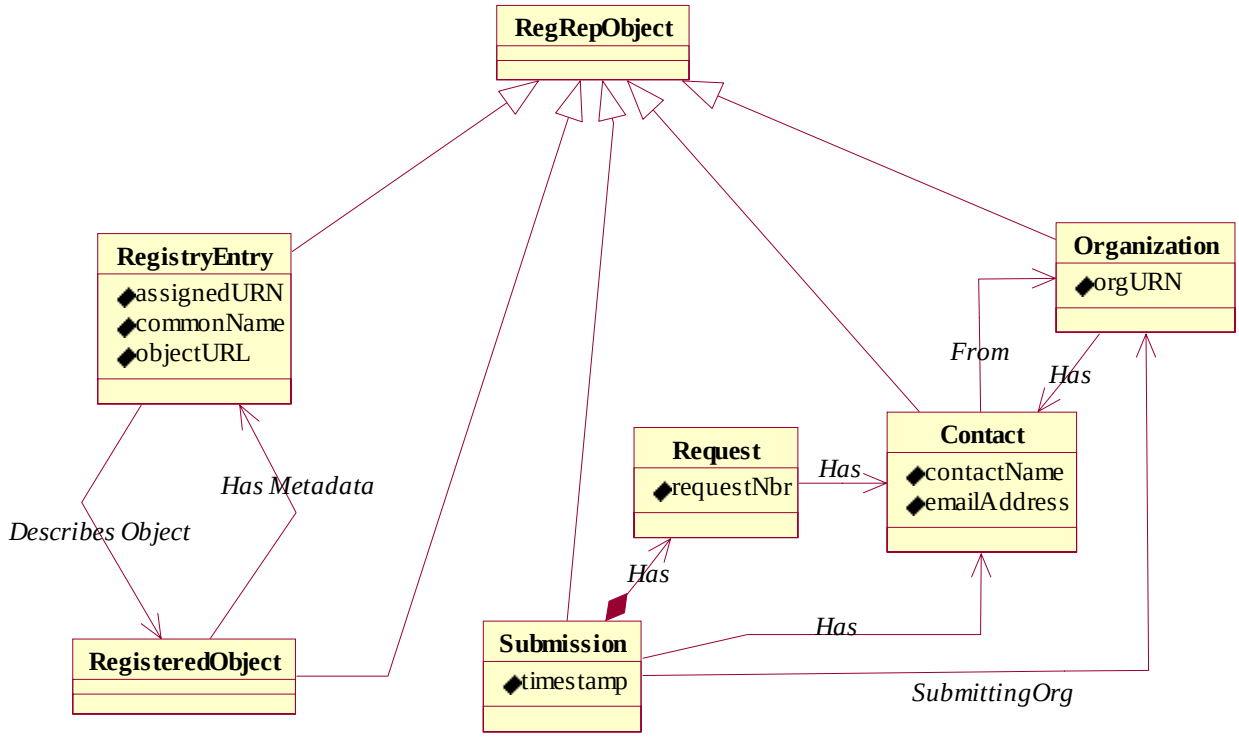

Figure 1 - Registry and Repository Objects

The Contact class identifies each person, role, or other entity within an organization that has some relationship to a registered object. A contact will consist of a contact name, i.e. an arbitrary string, used to identify the contact within an organization, and some specific contact information such as telephone number and email. Each contact instance has a non-public, local identifier created by the registration authority that is used to maintain required relationships among organizations, contacts, and registry entries. Each contact includes a mandatory reference to some organization and each organization must identify at least one contact for that organization. In addition, each submission must provide at least one contact. It is not necessary to maintain a global, unique name for contacts because the global name for the organization together with the common name and email address for the contact will be sufficient to identify the contact in any human context.

The Submission class identifies each submission made to a registry/repository implementation. A submission is a collection of requests, in the form of a message, sent from a submitting organization to a registry. For administrative accountability the registry logs each submission, timestamps it with the date and time received, and stores this information as a persistent instance of the Submission class. The required contact that accompanies a submission is stored as a Contact instance. Each request received as part of a submission is logged as an instance of the Request class for potential subsequent scrutiny. Any registry entities that are created, deleted, or modified by a request are traceable back to the submission instance and to its submitting organization. For accountability purposes, a registry will track all modifications to the metadata for any registered object. Thus there will be a many-to-many relationship among requests and registry entries to identify the type of impact each request has on one or more entries. The specific associations that must be maintained by a registry implementation are visible in Figure 4.

2.2 RegisteredObject and RegistryEntry

Details of the RegisteredObject and RegistryEntry classes are given in Figure 2. Registered objects are characterized by whether they are known or unknown objects. A known object is one whose data type and structure are known to the registry implementation. The registry will be able to open these objects and take appropriate actions based on their content. An unknown object is just a binary large object, i.e. BLOB, as far as the registry is concerned. For an unknown object, it will be up to the submitting organization to use the registry services interface to create all appropriate

metadata. The Known and Unknown classes in the figure represent this distinction, but they are essentially superfluous and have no attributes.

If a registered object's type is known to the registry, then the registry services interface will provide the ability to create and modify those objects. Initially, classification schemes and registry packages are the only registered objects whose content is managed by registry services. To create a new classification scheme, a submitting organization need only submit an XML document (see Section 8.7) that validates to the ClassificationScheme DTD described in Section 7.5. To create a new registry package, a submitting organization need only create and describe an empty registry package. Registry services can then be used to add and remove package elements. A DefineRegistryPackage service (see Section 8.8) provides other options for package creation.

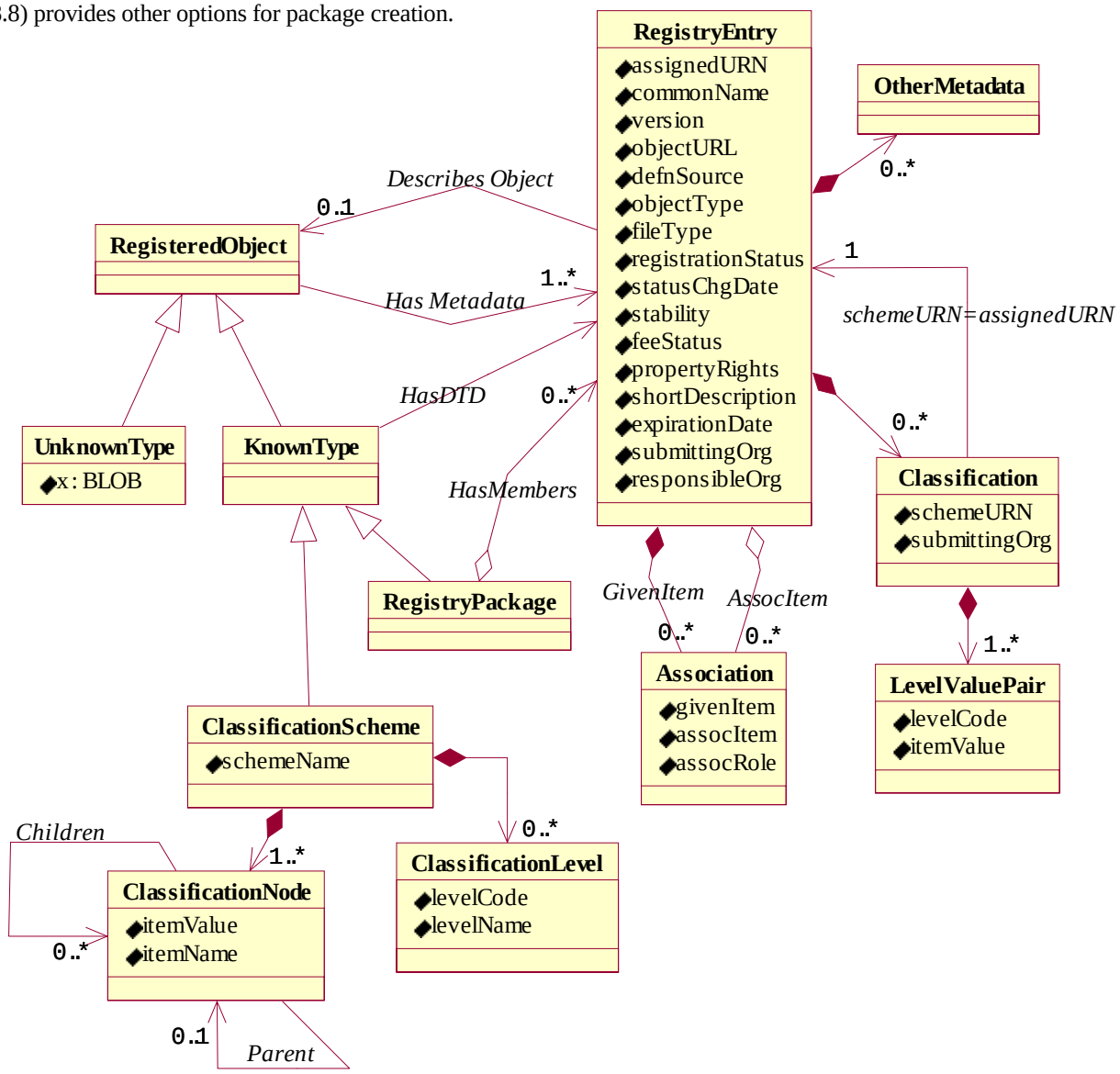

Figure 2 - Registry/Repository Class Diagram

The Association class records in the registry any real-world associations that may exist among the unknown registered objects. This allows new associations to be entered and retrieved through a standard interface without the need for accessing the internals of each such object. The Classification class allows the submitting organization to classify its registered objects according to any previously registered classification scheme. It also allows third parties to classify registered objects according to classification schemes they are more comfortable with. The Classification class includes a submittingOrg attribute to identify classifications submitted and owned by a third party. The RegistryEntry, Association, and Classification classes are the most important in this model. The OtherMetadata class is an abstract placeholder for simple convenient metadata features of the model whose details are presented in Figure 3.

2.3 Associations

The Association class represents binary associations between registered objects. If a registered object X has a specific association with a registered object Y, then that association is represented implicitly by a pairwise association between a registry entry for X with a registry entry for Y. One item in the pair is called the *given item* and the other item is called the *associated item*. Both the given item and the associated item are registry entries in the same registry as the association instance. The name of the original association between X and Y is represented by the *assocRole* attribute of the association instance.

For Oasis, the following are supported association roles:

Role Name	Meaning
Validates To	The given item validates to the specification provided by the associated item. If the objectType of a given item is *instance*, then a validatesTo association must exist and the associated item must be of type *definition*. Examples: an XML document validates to an XML schema; a classification scheme validates to the Classification Scheme DTD defined herein; a program validates to its UML class specification.
Requires	The given item requires the presence of the associated item. The expectation is that the associated item must be retrieved before the given item can be processed or used. If the given item is retrieved, the default action is that the required items are NOT retrieved along with it. However, a retrieve request may have variants that allow recursive retrievals. .Examples: an XML element requires the presence of some other XML element or entity that it references; a software program requires the installation of some other program before it will execute properly.
Contains	The given item is a registry entry for a registry package that has a reference to the associated item as one of its package members. This role is used exclusively to support the RegistryPackage class.
Is Superceded By	The given item is superceded by the associated item. Only the SO of a given item can say that it is superceded by some other registered object. The registered object of the given item is still registered, its registry entry still points to it, and its registrationStatus becomes *superceded*. Example: one version of a registered object is superceded by a newer version, but the old version remains available.
Is Replaced By	The given item is replaced by the associated item. Only the SO of the given item can say that it is replaced by another registered object. The given registered object is no longer registered, but its registry entry remains in the registry, its registrationStatus becomes *replaced*, and the objectURL of that registry entry now points to the other registered object. Example: a new, upward compatible version of a registered object replaces the existing version. All pointers to the old version, via the old registry entry, now point to the new version via the new objectURL.
Is Related To	The given item is related to the associated item. This is a very loose association that has no dependency implications. Example: a GIF graphic is related to the classification scheme that it visualizes, and *vice versa*; a company catalog is related to a company purchase order DTD, and *vice versa*. But each one-way relationship is created and maintained by the SO of the given item.

2.4 Classification schemes and classifications

A *classification scheme* is a fixed hierarchy of nodes, where each node has a name. The node names may not be unique so it is sometimes necessary to know a complete path of node names from the root of a hierarchy to a given node before that node can be uniquely determined. Classification schemes are useful tools to categorize and describe the various properties of a population. A well-known classification scheme is the 5-level hierarchy, NAICS, used to classify segments of North American industry (cf. http://www.census.gov/epcd/naics/naicscod.txt).

A *classification* is a reference to a single node of a classification scheme. For example, the NAICS code 11114 represents a node at the 4-th level of the NAICS classification tree. The value is really a sequence of 4 values 11, 1, 1, 4, where 11 represents "Agriculture, Forestry, Fishing and Hunting", the next digit 1 represents "Crop Production", the next digit 1 represents "Oilseed and Grain Farming", and the final digit 4 represents "Wheat Farming".

The most common uses of classification schemes are to identify enumeration domains. If an enumeration domain consists of N distinct values, then it can be specified as a 1-level classification scheme of N nodes, all at Level 1. More sophisticated examples of multi-layer classification schemes are the NAICS scheme described above, the 3-level IPTC scheme for classifying news articles (cf. http://www.iptc.org/SubjectView.zip)), and the 7-layer scheme used by biologists to classify all living things by Kingdom, Phylum, Class, Order, Family, Genus, and Species.

2.4.1 Classification scheme definition

A classification scheme S is a pair (N, ≤) where N is a set of nodes and ≤ is a partial ordering over N, with the additional requirement that the set of predecessors of every node is linearly ordered by the partial ordering and has a unique first element. Every node, x, is assigned a level number by the expression Level(x) = Card(Pred(x))+1, where Pred(x) is the set of predecessors of the node x under the partial ordering and Card(Pred(x)) is the cardinality of that set. All nodes that have no predecessors are at level 1. The number of levels in the classification scheme is defined to be the maximum of {Level(x) | x ∈ N}.

If N is a finite set with cardinality n, then every classification scheme S with N as its set of nodes has exactly k levels, for some integer k between 1 and n inclusive. The ClassificationScheme DTD specified in Section 7.5 allows a scheme-definer to declare an itemValue and an itemName for each node, an optional levelCode and levelName for each level, and a schemeName for each classification scheme S. A unique identifier for each node is assumed in order to represent the node hierarchy, but that identifier may not be visible to the user and plays no role in the registry services interface.

2.4.2 Representation

A classification scheme may be defined and represented by the ClassificationScheme DTD defined in Section 7.5. This DTD will have a registry entry, i.e. to declare its URN and to describe its registration status and other metadata, in every Oasis conformant registry/repository. This Oasis standard DTD, or some later upward compatible version of it, can then be used to register and exchange many different classification scheme instances.

Every registered object that is classified according to a given registered classification scheme will have a registry entry linked to a Classification instance that references a single node of that classification scheme. An XML Element to represent a classification instance is defined in Section 6.3. A registered object may be classified according to any number of different classification schemes.

2.4.3 Example

The following XML instance uses the ClassificationScheme DTD (Section 7.5) to define a 2-level named and coded classification scheme for StudentStatus in a university. As an option, it names the two levels as Primary Classification and Secondary Classification.

```
<ClassificationScheme schemeName="StudentStatus">
<ClassificationLevel levelCode="primary" levelName="Primary Classification" />
<ClassificationLevel levelCode="secondary" levelName="Secondary Classification/>
<ClassificationNode>
<ClassificationItem itemValue="FR" itemName="Freshman"/>
</ClassificationNode>
<ClassificationNode>
<ClassificationItem itemValue="SO" itemName="Sophomore"/>
</ClassificationNode>
<ClassificationNode>
<ClassificationItem itemValue="JR" itemName="Junior"/>
</ClassificationNode>
<ClassificationNode>
<ClassificationItem itemValue="SR" itemName="Senior"/>
</ClassificationNode>
<ClassificationNode>
<ClassificationItem itemValue="SP" itemName="Special"/>
    <ClassificationNode>
    <ClassificationItem itemValue="D" itemName="Degree Candidate"/>
    </ClassificationNode>
    <ClassificationNode>
    <ClassificationItem itemValue="N" itemName="Non-Degree Candidate"/>
    </ClassificationNode>
```

```
</ClassificationNode>
</ClassificationScheme>
```

The following XML instance uses the XML Classification Element (Section 6.3) to represent the classification of a given student as a Special Student in Non-Degree Candidate status according to the StudentStatus classification scheme.

```
<Classification  schemeURN="StudentStatus">
<LevelValuePair  levelCode="primary" itemValue="SP"/>
<LevelValuePair  levelCode="secondary" itemValue="N"/>
</Classification>
```

2.5 Registry packages

A registry package is a set of pointers to registry entries. Note that there are two levels of indirection here! A package is a set of pointers, not a set of registry entries; thus a registry entry can be represented as an element of many different registry packages without being copied multiple times. A registry package will often represent a collection of registered objects, but the registered objects can be determined only by first going to the registry entry that references that registered object to get its URL. This indirection is purposeful. It allows registered objects to be withdrawn, or superceded or replaced by other registered objects, without changing the content of the registry package. With this convention, a registry package should remain stable for a longer period of time and its references will never be lost. The user of a registry package always has the option to check to determine if a registry entry member of the package still references the original object. The status of the registered object can be determined from the registrationStatus attribute of the registry entry. The GetRegisteredObject registry service defined in Section 7.1 allows one to retrieve all of the registered objects referenced by the member elements of a package, to an arbitrary level of recursion.

The XML representation of an existing registry package is given by the RegistryPackage DTD defined in Section 7.6. This DTD will be used to return a registry package to a user as the result of a registry services request to get that package. The DefineRegistryPackage request element, specified in Section 8.8, can be used to submit a collection of metadata for different objects to a registry, with the result being that each object is registered individually and the whole collection is registered as a registry package instance.

2.6 Other metadata

The OtherMetadata class in Figure 2 is just a place holder for a number of other dependent UML classes that contain additional metadata for a registered object. This specification defines four other classes that give helpful structure for capturing various kinds of additional information. The AlternateName class is needed to capture different names for a registered object in different environments, including potentially different URN's assigned by other registries. The Description class is needed to provide an opportunity for descriptions of the registered object in any number of different human readable languages. The ExternalData class is very helpful for capturing references to other informational items that are strongly related to the registered object but not important enough to be registered themselves, and the Contribution class provides an opportunity to list the creators of a registered object. Figure 3 provides the details of the RegistryEntry class together with these other classes that are dependent on it. The following paragraphs give additional discussion for the intended usage of these classes.

2.6.1 External data

External data is a conceptual notion used to reference data objects that are related to a registered object but are not themselves registered objects in any conforming registry. This category of metadata is reserved for things like graphic visualizations, example sets, white papers, usage scenarios, extended documentation, vendor propaganda, etc. Sometimes such objects will be very important, e.g. usage documentation, and will themselves be registered objects. At other times, these objects will be less important support information for a registered object and will not be registered. In the later situation, the registry may maintain a simple list of references to the external data. Each such reference becomes an instance of the ExternalData class. The registration authority will keep a list of the names and types of the external data objects, with just enough additional information so that they can be presented as options on a web page. The external data will be created, held, and maintained by the submitting organization or some other external, non-conforming repository.

Each instance of the ExternalData class consists of a human readable name, a URL, a standardized classification or *related role* as a supporting object, a MIME file representation type, the size of the file in bytes, and a short human

readable comment. The *related role* may be defined by the registry's sponsoring organization, thereby giving a very specific reference to required or optional documentation. As with *association roles*, an advantage of the standard information model is that even though the related roles are defined for specific purposes, they can be declared, modified, or queried via standardized access methods.

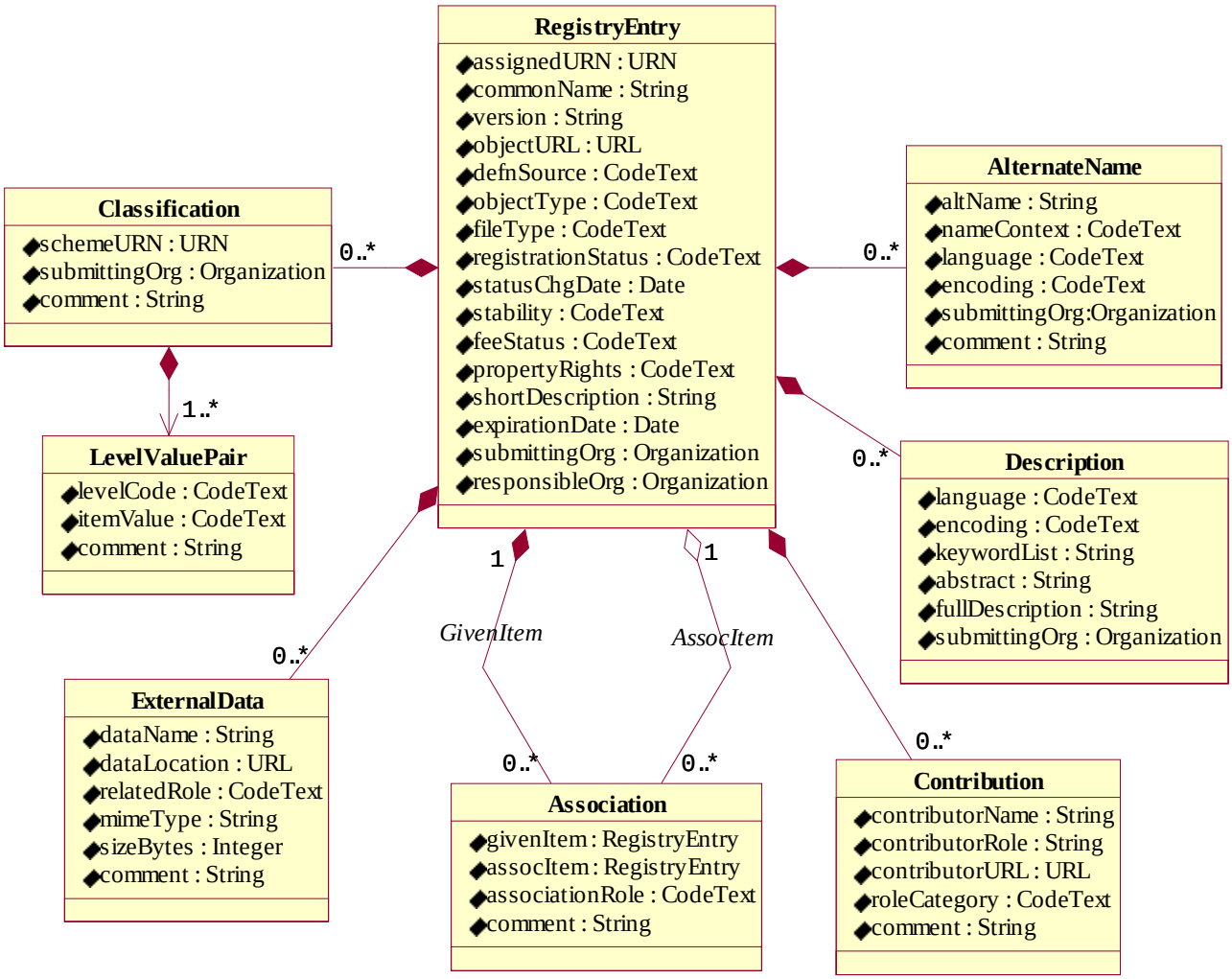

Figure 3 - RegistryEntry with dependent classes

2.6.2 Alternate names

Alternate name is a conceptual notion used to represent alternate or alias names for a registered object. Especially significant are names used in special circumstances, e.g. short names for local identifiers in a specific programming language context, or globally unique qualified names that satisfy a specific hierarchical qualification structure. In many cases the alternate names will include the globally unique names assigned to the registered object by other registration authorities. Alternate names can also be used for names in different human languages with characters encoded in unusual character sets.

An AlternateName instance consists of the alternate name, an abstract nameContext to identify the contextual category in which that name is used, and a human readable comment that further explains how that name should be used. Each AlternateName instance is tagged with a reference to its submitting organization, since alternate names may come from many different sources. An alternate name is also tagged with optional language and character set codes that apply to the alternate name and any optional comments. As with *association roles* and *related roles*, the nameContext of an

alternate name may be defined by the sponsor of a registry specification, thereby allowing unique usage but retaining standardized access methods.

2.6.3 Descriptions

A description is intended to play the role of an abstract and keyword list commonly required for library resources. The shortDescription attribute of the RegistryEntry class is too short to satisfy this purpose. The Description class allows longer and more complete plain text descriptions of registered objects in different human readable languages. The intent is that all description instances for a given registered object are semantically equivalent; each being a direct translation into a different human language of some source description in the original human language.

Each Description instance consists of a plain text fullDescription, as well as an optional keywordList and abstract. All of these attributes are tagged with a human language code and a character set code that apply to any text in the instance. In addition, each Description instance is tagged with a reference to its submitting organization, since descriptions may come from many different sources.

2.6.4 Contribution

Contribution is an abstract notion used to identify people, places, or organizations that contribute in any way to the creation of a registered object. This notion is particularly significant for library resources and educational or learning materials. Contribution instances are different than Contact instances, although they could overlap. Contacts are more like salespeople, who can speak to the final product, its availability, usage, registration status, and future plans, whereas Contribution instances will be much more like movie credits, giving credit even for very specialized contributions to the creation of the registered resource, e.g. editor, co-author, illustrator, technical support, etc. The intent is that contributions be presented in much the same way as credits at the end of a motion picture. It usually suffices to give a name and the role played.

A Contribution instance consists of the name of the entity deserving recognition, the role it played in the production, an abstract *roleCategory* with valid values defined by the sponsor of the specialization being used, an optional URL to help locate the home page of the named entity, and an optional comment that might further explain the role played by that entity in the production of the registered object. As with *association roles*, an advantage of the proposed information model is that the *roleCategory* can be defined by the sponsoring group so that contribution instances can be categorized in very specific ways. In addition, contribution instances can be declared, modified, or queried via standardized access methods.

2.7 Registry administration

An information model for a registry/repository is concerned with more than just the metadata for registered objects. It is also expected to provide support for maintaining some degree of version control for these objects as well as administrative accountability for the submission and management of registry entries. The UML class diagram in Figure 4 provides a structure whereby all of these administrative expectations can be met. Each class plays two roles: it identifies the structure of each instance and it gives the name of a container for all persistent instances of that type.

The administration facility must keep track of the following:

- An organization contact for each Organization instance.
- A submission contact for each Submission instance.
- Optional administrative and technical contacts for each Request instance.
- A date and time for each request with an appropriate historical log of changes to registry content.
- An Impact instance for each impact that a Request has on any registry entry.
- Authentication criteria for each organization.

2.7.1 Submission privileges

Different registries will have quite different rules for who is allowed to submit objects for registration. Some organizations will want to establish registries for presenting the intellectual efforts of their membership to the rest of the world, so it will be open to their entire membership. Others may be very specific and have only a very limited number of submitters. Some registries will be open to the public, soliciting and registering reviews of products or services. A

major application of Oasis registry/repository implementations will be for supporting the workflow and business processes of electronic commerce. As such they will be open to new input from nearly any legitimate business or trade organization.

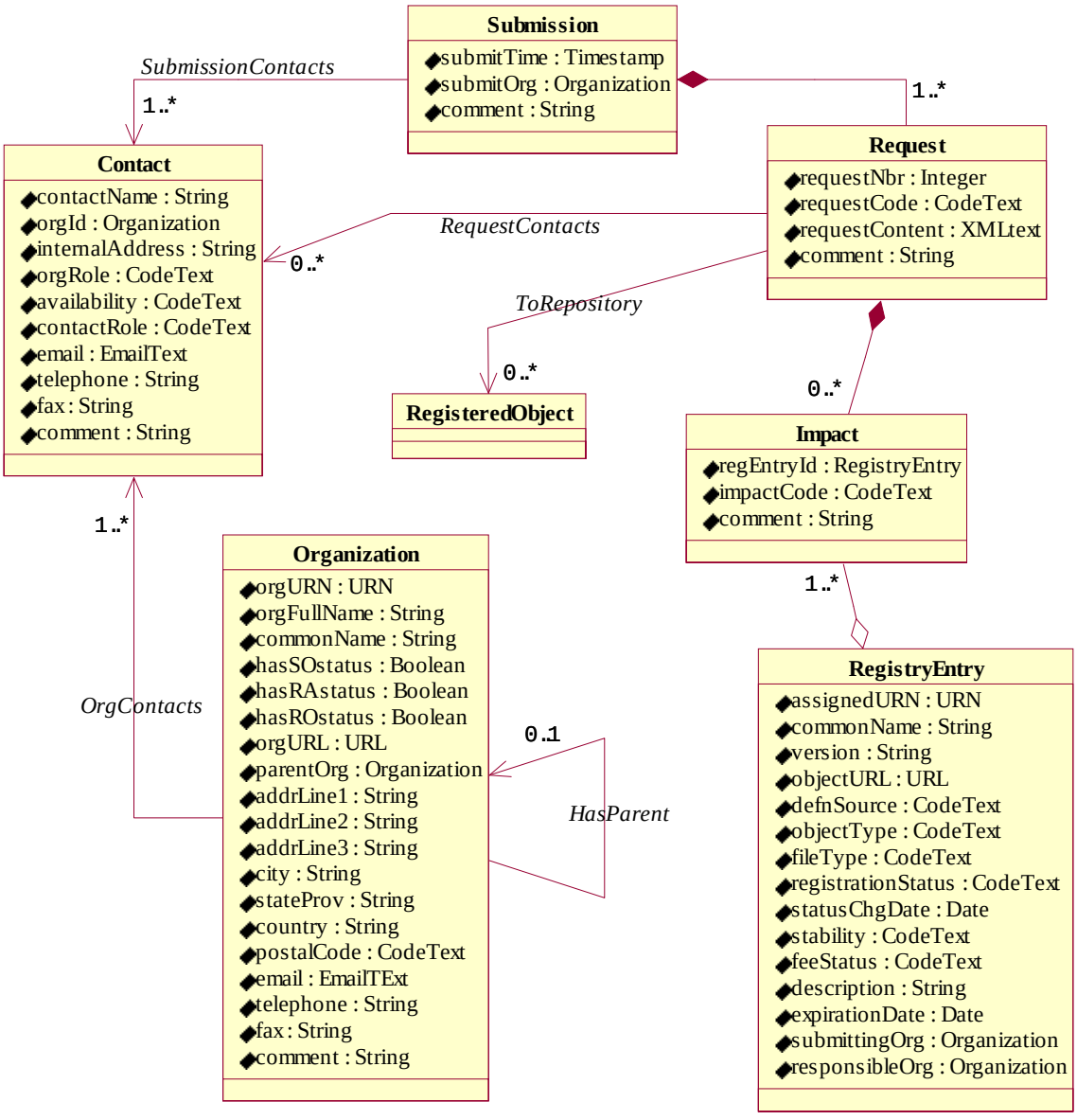

Figure 4 - Registry Administration

An Oasis registry/repository implementation will maintain a list of all of the organizations it is prepared to do business with, along with a minimal amount of contact information for each one. Any additional information needed could be obtained by the required submittal of a company profile for registration. Minimal contact information for an organization includes a full legal name, address, telephone, and one or more contact points with contact names and email addresses. The registry will pre-load the organization container with known organizations and then let others apply for status to make submissions. Any registry service request that will modify registry content must be authenticated as from some known submitting organization before the request will be executed. There needs to be a bootstrap registry service that allows organizations to request that they be recognized as submitting organizations. The RegisterSubmittingOrg request defined in Section 8.19 satisfies this need.

There are three types of organizations: registration authority (RA), responsible organization (RO), and submitting organization (SO). A registration authority will be certified by some central authority to maintain a conforming registry/repository. RA's will trust one another and freely share registry information. Responsible organizations will often be standards committees, consortia, or trade associations that develop and maintain specifications. RO's may or may not maintain registries for their own products. Submitting organizations will be involved with the bulk of registry interaction by submitting new information.

2.7.2 Submission workflow

A Submission is a collection of requests, in the form of a message, sent from a submitting organization to a registry. An assumption is that some transport service has delivered the message to the registry and that some authentication service has authenticated the submitter as a legitimate submitting organization. This specification includes an authentication token in the SubmitRequest DTD (Section 7.3) to assist in the authentication process. Once a SubmitRequest DTD has been received and its sender authenticated, the main registry/repository component swings into action.

The registry logs each new submission, assigns it a persistent object identifier, timestamps it with the date and time received, adds one new entry to the Submission container and one or more new entries to the Contact container. It then considers each request separately. The details of this process are specified by the semantic rules of the SubmitRequest DTD in Section 7.3, and its Request Elements specified in Section 8.

Each request is assigned a request number to distinguish it from other requests in the same submission. Since a request is part of a submission it is not necessary to assign each request a separate unique object identifier, although a request number is used to distinguish among requests. In addition, each request is assigned a *request code* from the RequestType enumeration domain specified in Section 4.3. The request codes determine a simple 1-level classification scheme for requests. They are in a one-to-one correspondence with registry service requests that can effect a change of registry content. Maintaining this information as part of registry content supports queries to determine which requests result in objects being superceded or replaced, or which requests alter other metadata content. The request code and the XML content of each request are stored in the Request container. After some period of time, the XML content of each request instance may be deleted, but the remainder of each request instance is kept permanently as part of the administrative record.

2.7.3 Impact workflow

A request may have an impact on one or more registry entries. For example, a request to supercede registered object A with a new registered object B will have impacts on the registry entries for both A and B. For accountability and versioning control, the registry must retain a record of all such impacts, including identification of the submitting organization, the date and time of the request, and the impact on registry content.

If a request creates a new registry entry, it is necessary to link that request to the entry it creates so that there will be an administrative record of the date and time the entry was created and who owns it. Similarly, if new associations or new external data items are submitted for an existing registry entry, it is necessary to maintain an administrative record of the date and time of each addition. As part of registry version control requirements we especially want to maintain a record of all requests that result in replacement of one registered object by another, the registration of a new version that supercedes a previous version, or the deprecation or withdrawal of a registered object. The classification, alternate name, and description instances may be submitted by organizations other than the owner of the registry entry they are linked to, so the registry should maintain an administrative record of all such additions or modifications.

The registry maintains an administrative record of all new versions and replacements and all other additions or modifications to registry content by supporting a many-to-many relationship between Request instances and RegistryEntry instances. The Impact class records this information. The registry will populate the Impact container as it performs the actions of each request. All impact instances are created and modified solely by the registration authority. Each impact instance is assigned an impactCode from the ImpactCode enumeration domain specified in Section 4.4.

The impact codes determine a simple 1-level classification scheme for impact instances. They are in a one-to-one correspondence with add, delete, and update operations on the ten basic classes that make up the registry/repository information model. This classification will support queries on registry content to determine which requests result in impacts of a given type on registered objects. All impact instances are permanently maintained as part of the administrative record.

3. Registry Classes

3.1 RegistryEntry

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
assignedURN	URN	Mandatory	RA	This Standard
commonName	LongName	Mandatoty	SO	SO
version	CodeText		SO	SO
objectURL	URL		SO	W3C
defnSource	CodeText	Mandatory	RA	This Standard
objectType	CodeText	Mandatory	SO	defnSource
fileType	CodeText		SO	defnSource
registrationStatus	CodeText	Mandatory	RA	defnSource
statusChgDate	Date	Mandatory	RA	This Standard
stability	CodeText	Mandatory	SO	defnSource
feeStatus	CodeText		SO	defnSource
propertyRights	CodeText		SO	defnSource
shortDescription	CommentText	Mandatory	SO	SO
expirationDate	Date	Mandatory	RA	RA
submittingOrg	Organization	Mandatory	RA	This Standard
responsibleOrg	Organization		SO	This Standard
regEntryId	*RegistryEntry*	*Mandatory*	*RA*	*This Standard*

Semantic Rules

1. The RegistryEntry class represents the set of all registry entries in the Registry. Each instance identifies a single registry entry. A registry entry instance holds only some of the metadata for a registered object; other metadata is held by other classes in the Registry.

2. Each registry entry instance is assigned a unique identifier by the Registration Authority (RA). This hidden attribute, herein called regEntryId, is said to be of type RegistryEntry. It is used to represent relationships of this instance with other information in the Registry.

3. The assignedURN is assigned by the RA. It is created to be a unique identifier for each RegistryEntry instance within this Registry. In most cases it will be a natural construction derived from the commonName and version attributes of the registry entry. The assignedURN may or may not be a unique identifier for the referenced registered object within all OASIS conforming Registry/Repository implementations. When a Submitting Organization (SO) makes a submission to the Registry, it may provide a suggested URN for that object. The RA may create a new URN or use the suggested URN. If the registry entry references a registered object in some other Oasis conforming registry, then the RA may choose to use the other registry's URN as the assignedURN for the registry entry. If a group of registries are cooperating to all hold registry entries for the same registered object, then all registries may choose to use the URN created by the registry in whose repository the original registered object resides.

4. The commonName is provided by the SO. It is the name commonly used among humans to reference the registered object.

5. The version is provided by the SO. It can have an arbitrary format and is used only to help distinguish one registry entry from another having the same common name. The assignedURN will be different for different versions.

6. The objectURL is a URL that identifies the location of the registered object. If the RA is also a repository for the item, then the RA will receive the object, store it in the Repository, and create an anonymous FTP locator as a value for objectURL. If the Registry is not also a Repository, then the objectURL is provided by the SO and the RA has no further responsibility. Depending on the value of the feeStatus attribute, the objectURL URL may need to be supplemented with payment or password information before the file containing the object can be retrieved. Some Registries may distinguish themselves by also being the Repository for any free, publicly available objects described in the Registry.

7. The defnSource takes its value from the DefinitionSource enumeration domain defined in Section 4.1. It identifies a collection of accredited Registry/Repository development organizations. If the Registry claims conformance the OASIS Registry/Repository, then the defnSource is OASIS.

8. The objectType is provided by the SO and takes its value from the ObjectType enumeration domain defined in Section 5.1.

9. The fileType is provided by the SO and takes its value from the FileType enumeration domain defined in Section 5.2. EDITOR's NOTE: How do we handle Other?

10. The registrationStatus is provided by the RA with its value taken from the RegistrationStatus enumeration domain defined in Section 5.3.

11. The stability attribute is provided by the SO with its value taken from the Stability enumeration domain defined in Section 5.4. There is no default value for stability. The SO is required to declare a stability categorization for each submittal by choosing from among the codes for *Static*, *Dynamic*, or *Compatible*.

12. The feeStatus attribute is provided by the SO with its value taken from the FeeStatus enumeration domain defined in Section 5.5. The default value is the code for *Free*, meaning that the registered object is freely available with no password or payment required.

13. The propertyRights attribute is provided by the SO with its value taken from the PropertyRights enumeration domain defined in Section 5.6. The default value is the code for *None*, meaning that there are no property rights specified for this registered object.

14. The shortDescription is provided by the SO. It is a very abbreviated description of the registered object. More complete descriptions in various human readable languages are held in the Description class.

15. The expirationDate is assigned by the RA upon suggestion from the SO. Some RA's may follow very definite procedures for the length of time an object can remain registered before an affirmation or withdrawal action is required. If the Expiration date passes without an SO action, then the RA may initiate an expiration action.

16. The submittingOrg identifies the organization submitting the registered object. It points to a unique instance of the Organization class whose hasSOstatus attribute is true. An organization can achieve SO status via the RegisterSubmittingOrg request (Section 8.19).

17. The responsibleOrg identifies the organization responsible for creation and maintenance of the registered object. It points to a unique instance of the Organization class. The RO may be a formal accredited standards development organization or it may be the SO. If it is not the SO, then its hasROstatus attribute must be true. The procedure for an organization to achieve RO status is not part of this specification.

18. Let SO be the original submitting organization of a registry entry RE that was submitted with some registered object R. Then SO is the *owner* of both RE and R. Some other submitting organization SO' may submit a registry entry RE' whose objectURL points to R. SO' is the *owner* of RE', but SO remains the owner of R. Only the owner of R may change the following attributes in any registry entry whose objectURL is a locator for R: registrationStatus, stability, feeStatus, or propertyRights.

3.2 Association

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
givenItem	RegistryEntry	Mandatory	RA	This Standard
associationRole	CodeText	Mandatory	SO	defnSource
assocItem	RegistryEntry	Mandatory	RA	This Standard
comment	CommentText		SO	SO

Semantic Rules

1. The Association class represents a many-to-many relationship from the RegistryEntry class to itself. Each association instance relates a given registry entry instance to an associated registry entry instance. The given item plays the specified associationRole with the associated item.

2. The givenItem attribute identifies the given item in the association instance. The given item identifies the parent registry entry upon which the association instance is dependent.

3. The associationRole is provided by the SO with its value taken from the AssociationType enumeration domain defined in Section 5.7.

4. The assocItem attribute identifies the associated item in the association instance. The association instance is not dependent upon the assocItem; if a referenced assocItem should be deleted, then this reference may point to a non-existent object, but the association instance remains. The RA may provide a warning message to the deleter of the associated item and may choose to append information about the deleted item to the comment.

5. The triple (givenItem, associationRole, assocItem) uniquely determines an association instance.

6. The comment is provided by the SO. It may explain in more detail the relationship between the given and associated items.

3.3 Classification and LevelValuePair

Format

Classification

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
schemeURN	URN	Mandatory	SO	RA
submittingOrg	Organization	Mandatory	RA	This Standard
comment	CommentText		SO	SO

LevelValuePair

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
levelCode	CodeText	Mandatory	SO	This Standard
itemValue	CodeText	Mandatory	SO	This Standard
comment	CommentText		SO	SO

Semantic Rules

1. The Classification class identifies all classifications of a registry entry.

2. Each Classification instance has an implicit value, regEntryId, of type RegistryEntry that links this instance to its parent registry entry. The classification instance is dependent on that registry entry.

3. The schemeURN is provided by the SO. It identifies the assignedURN of some registry entry in the same Registry as the classification instance, and the object referenced by that registry entry must be a classification scheme.

4. The submittingOrg is provided by the RA. It identifies the organization that submits the classification for the parent registry entry; it may differ from the organization that owns the parent registry entry.

5. The pair (schemeURN, submittingOrg) uniquely determines a classification instance for the parent registry entry.

6. The optional Classification comment attribute is provided by the SO. It may explain why this particular classification scheme was chosen to classify the registered object and it may identify the context of the classification.

7. Each LevelValuePair instance has implicit values for schemeURN and submittingOrg that link this instance to its parent Classification instance.

8. The set of LevelValuePair instances linked to a Classification instance identify a branch in a classification scheme hierarchy that terminates in a specific node. The registry entry maps to that node.

9. The levelCode is provided by the SO, or the RA fills in "leaf" by default, depending on the type of classification scheme identified by the schemeURN.

10. The itemValue attribute is provided by the SO. It must be a legal itemValue as defined by the registered classification scheme.

11. The optional LevelValuePair comment is supplied by the SO. It might further explain the meaning of the itemValue.

3.4 Organization

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
orgURN	URN	Mandatory	RA	This Standard
orgFullName	LongName	Mandatory	SO	This Standard
commonName	ShortName		SO	SO
orgURL	URL		SO	W3C
hasSOstatus	Boolean		RA	This Standard
hasROstatus	Boolean		RA	This Standard
hasRAstatus	Boolean		RA	Higher Authority
parentOrg	Organization		RA	This Standard
addrLine1	AddrLineText		SO	This Standard
addrLine2	AddrLineText		SO	This Standard
addrLine3	AddrLineText		SO	This Standard
city	ShortName		SO	This Standard
stateProv	ShortName		SO	This Standard
country	ShortName	Mandatory	SO	This Standard
postalCode	CodeText		SO	This Standard
email	EmailText		SO	Internet
telephone	TelephoneText		SO	This Standard
fax	TelephoneText		SO	This Standard
comment	CommentText		SO	SO
orgId	*Organization*	*Mandatory*	*RA*	*This Standard*

Semantic Rules

1. The Organization class represents the set of all companies and organizations known to a Registry. Each instance represents a single organization or company. If an organization is part of another organization, or if a company is a sub-unit of another company, then that organization is linked to its parent.

2. Each Organization instance is assigned a unique identifier by the Registration Authority (RA). This hidden attribute, herein called orgId, is said to be of type Organization. It is used to represent relationships of this instance with other information in the Registry.

3. Each organization instance has a relationship with one or more Contact instances. The registry services defined herein may provide a method to access these contact(s) for each organization.

4. The orgURN is assigned by the Registration Authority (RA). In most cases it will be a natural construction derived from the commonName of the company. The orgURN attribute uniquely determines an Organization instance in this Registry. It may or may not be a unique identifier for this organization within all OASIS conforming Registry/Repository implementations; however, cooperating registries may choose to make it so.

5. The orgFullName is provided by the Submitting Organization (SO) or by the Responsible Organization (RO). It is intended to be the full legal name of the organization in the home country of its existence. The orgFullName and country values together uniquely determine an organization instance.

6. The commonName is provided by the SO. It is the name or symbol that is used in common discourse to identify the company or organization.

7. The hasSOstatus attribute identifies whether the organization is properly registered for submitting requests to this registry that have the effect of changing registry content. The RA determines this status based on published criteria that are not part of this specification. See definition of Submitting Organization in Section 4.2.

8. The hasROstatus attribute identifies whether the organization can have responsibility for registered objects that they have not themselves submitted to the registry. The RA determines this status based on published criteria that are not part of this specification. .See definition of Responsible Organization in Section 4.2.

9. The hasRAstatus attribute identifies whether the organization is recognized as a Registration Authority. This determination is made by a higher authority than the RA and the criteria are not part of this specification. See definition of Registration Authority in Section 4.2.

10. The orgURL is provided by the SO. It is the locator for the home page of that company or organization on the world wide web.

11. The parentOrg is provided by the SO or the RA. If the company or organization is a subsidiary of a company already listed in this entity, then parentOrg is the identifier of the parent organization. If the organization has no known parent, then this attribute is null.

12. The three addressLine attributes identify the street or post office address of the company or organization. addressLines that are not needed are set to null. They are provided by the SO.

13. The city is provided by the SO. It identifies the city associated with the AddressLines.

14. The stateProv is provided by the SO. It identifies the state, province, territory, or some other political unit between the city and country.

15. The country is provided by the SO. It identifies the home country of the company or organization.

16. The postalCode is provided by the SO. Together with the street address, city, and country, it is the mailing address of the company or organization headquarters.

17. The email is provided by the SO. It identifies the best email address to use when trying to reach the company or organization headquarters.

18. The telephone attribute is provided by the SO. It identifies the best international telephone number to use when trying to reach the company or organization headquarters.

19. The fax attribute is provided by the SO. It identifies the best international facsimile telephone number to use when trying to reach the company or organization headquarters.

20. The comment is provided by the SO or RA. It conveys any important information not captured by the other attributes.

3.5 Contact

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
contactName	ShortName	Mandatory	SO	SO
orgId	Organization	Mandatory	RA	This Standard
internalAddr	AddrLineText		SO	This Standard
orgRole	CodeText	Mandatory	SO	This Standard
availability	CodeText	Mandatory	SO	defnSource
contactRole	CodeText	Mandatory	SO	defnSource
email	EmailText	Mandatory	SO	Internet
telephone	TelephoneText		SO	This Standard
fax	TelephoneText		SO	This Standard
comment	CommentText		SO	SO
contactId	*Contact*	*Mandatory*	*RA*	*RA*

Semantic Rules

1) The Contact class identifies all of the contacts known to the Registry. A Contact instance is a single contact. A contact may be a person, a company office, or a job title that can respond to Registry related issues.

2) Each Contact instance is assigned a unique identifier by the Registration Authority (RA). This hidden attribute, herein called contactId, is said to be of type Contact. It is used to represent relationships of this instance with other information in the Registry.

3) Each contact instance is linked to a single organization. However, the contact instance is not dependent on that organization instance.

4) The contactName is provided by the SO. It identifies a person or office associated with the SO able to speak to technical or administrative questions about registered objects. The name need not be globally unique, but it is intended that it be recognized by whoever may answer the telephone number or email address provided in this instance.

5) The orgId is provided by the RA. It is derived from the orgURN provided by the SO as part of a submission or request. Each contact instance is linked to a single organization; however, that organization need not be the same as the submitting organization if the contact is named in a submission or request.

6) The internalAddr is provided by the SO. It identifies the mail stop, building, or room number of the contact. When appended to the organization address it provides a complete mailing address for the contact.

7) The orgRole is provided by the SO, with its value taken from the OrganizationRole enumeration domain defined in Section 4.2. The default value is the code for *Submitting Organization*, meaning that the contact is from the same organization as the one making the submission that contains this contact instance.

8) The availability code is provided by the SO with its value taken from the ContactAvailability enumeration domain defined in Section 5.9. The default value is the code for *Public*, meaning that the contact name and email address are available to all users of the registry.

9) The contactRole is provided by the SO with its value taken from the ContactRole enumeration domain defined in Section 5.10. The default value is the code for *All Issues*, meaning that the contact is able to respond to all issues, e.g. both technical and administrative, involving each registry entry the contact is associated with.

10) The email address is provided by the SO. It is a global email address that is the best address to use when trying to contact someone who can respond to issues involving registered items. NOTE: The email address is mandatory in this specification because we anticipate follow-on *subscription* services that may require notification of certain identified contacts based on registry actions.

11) The telephone number is provided by the SO. It is a single international telephone number, possibly with an extension, that is the best number to use when trying to contact a person who can speak to issues involving registered items.

12) The fax number is provided by the SO. It is the best facsimile telephone number to use when trying to contact a person who can speak to issues involving registered items.

13) The comment is provided by the SO. It explains in natural language the relationship of the contact to the item or items submitted for registration. It may explain further the role of the contact.

3.6 Submission

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
submitTime	Datetime	Mandatory	RA	This Standard
submittingOrg	Organization	Mandatory	RA	This Standard
comment	CommentText		SO	SO
submitId	Submission	Mandatory	RA	RA

Semantic Rules

1. The Submission class represents a set of submissions made to a Registration Authority (RA) from a Submitting Organization (SO). The RA will keep a record of all submissions for an indefinite period of time determined by some higher authority.

2. Each instance of the Submission class represents a single submission. If the SO is not known to the RA, then the Submission must consist of a single request, RegisterSubmittingOrg, that asks the RA to certify the submitter as an SO for that Registry (see Section 8.19).

3. Each Submission instance is assigned a unique identifier by the Registration Authority (RA). This hidden attribute, herein called submitId, is said to be of type Submission. It is used to represent relationships of this instance with other information in the Registry. This identifier is locally determined by the RA, but will often be a timestamp.

4. A submission must be related to one or more contacts. Each contact is represented as an instance of the Contact class.

5. A submission must be related to one or more requests. Each request is represented as an instance of the Request class.

6. The submitTime attribute is provided by the RA. It identifies the time, with precision to the minute or finer, that the submission was first received. In many cases this attribute will just be a truncation of the more accurate timestamp used for the submission identifier.

7. The submittingOrg attribute identifies the Submitting Organization. If this is not a submission requesting certification of a company or an organization as a new SO, then the submitting organization must be previously known to the RA. The SO and RA will identify themselves to one another by using their respective orgURN's.

1. The comment attribute is provided by the SO. It may describe the purpose of the submission or give a human readable introduction to the containing requests.

8. Each Submission instance has a relationship with one or more Contact instances. The registry services defined herein may provide a method to access these contact(s) for each submission.

3.7 Request

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
requestNbr	SmallInt	Mandatory	RA	This Standard
requestCode	CodeText	Mandatory	SO	defnSource
contentXML	XMLtext	Mandatory	SO	This Standard
comment	CommentText		SO	SO

Semantic Rules

2. The Request class represents a set of requests made to a Registration Authority (RA) from Submitting Organizations (SO).

3. Each Request instance must be received by the RA as part of a submission package. Each Request instance has an implicit value, submitId, of type Submission that identifies the submission package.

4. Each submission may consist of multiple requests. The requestNbr attribute is a positive integer provided by the RA to distinguish among multiple requests in the same submission. If there are N requests in a submission package, then the requestNbr's will vary from 1 to N inclusive. The requestNbr may make it easier to locate the request in the contentXML.

5. The requestCode attribute is provided by the RA after receiving an XML Request element from an SO. Its value is the code defined by the RegistryRequest enumeration domain specified in Section 4.3. The list of valid requestCode's will likely expand in an upward compatible fashion as registry services expand.

6. The contentXML attribute is provided by the SO. It is the exact XML Request element provided by the SO as part of a submission package. The RA will keep a record of all such requests for an indefinite period of time determined by some higher authority.

7. The comment attribute is provided by the SO. It may provide further explanation about the purpose of the request.

8. Each Request instance has a relationship with zero or more Contact instances. The registry services defined herein may provide a method to access these contact(s) for each request.

3.8 AlternateName

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
altName	LongName	Mandatory	SO	This Standard
nameContext	CodeText	Mandatory	SO	defnSource
submittingOrg	Organization	Mandatory	RA	This Standard
language	LanguageCode		SO	W3C
encoding	CharEncoding		SO	W3C
comment	CommentText		SO	SO

Semantic Rules

1. The AlternateName class is the set of alternate names that are associated with registered objects. Each instance gives the alternate name, its language and encoding, and its context.

2. Each AlternateName instance has an implicit value, regEntryId, of type RegistryEntry that links this instance to its associated registry entry. The alternate name instance is dependent on that registry entry.

3. The altName attribute is provided by the SO.

4. The nameContext attribute is provided by the SO. It is a code value that identifies an item from the NameContext enumeration domain defined in Section 5.8. EDITOR's NOTE: How to represent Other values?

5. The submittingOrg attribute is provided by the RA. It identifies the organization that submits the alternate name for the parent registry entry; it may differ from the organization that owns the parent registry entry.

6. The triple (altName, nameContext, submittingOrg) uniquely determines an alternate name instance for each registry entry.

7. The language attribute is provided by the SO. It is of datatype LanguageCode, defined in Section 3.14. The default value is the default language of the submission, if known.

8. The encoding attribute is provided by the SO. It is of datatype CharEncoding, defined in Section 3.14. The default value is the default encoding of the submission, if known.

9. The comment is provided by the SO. It may give further details about how the name should be used.

10. All attribute values for an AlternateName instance are encoded in the encoding scheme identified by the encoding attribute.

3.9 ExternalData

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
dataName	ShortName	Mandatory	SO	SO
dataLocation	URL	Mandatory	SO	W3C
relatedRole	CodeText	Mandatory	SO	defnSource
mimeType	MIMEtype		SO	IANA
sizeBytes	Integer		SO	This Standard
comment	CommentText		SO	SO

Semantic Rules

1. The ExternalData class represents the set of non-registered objects that are related to registered objects. Each instance is a pairwise relationship between a single registry entry and a single external data item. A registry entry may map to many external data items.

2. Each instance of ExternalData depends upon a RegistryEntry instance. This dependency is represented by an implicit value, regEntryId, of type RegistryEntry.

3. The dataName attribute is provided by the SO. It is intended that this be the link name for the dataLocation if external data items are presented visually to a user. The dataName uniquely identifies an ExternalData instance for each registry entry.

4. The dataLocation is provided by the SO. This link is not under the control of the RA and it may point anywhere. The RA is under no obligation to ensure that the URL is a valid one.

5. The relatedRole is provided by the SO and takes its value from the RelatedRole enumeration domain defined in Section 5.11. EDITOR's NOTE: How to represent Other values?

6. The mimeType is provided by the SO. It identifies the MIME type (see Section 3.14) of the external data item. The RA is under no obligation to ensure that the declared mimeType is consistent with the actual file type of the file referenced by dataLocation. The default value is *text*.

7. The comment is provided by the SO. It may further explain the relationship between the external data instance and the registry entry it is linked to.

3.10 Description

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
language	LanguageCode	Mandatory	SO	W3C
submittingOrg	Organization	Mandatory	RA	This Standard
abstract	CommentText		SO	SO
keywordList	CommentText		SO	SO
fullDescription	DescriptionText	Mandatory	SO	SO
encoding	CharEncoding		SO	W3C

Semantic Rules

1. The Description class is a set of descriptions in various human languages that are associated with registered objects. Each instance is a single description associated with a single registry entry, with the human language and character encoding identified.

2. Each Description instance has an implicit value, regEntryId, of type RegistryEntry that links this instance to its associated registry entry. The description instance is dependent on that registry entry.

3. The language attribute is provided by the SO. It is of datatype LanguageCode, defined in Section 3.14. The default value is the default language of the submission.

4. The submittingOrg attribute is provided by the RA. It identifies the organization that submits the alternate name for the parent registry entry; it may differ from the organization that owns the parent registry entry.

5. The pair (language, submittingOrg) uniquely determines a Description instance for the parent registry entry.

6. The abstract attribute presents a short description of the parent registry entry. It may or may not be a direct translation of the shortDescription attribute of the parent registry entry into the language identified by the language attribute.

7. The keywordList attribute is a text string of words or short phrases, separated by a semicolon character and the space character, i.e. "; ". The words or phrases should be understandable in the language identified by the language attribute.

8. The fullDescription is provided by the SO. The intent is that each description is a direct translation of the fullDescription attribute of the associated registry entry into the language and encoding identified by the language and encoding attributes.

9. The encoding attribute is provided by the SO. It is of datatype CharEncoding, defined in Section 3.14 The default encoding is the default encoding of the submission.

3.11 Contribution

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
contributorName	LongName	Mandatory	SO	SO
contributorRole	LongName	Mandatory	SO	SO
contributorURL	URL		SO	W3C
roleCategory	CodeText		SO	defnSource
comment	CommentText		SO	SO

Semantic Rules

1. The Contribution class is the set of contributors that are associated with registered objects. Each instance gives the name and role of a contributor.

2. Each Contribution instance has an implicit value, regEntryId, of type RegistryEntry that links this instance to its associated registry entry. The contribution instance is dependent on that registry entry.

3. The contributorName attribute is provided by the SO. It identifies the name of a contributor to the creation or maintenance of the associated registered object.

4. The contributorRole attribute is provided by the SO. It identifies the role played by the contributorName in the creation or maintenance of the registered object.

5. The contributorName and contributorRole pair uniquely determine a contribution instance for each registry entry.

6. The roleCategory is provided by the SO. It is a code value that identifies an item from the RoleCategory enumeration domain defined in Section 5.12. EDITOR's NOTE: These roles are still undefined!

7. The comment is provided by the SO. It may give further details about the role played by the contributor.

3.12 Impact

Format

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
regEntryId	RegistryEntry	Mandatory	RA	This Standard
impactCode	CodeText	Mandatory	RA	This standard
comment	CommentText		RA	RA

Semantic Rules

1. The Impact class represents a many-to-many relationship between the Request class and the RegistryEntry class. Each Impact instance relates a single Request instance to a single RegistryEntry instance.

2. A request, e.g. RegisterSubmittingOrg, may not have an impact on any registry entry, whereas other requests, e.g. AddAssociation, will have an impact on both the given and associated items. Thus the need for a many-to-many mapping.

3. An Impact instance has implicit values for submitId and requestNbr that link this instance to its parent Request instance and its grandparent Submission instance. An impact instance is dependent upon its parent request.

4. The Impact class, together with its parent Submission and Request classes, will maintain a historical record of all SO-initiated modifications of a registry entry from first creation to final deletion.

5. Every registry entry must derive from at least one request, e.g. the request that created it. But it can be impacted by many subsequent requests, e.g. a sequence of updates over time.

6. The regEntryId attribute is provided by the RA. It identifies the impacted registry entry.

7. The impactCode attribute is provided by the RA. This code identifies one of the impacts specified by the RegistryImpact enumeration domain defined in Section 4.4.

8. The regEntryId and impactCode pair uniquely determine an Impact instance for each Request instance.

9. The comment is provided by the RA. It may explain some unusual operational consideration.

3.13 ClassificationScheme

Format

ClassificationScheme

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
schemeURN	URN	Mandatory	RA	RA
comment	CommentText	Mandatory	SO	SO
regObjectId	*RegisteredObject*	*Mandatory*	*RA*	*This Standard*
regEntryId	*RegistryEntry*	*Mandatory*	*RA*	*This Standard*

ClassificationNode

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
nodeId	ClassificationNode	Mandatory	RA	RA
itemValue	CodeText	Mandatory	SO	This Standard
itemName	LongName		SO	This Standard
parentId	ClassificationNode	Mandatory	RA	RA
levelNbr	Integer	Mandatory	RA	ThisStandard
comment	CommentText		SO	SO

ClassificationLevel

Attribute Name	Datatype	Presence	Supplies Value	Defines Meaning
levelCode	CodeText	Mandatory	SO	This Standard
levelName	LongName		SO	This Standard
levelNbr	Integer	Mandatory	RA	This Standard
comment	CommentText		SO	SO

Semantic Rules

1. The ClassificationScheme class identifies the classification schemes known to a Registry. Each instance of this class identifies one classification scheme. Each classification scheme is a registered object and thus has a hidden regObjectId attribute that is an object identifier of type RegisteredObject.

2. Since each registered object has an associated registry entry, the hidden regEntryId attribute maps the classification scheme to its registry entry. NOTE: A registered object may have multiple registry entries that point to it; this reference is to the *original* registry entry submitted with the registered object.

3. The schemeURN attribute is a dependent copy of the assignedURN attribute of the associated registry entry. This value is assigned by the RA and cannot be updated by any SO.

4. The comment attribute of the ClassificationScheme class is provided by the SO. It may expand upon the description provided by the shortDescription attribute of the associated registry entry.

5. The ClassificationNode class represents the classification hierarchy of a classification scheme. Each instance represents a node in that hierarchy.

6. Each instance of a ClassificationNode depends upon a ClassificationScheme. This dependency is represented by an implcit value of regObjectId that links each node to its parent classification scheme.

7. The nodeId attribute identifies each node of the classification hierarchy. The nodeId values are provided locally by the RA and are not visible to registry services; they are said to be of datatype ClassificationNode. The root of each classification hierarchy is the classification scheme itself. The root is a virtual node, not represented by a ClassificationNode instance, that is assumed to have a nodeId that maps to the parent classification scheme. This special nodeId may be referenced in registry services by the keyword ROOT.

8. The itemValue is provided by the SO. It identifies the value that will be referenced as the itemValue in a Classification.

9. The optional itemName, if present, is provided by the SO. It may be a longer descriptive name that more completely identifies the item. The itemName is not required in an XML Classification element (Section 6.3); instead, the itemValue is always required.

10. The parentId is supplied by the RA. It references the nodeId of the parent node in the classification hierarchy. If the node is at Level 1 in the classification hierarchy, then the ParentId is zero, to indicate that its parent node is the virtual root of the hierarchy. This special nodeId may be referenced in registry services by the keyword ROOT.

11. The comment in the ClassificationNode class is provided by the SO. It may further describe the node.

12. According to the mathematical defintion of classification scheme in Section 2.4, each node in the classification hierarchy is assigned a level. That level is represent by the levelNbr attribute in both the ClassificationNode class and the ClassificationLevel class.

13. The ClassificationLevel class identifies the levels of a classification scheme. Each instance represents a single level of the scheme. The number of levels must match the maximum level number of all nodes in the hierarchy. The levels are numbered consecutively as they would appear in the ClassificationLevel element of the ClassificationScheme DTD (Section 7.5).

14. Each instance of the ClassificationLevel class depends upon a ClassificationScheme instance. This dependency is represented by an implcit value of regObjectId that links each level to its parent classification scheme

15. The levelCode is provided by the SO. It identifies the value that will be referenced as the level Code in a classification (see Section 2.4.1).

16. The optional levelName, if present, is provided by the SO. It may be a longer descriptive name that more completely identifies the level. The levelName is not required in an XML Classification element (Section 6.3); instead, the levelCode is always used.

17. The comment in the ClassificationLevel class is provided by the SO. It may further describe the level.

3.14 Attribute type definitions

Some attribute values in the information model are references to class instances, some are codes that reference values of enumeration domains, and some are primitive types representable as character strings, numbers, dates, or dates and times. All codes that reference a value of an enumeration domain are of the type CodeText defined below.

The reference types are:

RegistryEntry
RegisteredObject
Organization
Contact
Submission
ClassificationNode

The following are base types used in this specification:

CodeText -- a character string consisting entirely of visible characters from an implied character set. The presence of non-visible characters, even blank spaces, is an error. The length of a CodeText string is between 1 and 32 characters, inclusive. In most cases, CodeText characters will be from the International Reference Version of ISO 646 so that they will be interpreted identically in all encodings. In XML environments, CodeText may not contain XML characters with special meaning. These include the ampersand (&), forward slash (/), special sequences like ++, etc. EDITOR's NOTE: Need to be more specific here!

ShortName -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of a ShortName string is between 1 and 50 characters, inclusive. The first and last characters must be visible characters that are not the space character.

LongName -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of a LongName string is between 1 and 150 characters, inclusive. The first and last characters must be visible characters that are not the space character.

EmailText -- a character string consisting entirely of visible characters from an implied character set. The presence of non-visible characters, even blank spaces, is an error. The length of an EmailText string is between 1 and 50 characters, inclusive.

TelephoneText -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of a TelephoneText string is between 1 and 50 characters, inclusive. The first and last characters must be visible characters that are not the space character.

AddrLineText -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of an AddrLineText string is between 1 and 50 characters, inclusive. The first and last characters must be visible characters that are not the space character.

CommentText -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces, tab characters, and return or line feed characters. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of a CommentText string is between 1 and 250 characters, inclusive. The first and last characters must be visible characters that are not the space character.

DescriptionText -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces, tab characters, and return or line feed characters. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype. The length of a DescriptionText string is between 1 and 5000 characters, inclusive. The first and last characters must be visible characters that are not the space character.

XMLtext -- a character string of indefinite length conforming to some XML document or type definition.

Date -- a value that represents a calendar date, constrained by the natural rules for dates using the Gregorian calendar. A Registry will be able to respond to queries involving minimal date arithmetic, e.g. finding all instances of an entity having dates for a given attribute that fall within a given range, or finding all instances having dates in the past 30 days, or finding all registry entries whose registration is scheduled to expire in the next 3 months, etc. More advanced date arithmetic or date manipulation is at the discretion of the Registry.

Date Literal -- a character string value that identifies a specific date. A date literal string is of the form YYYY-MM-DD where YYYY is an integer literal for the year, MM is an integer literal for the month of the year, and DD is an integer literal for the day of the month. Whenever a date value is presented to a user, or requested from a user, the date value is presented or transmitted as the equivalent date literal.

Datetime -- a value that represents a calendar date and a time within that date, with time precision to the minute, or finer. Unless otherwise indicated time is Universal Coordinated Time based on a 24-hour clock. A Registry has the capability to convert a Datetime type to a Date type, with the expected loss of precision. Any other datetime arithmetic or datetime manipulation is at the discretion of the Registry.

Datetime Literal -- a character string value that identifies a specific datetime. A datetime literal string is of the form YYYY-MM-DD HH:MM:SS where YYYY is an integer literal for the year, MM is an integer literal for the month of the year, DD is an integer literal for the day of the month, HH is an integer literal for the hour (assuming 24-hour clock), MM is an integer literal for the minute within the hour, and SS is an integer literal for the second within the minute. Whenever a datetime value is presented to a user, or requested from a user, the datetime value is presented or transmitted as the equivalent datetime literal.

Integer -- As used in this specification, a non-negative integer with value less than 2**32.

URN -- a character string that conforms to the format of a Uniform Resource Name (URN) as specified by IETF RFC 1241. The length of a URN string is less than or equal to 150 characters.
(See http://www.ietf.cnri.reston.va.us/rfc/rfc2141.txt?number=2141)

URL -- a character string that conforms to the format of a Uniform Resource Locator (URL) as specified by W3C. The length of a URL string is less than or equal to 150 characters.
(See http://www.w3.org/Addressing/URL/5_BNF.html)

FTP -- a character string that conforms to the format of a File Transfer Protocol (FTP) Uniform Resource Locator (URL) as specified by W3C. The default user name is "anonymous". The length of an FTP string is less than or equal to 150 characters. (See http://www.w3.org/Addressing/URL/5_BNF.html)

MIMEtype – a character string that identifies a MIME type, as listed in the official list of all MIME media-types assigned by the IANA (Internet Assigned Number Authority). NOTE: The slash character "/" is not legal in XML CDATA attributes, so the hyphen character "-" is used instead. The length of a MIMEtype string is less than or equal to 150 characters. (See ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types)

LanguageCode -- a character string that identifies a human language and a country where that language has evolved. In general, it is of the form "xx-CC", where xx is a two character code (lowercase) for a human language and CC is a two character country code (uppercase). Legal strings are specified by Language Identifier, definitions [33] through [38] in W3C XML 1.0. (http://www.w3.org/TR/REC-xml#sec-lang-tag).

CharEncoding -- a character string that identifies the encoding of a character set. It is specified by the encoding name (EncName) of an Encoding Declaration, definition [81] in W3C XML 1.0.
(http://www.w3.org/TR/REC-xml#charencoding).

4. General Enumeration Domains

4.1 DefinitionSource

Format

Code	Organization	Description
ebXML	ebXML	Owner of the ebXML Registry/Repository specification.
LTSC_LOM	IEEE Learning Technology Standards Committee - Learning Object Model	Owner of the IEEE LOM Registry specification.
IMS	IMS	Owner of the IMS Registry specification.
OASIS	Organization for the Advancement of Structured Information Standards	Owner of the OASIS Registry/Repository Technical Specification.
*****	Other	

Semantic Rules

1. The format for DefinitionSource identifies a potentially expanding list of organizations that may adopt the information model of this standard, but specify other or additional values for specific enumeration domains.

2. The defnSource attribute of the RegistryEntry class (Section 3.1) takes its value from the code column.

3. The symbol ***** for Code indicates that other values will be defined in future versions of this specification.

4.2 OrganizationRole

Format

Code	Name	Description
RA	Registration Authority	
RO	Responsible Organization	
SO	Submitting Organization	

Semantic Rules

1. The names specified in the second column are defined and explained by International Standard ISO 11179.

2. The orgRole attribute of the Contact class (Section 3.5) takes its value from the Code column.

4.3 RequestCode

Format

requestCode	Request Service Name
addAssoc	AddAssociation
addClassif	AddClassification
addAltName	AddAlternateName
addContrib	AddContribution
addExtData	AddExternalData
addDescrip	AddDescription
defClassSchm	DefineClassificationScheme
defRegPkg	DefineRegistryPackage
delAssoc	DeleteAssociation
delClassif	DeleteClassification
delAltName	DeleteAlternateName
delContrib	DeleteContribution
delExtData	DeleteExternalData
delDescrip	DeleteDescription
modClassif	ModifyClassificationScheme
modRegPkg	ModifyRegistryPackage
modRegEntry	ModifyRegistryEntry
regObj	RegisterObject
regSO	RegisterSubmittingOrganization
reaffRegObj	ReaffirmRegisteredObject
repRegObj	ReplaceRegisteredObject
supRegObj	SupercedeRegisteredObject
wdrRegObj	WithdrawRegisteredObject
*****	Others under development

Semantic Rules

1. Each requestCode identifies an OASIS registry services request.

2. The requestCode attribute of the Request class (Section 3.6) takes its value from the requestCode column.

3. The symbol ***** for itemValue indicates that other values are possible. The values added will always be done in an upward compatible fashion as registry services expand.

4.4 ImpactCode

Format

impactCode	Impact Name
AAS	Add Association
ACF	Add Classification
ACT	Add Contact
AAN	Add Alternate Name
ACB	Add Contribution
ARO	Add Registered Object
ARE	Add Registry Entry
AED	Add External Data
ASO	Add Submitting Organization
ADS	Add Description
DAS	Delete Association
DCF	Delete Classification
DCT	Delete Contact
DAN	Delete Alternate Name
DCB	Delete Contribution
DRO	Delete Registered Object
DRE	Delete Registry Entry
DED	Delete External Data
DSO	Delete Submitting Organization
DDS	Delete Description
UAS	Update Association
UCF	Update Classification
UCT	Update Contact
UAN	Update Alternate Name
UCB	Update Contribution
URO	Update Registered Object
URE	Update Registry Entry
UED	Update External Data
USO	Update Submitting Organization
UDS	Update Description

Semantic Rules

1. Each impactCode identifies an OASIS registry services impact on some class of the information model.

2. The impactCode attribute of the Impact class (Section 3.12) takes its value from the impactCode column.

3. These values are fixed unless the Information Model itself expands to include new features not representable in existing structures.

5. OASIS Enumeration Domains

5.1 ObjectType

Format

Source	Code	Name	Description
OASIS	defn	Definition	An XML or SGML definition document.
OASIS	inst	Instance	An XML or SGML instance document.
OASIS	rpkg	Registry Package	A registry package instance. Validates to RegistryPackage DTD.
OASIS	schm	Classification Scheme	A classification scheme instance. Validates to ClassificationScheme DTD.
OASIS	mime	MIME type	A MIME type as specified by IANA.
OASIS	othr	Other	A document with no specific object type.

Semantic Rules

1. The ObjectType enumeration domain provides the set of coded values for the objectType attribute of the RegistryEntry class (Section 3.1). Each code in the Code column represents the object type in the Name column.

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. The objectType attribute of E specifies the object type of R. The object types named above have the following meaning:

   Case:

   a) If the object type of R is *Definition*, then R is an XML or SGML definition document. The FileType enumeration domain in Section 5.2 identifies the valid XML and SGML definition file types.

   b) If the object type of R is *Instance*, then R is an XML or SGML document that validates to some XML or SGML definition. A registry entry for the definition document it validates to must be present in the same registry. Call that registry entry D. There must be a *Validates To* association instance in the registry with E as the given item and D as the associated item.

   c) Registry Package is a specialization of Instance. If the object type of R is *Registry Package*, then R is a registry package that validates to the RegistryPackage DTD defined in Section 7.6. E must have a *Validates To* association with a registry entry for that DTD.

   d) Classification Scheme is a specialization of Instance. If the object type of R is *Classification Scheme*, then R is a classification scheme that validates to the ClassificationScheme DTD defined in Section 7.5. E must have a *Validates To* association with a registry entry for that DTD.

   e) If the object type of R is *MIME type*, then R is a document that can be classified by one of the MIME types enumerated by the Internet Assigned Number Authority (IANA) as identified in Section 3.14. The specific MIME classification will be given by the fileType attribute of E.

   f) If the object type of R is *Other*, then R is of no specific type. Instead, the fileType of R may specify that R is an HTML or some other type of document.

3. More?

5.2 FileType

Format

Source	Code	Name	Description
OASIS	charEntSet	character-entity-set	
OASIS	rdfSchema	rdf-schema	This is a permitted fileType when the objectType is Definition.
OASIS	sgmlAttrib	sgml-attribute	
OASIS	sgmlAttSet	sgml-enumerated-attribute-set	
OASIS	sgmlAttVal	sgml-enumerated-attribute-value	
OASIS	sgmlDTD	sgml-dtd	This is a permitted fileType when the objectType is Definition.
OASIS	sgmlElement	sgml-element	This is a permitted fileType when the objectType is Definition.
OASIS	sgmlEntity	sgml-parameter-entity	This is a permitted fileType when the objectType is Definition.
OASIS	soxSchema	sox-schema	This is a permitted fileType when the objectType is Definition.
OASIS	xdrSchema	xdr-schema	This is a permitted fileType when the objectType is Definition.
OASIS	xmlAttrib	xml-attribute	
OASIS	xmlAttSet	xml-enumerated-attribute-set	
OASIS	xmlAttVal	xml-enumerated-attribute-value	
OASIS	xmlDTD	xml-dtd	This is a permitted fileType when the objectType is Definition.
OASIS	xmlElement	xml-element	This is a permitted fileType when the objectType is Definition.
OASIS	xmlEntity	xml-parameter-entity	This is a permitted fileType when the objectType is Definition.
OASIS	xmlSchema	xml-schema	This is a permitted fileType when the objectType is Definition.
OASIS	xml	XML text document	This is a permitted fileType when the objectType is Instance.
OASIS	zip	zip	This is a permitted fileType when the objectType is Other.
OASIS	*****	MIME type declaration	Any valid mime type specified by IANA. With hyphen (-) as separator in place of slash (/). This is a permitted fileType when the objectType is MIME.
OASIS	sgml	SGML text document	This is a permitted fileType when the objectType is Instance.
OASIS	html-1	HTML Level 1	This is a permitted fileType when the objectType is Instance. The implementation will create an appropriate ValidatesTo association.
OASIS	html-2	HTML Level 2	This is a permitted fileType when the objectType is Instance. The implementation will create an appropriate ValidatesTo association.
OASIS	html-3	HTML Level 3	This is a permitted fileType when the objectType is Instance. The implementation will create an appropriate ValidatesTo association.
OASIS	html-4	HTML Level 4	This is a permitted fileType when the objectType is Instance. The implementation will

			create an appropriate ValidatesTo association.
OASIS	html-iso	ISO/IEC 15445:2000	This is a permitted fileType when the objectType is Instance. The implementation will create an appropriate ValidatesTo association.
OASIS	xhtml	W3C Extensible HyperText Markup Language (XHTML) http://www.w3.org/TR/xhtml1).	This is a permitted fileType when the objectType is Instance. The implementation will create an appropriate ValidatesTo association.
OASIS	*****	User provided file type	Any text supplied by the user that is of type CodeText as defined in Section 3.14.

Semantic Rules

1. The FileType enumeration domain provides a set of coded values for the fileType attribute of the RegistryEntry class (Section 3.1). The fileType attribute may be dependent on the value declared for the objectType attribute. Four asterisks indicate any valid string of type CodeText as defined in Section 3.14.

2. More?

5.3 RegistrationStatus

Source	Code	Name	Description
OASIS	sub	Submitted	Result of the RegisterObject request.
OASIS	urw	Under Review	Submitted object is under review by panel.
OASIS	reg	Registered	Object is registered.
OASIS	sup	Superceded	Object is superceded by another object.
OASIS	dep	Deprecated	Object is deprecated.
OASIS	rep	Replaced	Object is replaced by another object.
OASIS	wth	Withdrawn	Object is withdrawn.
OASIS	exp	Expired	Registration of object has expired.

Semantic Rules

1. The RegistrationStatus enumeration domain provides the set of coded values for the registrationStatus attribute of the RegistryEntry class (Section 3.1).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. The registrationStatus attribute of E specifies the registration status of R. The registration status alternatives named above have the following meaning:

   Case:

   a) If the registration status of R is *Submitted*, then neither E nor R are visible to outside users. E and its associated metadata are visible only to the SO and the RA.

   b) If the registration status of R is *Under Review*, then both E and R are visible to a select group of reviewers. The structure and design of that review process is beyond the scope of this specification. E and R may not be available to other users during this review process. At some point the review process is complete and R receives a new registration status. If Registry conformance is *with Validation* (see Section 10.4), then this review process may include validating R to its specification.

   c) If the registration status of R is *Registered*, then E and R are available to the general public subject to other conditions of this specification.

   d) If the registration status of R is *Superceded*, then the RA has received and acted upon a request from the SO that submitted E and R to supercede them with a new registered object and a new registry entry. The original registered object remains available for continued use. The objectURL attribute of E continues to reference R, and a new registry entry E' will reference the new object R'. In addition there will be an *Is Supercede By* association instance with E and the given item and E' as the associated item. See the SupercedeRegisteredObject request (Section 8.22) for details.

   e) If the registration status of R is *Replaced*, then the RA has received and acted upon a request from the SO that submitted E and R to replace R with a new registered object. The original registered object R is no longer available, although its metadata E will remain in the registry for an indefinite period of time. The objectURL attribute of E will now reference the new object R', and a new registry entry E' will also point to R'. In addition there will be an *Is Replaced By* association instance with E and the given item and E' as the associated item. See the ReplaceRegisteredObject request (Section 8.21) for details.

   f) If the registration status of R is *Deprecated*, then the SO has indicated that R will soon be replaced or withdrawn. No other metadata changes. Some organizations have a policy that an object must be deprecated for a certain period of time before it can be withdrawn or replaced. Enforcement of such a policy is beyond the scope of this specification. EDITOR's NOTE: There is no registry service yet to achieve this registration status.

g) If the registration status of R is *Withdrawn*, then the RA has received and acted upon a request from the SO that submitted R to withdraw it (Section 8.23). The original registered object is no longer available, although its metadata will remain in the registry for an indefinite period of time. The objectURL attribute of E will now have no value.

h) If the registration status of R is *Expired*, then the expirationDate attribute of E has passed with no reaffirmation action (Section 8.20) from the SO. The original registered object R may or may not still be available, although the registry entry E and other relevant metadata will remain in the registry for an indefinite period of time.

3. Let R be any registered object and let SO be the *owner* of R as defined in Section 3.1. The registration status of R may only be modified by SO.

5.4 Stability

Format

Source	Code	Name	Description
OASIS	comp	Compatible	Registered object may be replaced only by an upward compatible object.
OASIS	dynm	Dynamic	Registered object may change at any time.
OASIS	stat	Static	Registered object will not change before expiration.

Semantic Rules

1. The Stability enumeration domain provides the set of coded values for the stability attribute of the RegistryEntry class (Section 3.1).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. The stability attribute of E specifies the stability of R. The stability alternatives named above have the following meaning:

   Case:

   a) If the stability of R is *Static*, then the SO declares that R will not be replaced or withdrawn before the expiration date identified by the expirationDate attribute of E. It is the responsibility of the RA to ensure as best it can that this declaration is not violated.

   b) If the stability of R is *Dynamic*, then the SO declares that R may change without notice, possibly in incompatible ways.

   c) If the stability of R is *Compatible*, then the SO declares that R will not be replaced with an *incompatible* object before the expiration date identified by the expirationDate attribute of E.

3. There is no default for stability. Each SO is required to declare an appropriate stability categorization for each submission.

4. Let R and R' be XML or SGML definition objects (Section 5.1 and Section 5.2). R' is said to be compatible with R if an XML document that validates to R also validates to R'.

5. Let R and R' be classification scheme instances (Section 7.5). R' is said to be compatible with R if every level of R is a level of R' and if every node of R is a node of R' at the same level.

6. Let R and R' be registry package instances (Section 7.6). R' is said to be compatible with R if every package member E of R is a package member of R' and if the stability attribute of every such E is not *Dynamic*.

7. Let R and R' be abstract data types. R' is said to be compatible if every attribute of R is an attribute of R' and if every method of R is a method of R' having the same effect.

8. Let R be any registered object and let SO be the *owner* of R as defined in Section 3.1. The stability of R may only be modified by SO.

9. EDITOR's NOTE: The notion of *compatibility* still needs to be examined for all other instance, html, mime, and other object types!

5.5 FeeStatus

Format

Source	Code	Name	Description
OASIS	debit	Debit	The registered object is available on-line, but requires payment of a fee before access is granted.
OASIS	free	Free	The registered object is freely available with no password or payment required.
OASIS	pswd	Password	The registered object is free, but password is required for retrieval.

Semantic Rules

1. The FeeStatus enumeration domain provides the set of coded values for the feeStatus attribute of the RegistryEntry class (Section 3.1).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. The feeStatus attribute of E specifies the fee status of R. The fee status alternatives named above have the following meaning:

   Case:

   a) If the feeStatus of R is *Free*, then the URL identified by the objectURL attribute of E follows an anonymous protocol that allows retrieval of R with no password or payment obstructions.

   b) If the feeStatus of R is *Password*, then the URL identified by the objectURL attribute of E may require a password, but access to R is free. The procedure for obtaining a password is not part of this specification; it may include agreeing to property rights specified by the submitting organization.

   c) If the feeStatus of R is *Debit*, then the URL identified by the objectURL attribute of E may require a payment before it is activated. The procedure for handling the payment is not part of this specification.

3. The default FeeStatus is *Free*.

4. Let R be any registered object and let SO be the *owner* of R as defined in Section 3.1. The feeStatus of R may only be modified by SO.

5.6 PropertyRights

Format

Source	Code	Name	Description
OASIS	xxx	XXX	TO BE DETERMINED
OASIS	none	None	There are no property rights for this registered object. It can be used by any user as desired.

Semantic Rules

1. The PropertyRights enumeration domain provides the set of coded values for the propertyRights attribute of the RegistryEntry class (Section 3.1).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. The propertyRights attribute of E specifies the property rights of R. The property rights alternatives named above have the following meaning:

   Case:

   a) [NOT FINISHED]

   b) [NOT FINISHED]

   c) If the property rights of R is *None*, then there are no restrictions on the use of R by any user of registry services.

3. The default PropertyRights is *None*. [NOTE: Some other default may be more appropriate!]

4. Let R be any registered object and let SO be the *owner* of R as defined in Section 3.1. The property rights of R may only be modified by SO.

5.7 AssociationRole

Format

Source	Code	Name	Description
OASIS	val	Validates To	The given item validates to the specification provided by the associated item.
OASIS	req	Requires	The given item requires the presence of the associated item before it can be processed or used.
OASIS	cnt	Contains	The given item is a package that contains the associated item.
OASIS	sup	Is Superceded By	The given item supercedes associated item, but associated item remains intact.
OASIS	rep	Is Replaced By	The given item replaces associated item, and associated item is no longer available.
OASIS	rel	Is Related To	The given item is related to and provides supplemental information for the associated item.

Semantic Rules

1. The AssociationRole enumeration domain provides the set of coded values for the associationRole attribute of the Association class (Section 3.2).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let X be an Association instance linked to E with E as the given item of X and E' as the associated item of X. Let R and R' be the registered objects referenced by the objectURL attributes of E and E', respectively. The associationRole attribute of X specifies the association role of R with respect to R'. The association role alternatives named above have the following meaning:

   Case:

   a) If the association role of R with respect to R' is *Validates To*, then the SO declares that R validates to the definition of R'. The RA is under no obligation to check whether this declaration is or remains true.

   b) If the association role of R with respect to R' is *Requires*, then the SO declares that use of R requires the presence of R'.

   c) If the association role of R with respect to R' is *Contains*, then the SO declares that R is a registry package that contains E' as a package member.

   d) If the association role of R with respect to R' is *Is Superceded By*, then the SO declares that R is superceded by R'. The effect is that E remains in the registry, its registrationStatus attribute now indicates *Superceded*, and its objectURL attribute still references R. For processing details see Section 8.22.

   e) If the association role of R with respect to R' is *Is Replaced By*, then the SO declares that R is replaced by R'. The effect is that E remains in the Registry and its registrationStatus attribute now indicates *Replaced*, but the objectURL attributes of both E and E' now reference R'. For processing details see Section 8.21.

   f) If the association role of R with respect to R' is *Is Related To*, then the SO declares that R is related to R'.

3. There is no default AssociationRole; an explicit role must be declared for each association.

5.8 NameContext

Format

Source	Code	Name	Description
OASIS	cpp	C++ Name	For Programming Language C++ usage.
OASIS	code	Code Name	A name consisting of visible characters only, i.e. no embedded space characters.
OASIS	java	Java Name	For Programming Language Java usage.
OASIS	lang	Language Name	For use in a specified human language.
OASIS	long	Long Name	1 to 150 characters
OASIS	short	Short Name	1 to 8 characters
OASIS	sql	SQL Name	For Database Language SQL usage.
OASIS	urn	Uniform Resource Name	Name conforms to W3C URN syntax

Semantic Rules

1. The NameContext enumeration domain provides the set of coded values for the nameContext attribute of the AlternateName class (Section 3.8).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. Let X be an AlternateName instance linked to E and let AN be the name specified by the altName attribute of X. The nameContext attribute of X specifies the context of AN as an alternate name for R. The NameContext alternatives named above have the following meaning:

   Case:

   a) If the name context of AN is *Uniform Resource Name*, then AN satisfies the syntactic requirements of a W3C URN. The intent is that this context be used when it is discovered that R has been registered in some other conforming Registry with an assignedURN, equal to AN, that differs from the one assigned by this Registry.

   b) If the name context of AN is *Code Name*, then AN satisfies the syntactic constraints of the CodeText datatype defined in Section 3.14. The intent is that AN become the name for R in programming environments with strict limitations on name format and name length.

   c) Short Name is a specialization of Code Name, restricting the length of AN to 8 characters.

   d) C++ Name is a specialization of Code Name, restricting the length and format of AN to a name acceptable for use in C++ programming environments.

   e) Java Name is a specialization of Code Name, restricting the length and format of AN to a name acceptable for use in Java programming environments.

   f) SQL Name is a specialization of Code Name, restricting the length and format of AN to a name acceptable for use in SQL programming environments.

   g) If the name context of AN is *Long Name*, then AN satisfies the syntactic constraints of the LongName datatype defined in Section 3.14. The intent is that this context be declared when AN is just another alternate name for R with no special usage expectations.

   h) If the name context of AN is *Language Name*, then AN is an alternate name for R expressed in the explicit human language identified by the language attribute of X.

3. The default NameContext is *Long Name*.

5.9 ContactAvailability

Format

Source	Code	Name	Description
OASIS	pri	Private	Contact available only to SO and RA.
OASIS	pro	Protected	Contact available only to RA's.
OASIS	pub	Public	Contact available to all users of registry.

Semantic Rules

1. The ContactAvailability enumeration domain provides the set of coded values for the availability attribute of the Contact class (Section 3.5).

2. Let X be a Contact instance. In order of precedence, X will be associated with a Request instance, a Submission instance, or an Organization instance. The availability attribute of X specifies the availability of X with respect to answering questions related to the Request, the Submission, or the Organization (see Figure 4 in Section 2.7). The ContactAvailability alternatives named above have the following meaning:

   Case:

   a) If the availability of X is *Private*, then the RA will not reveal X to any other organization.

   b) If the availability of X is *Protected*, then the RA may reveal X to other legitimate authorities, where legitimate authorities for Oasis implementations are other Oasis approved Registration Authorities, or individuals or organizations involved in the review or approval process for registry entries or registered objects.

   c) If the availability of X is *Public*, then the RA may reveal X to any user of the registry maintained by the RA.

3. The default ContactAvailability is *Public*.

5.10 ContactRole

Format

Source	Code	Name	Description
OASIS	admn	Administrative	Contact addresses administrative issues.
OASIS	all	All Issues	Contact addresses all issues.
OASIS	tech	Technical	Contact addresses technical issues.

Semantic Rules

1. The ContactRole enumeration domain provides the set of coded values for the contactRole attribute of the Contact class (Section 3.5).

2. Let X be a Contact instance. In order of precedence, X will be associated with a Request instance, a Submission instance, or an Organization instance. The contactRole attribute of X specifies the role of X with respect to answering administrative questions related to the Request, the Submission, or the Organization (see Figure 4 in Section 2.7). The ContactRole alternatives named above have the following meaning:

   Case:

   a) If the role of X is *Administrative*, then the contact is prepared to address administrative questions or issues related to the associated request, submission, or organization.

   b) If the role of X is *Technical*, then the contact is prepared to address technical questions or issues related to the associated request, submission, or organization.

   c) If the role of X is *All Issues*, then the contact is prepared to address all questions or issues related to the associated request, submission, or organization.

3. The default ContactRole is *All Issues*.

5.11 RelatedRole

Format

Source	Code	Name	Description
OASIS	Changelog	changelog	
OASIS	CvrLetter	cover-letter	
OASIS	DistribHP	distribution-home-page	
OASIS	DocSet	documentation-set	
OASIS	DocSetInfo	documentation-set-information	
OASIS	DSSLSS	dssl-style-sheet	
OASIS	DSSLSSinfo	dssl-style-sheet-information	
OASIS	EmailInfo	email-discussion-list-information	
OASIS	Example	example	
OASIS	ExpSet	example-set	
OASIS	ExpSetInfo	example-set-information	
OASIS	FAQ	faq	
OASIS	Other	other	
OASIS	PublicText	public-text	
OASIS	ReadMe	readme	
OASIS	RefMan	reference-manual	
OASIS	RegInfo	registration-information	
OASIS	RelDataGrp	related-data-group	
OASIS	SchemaHP	schema-home-page	
OASIS	SGMLdeclar	sgml-declaration	
OASIS	SGMLopnCat	sgml-open-catalogue	
OASIS	StyleSinfo	style-sheet-information	
OASIS	ToolInfo	tool-information	
OASIS	UserGuide	user-guide	
OASIS	WhitePaper	white-paper	
OASIS	XSLSS	xsl-style-sheet	
OASIS	XSLSSinfo	xsl-style-sheet-information	

Semantic Rules

1. The RelatedRole enumeration domain provides the set of coded values for the relatedRole attribute of the ExternalData class (Section 3.9).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. Let X be an ExternalData instance linked to E. The relatedRole attribute of X specifies the relationship role of the ExternalData instance to R. Each of these roles is self-explained by the Name and/or Description given above.

3. EDITOR's NOTE: It seems that many of the above XML and SGML-related items could be deleted, since style sheets and SGML declarations would likely be registered objects, not external data references. Also, ChangeLog replicates the intent of the Impact class, and Cover Letter has very little meaning.

4. More? EDITOR's NOTE: Is this an open or closed list?

5.12 RoleCategory

Format

Source	Code	Name	Description
OASIS	creation	Creation	Authors, Co-authors, or others directly responsible for creation of the registered object.
OASIS	approval	Approval	Individuals or organizations responsible for approval of the registered object.
OASIS	support	Support	Administrative or technical staff or functions helpful to development of the registered object.
OASIS	review	Review	Individuals or functions involved in critical review of the registered object but not in its approval.
OASIS	other	Other	The contribution has no specific categorization.

Semantic Rules

1. The RoleCategory enumeration domain provides the set of coded values for the roleCategory attribute of the Contribution class (Section 3.11).

2. Let E be a registry entry whose defnSource attribute identifies OASIS as the defining organization. Let R be the registered object referenced by the objectURL attribute of E. Let X be a Contribution instance linked to E. The roleCategory attribute of X specifies the role category of the contribution to R. The RoleCategory alternatives named above have the following meaning:

   Case:

   a) If the role category of X to R is *Creation*, then X had an important role in the creation of R. This may include authors, co-authors, editors, co-editors, producers, members of specification development committees, etc.

   b) If the role category of X to R is *Approval*, then X had an important role in the approval process that led to registration of R. This may include voting members of the group that votes a specification up or down. The contributorRole attribute of X may distinguish approval, disapproval, and abstention positions of voting members since those positions could have important consequences for implementation of a specification.

   c) If the role category of X to R is *Support*, then X had a support role in the creation of R. This may include administrative support staff, illustrators, technicians, etc.

   d) If the role category of X to R is *Review*, then X had an important role in reviewing the content of R. This may include independent technical reviews, quality reviews, usability reviews, conformance reviews, etc. In general, such reviewers do not actually vote on approval or disapproval of R's acceptance.

   e) If the role category of X to R is *Other*, then X has no specific categorization. In this case, the contributorRole attribute identifies the specific role of X to R, with possible use of the comment attribute for additional explanation.

3. The default RoleCategory is *Creation*.

6. XML Representations

The OASIS Information Model specifies the logical structures of an OASIS Registry/Repository in terms of a UML specified data model. The XML definitions specified in this section, and the semantic rules associated with them, are defined in terms of the classes and attributes of that data model. The attributes of those classes are represented either as XML elements or as XML attributes, whichever seems most appropriate.

These ELEMENT's and DTD's are intended to be the basis of communication between a Submitting Organization and an OASIS conformant registry/repository, or between two conforming registry/repositories. The slightly different versions of each element provide an appropriate structure for each purpose.

There are at most three versions of each important class instance, one for use in a submission of that instance from a submitting organization to a registry, one for representation of that instance in a nested XML document for registry-to-registry exchange, and one for representation of a set of class instances in a flat file. The flat file representations generally require identifiers for each instance and pointers among instances to represent their relationships, whereas the submission and nested forms represent the relationships in the structured nesting of elements.

NOTE: The authors of this document do not feel confident about when it is best to represent something as an XML element and when it is better to represent it as an XML attribute. It will be straight-forward to do the right thing after some good discussions. EDITOR's NOTE: Act on this in December or delete!!

6.1 RegistryEntry Elements

Purpose

To define or represent a registry entry.

Definition

```
<!ELEMENT RegistryEntrySubmit
      ( CommonName, ShortDescription )>
<!ATTLIST RegistryEntrySubmit
      suggestedURN      CDATA             #IMPLIED
      version              CDATA             #IMPLIED
      objectURL            CDATA             #REQUIRED
      defnSource        (%defnSourceList;)   #REQUIRED
      objectType        (%objectTypeList;)   #REQUIRED
      fileType          CDATA             #IMPLIED
      stability         (%stabilityList;)    #REQUIRED
      feeStatus         (%feeStatusList;)    "free"
      propertyRights    (%propRightsList;)   "none"
      expirationDate       CDATA             #IMPLIED
      respOrgURN           CDATA             #IMPLIED
      regEntryId              ID             #IMPLIED  >

<!ELEMENT CommonName  (#PCDATA )>

<!ELEMENT ShortDescription  (#PCDATA )>

<!ELEMENT RegistryEntryInstance
      ( CommonName, ShortDescription )>
<!ATTLIST RegistryEntryInstance
      regEntryId           ID                #REQUIRED
      assignedURN          CDATA             #REQUIRED
      version              CDATA             #IMPLIED
      objectURL            CDATA             #IMPLIED
      defnSource        (%defnSourceList;)   #REQUIRED
      objectType        (%objectTypeList;)   #REQUIRED
      fileType          CDATA             #IMPLIED
      registrationStatus (%regStatusList;)   #REQUIRED
      statusChgDate        CDATA             #REQUIRED
      stability         (%stabilityList;)    #REQUIRED
      feeStatus         (%feeStatusList;)    #REQUIRED
      propertyRights    (%propRightsList;)   #REQUIRED
      expirationDate       CDATA             #REQUIRED
      submitOrgURN         CDATA             #REQUIRED
      submitOrgREF         IDREF             #IMPLIED
      respOrgURN           CDATA             #IMPLIED
      respOrgREF           IDREF             #IMPLIED  >
```

Semantic Rules

1. A submission of a proposed new registry entry from a Submitting Organization (SO) to a Registry uses the RegistryEntrySubmit element. An exchange or representation of an existing registry entry uses the RegistryEntryInstance element. The RegistryEntry class definition is in Section 3.1. All XML ENTITY definitions are in Section 6.18.

2. The suggestedURN attribute in RegistryEntrySubmit is only a suggestion by the SO as to what the assignURN attribute of a registry entry should be. The final determination is made by the RA.

3. The expirationDate attribute in RegistryEntrySubmit is only a suggestion by the SO. The RegistrationAuthority decides the expirationDate based on Registry policy. The expirationDate is represented by the Date Literal datatype (see Section 3.14).

4. The objectURL attribute in RegistryEntrySubmit may be a URL or FTP reference to a remote file, or it may be a local file reference to a file provided with the same submission.

5. Depending on the value of the objectType, the fileType may sometimes be required. See the Semantic Rules for the FileType enumeration domain in Section 5.2. If it is required, its value may come from the fileTypeList entity specified in Section 6.18.

6. If the registry entry describes a registered object that has been withdrawn, then the objectURL is not required in the RegistryEntryInstance element. In all other cases, objectURL is required.

7. The submitOrgURN attribute represents the submittingOrg attribute of the RegistryEntry class. It is a URN that identifies an organization known to the RA and whose hasSOstatus attribute is true.

8. The respOrgURN attribute represents the responsibleOrg attribute of the RegistryEntry class. If present, it is a URN that identifies an Organization instance known to the RA. If it is not the SO, then its hasROstatus attribute must be true.

9. The submitOrgREF and responsibleOrgREF attributes in RegistryEntryIinstance represent the submittingOrg and responsible Org attributes of the RegistryEntry class. If present, each must identify an Organization element in the same XML document.

10. The regEntryId attribute corresponds to the hidden regEntryId attribute of the RegistryEntry class. It need be unique only within a given XML document.

6.2 Association Elements

Purpose

To define or represent an association instance.

Definition

```
<!ELEMENT Association ( Comment? )>
<!ATTLIST Association
      assocRole      (%assocRoleList;)    #REQUIRED
      assocItemURN         CDATA          #IMPLIED
      assocItemREF         IDREF          #IMPLIED  >

<!ELEMENT AssociationFlat  ( Comment? )>
<!ATTLIST AssociationFlat
      givenItemURN         CDATA          #REQUIRED
      givenItemREF         IDREF          #IMPLIED
      assocRole      (%assocRoleList;)    #REQUIRED
      assocItemURN         CDATA          #REQUIRED
      assocItemREF         IDREF          #IMPLIED  >

<!ELEMENT Comment  (#PCDATA )>
```

Semantic Rules

1. A submission of a proposed new association instance from a Submitting Organization (SO) to a Registry uses the Association element. An exchange or representation of an existing association instance in a nested format, where the given item instance is known, uses the Association element. An exchange or representation of an existing association instance as a flat file uses the Association Flat element. The Association class definition is in Section 3.2 and all XML ENTITY definitions are in Section 6.18.

2. In the Association element, one of the attributes, either assocItemURN or assocItemREF, is required.

3. The assocRole attribute represents the associationRole attribute of the Association class.

4. The assocItemURN and assocItemREF attributes represent the assocItem attribute of the Association class.

5. The givenItemURN and givenItemREF attributes represent the givenItem attribute of the Association class.

6. The givenItemREF attribute, if present, must identify a RegistryEntry element in the same XML document.

7. The assocItemREF attribute, if present, must identify a RegistryEntry element in the same XML document.

6.3 Classification Elements

Purpose

To define or represent a classification instance.

Definition

```
<!ELEMENT Classification  ( LevelValuePair+, Comment? )>
<!ATTLIST Classification
      schemeURN      CDATA    #REQUIRED
      schemeName     CDATA    #IMPLIED
      submitOrgURN   CDATA    #IMPLIED
      submitOrgREF   IDREF    #IMPLIED >

<!ELEMENT LevelValuePair ( Comment? )>
<!ATTLIST LevelValuePair
      levelCode      CDATA    "leaf"
      itemValue      CDATA    #REQUIRED
      levelNbr       CDATA    #IMPLIED
      itemName       CDATA    #IMPLIED   >


<!ELEMENT ClassificationFlat  ( Comment? )>
<!ATTLIST ClassificationFlat
      regEntryId     IDREF    #REQUIRED
      regEntryURN    CDATA    #IMPLIED
      schemeURN      CDATA    #REQUIRED
      schemeId       IDREF    #IMPLIED
      levelCode      CDATA    #REQUIRED
      itemValue      CDATA    #REQUIRED
      itemName       CDATA    #IMPLIED
      submitOrgURN   CDATA    #IMPLIED
      submitOrgREF   IDREF    #IMPLIED >
```

Semantic Rules

1. A submission of a proposed new classification from a Submitting Organization (SO) to a Registry uses the Classification element. An exchange or representation of an existing classification in a nested format, where the given item instance is known, uses the Classification element. An exchange or representation of an existing classification instance as a flat file uses the ClassificationFlat element. The Classification class definition is in Section 3.3.

2. The schemeName , if present, matches the commonName attribute of the registry entry identified by schemeURN.

3. The levelNbr attribute, if present, must match the levelNbr of the level identified by levelCode.

4. The itemName attribute, if present, must match the itemName of the item identified by itemValue.

5. The regEntryId attribute corresponds to the regEntryId attribute of the Classification class in the information model. It must identify a RegistryEntry element in the same XML document.

6. The regEntryURN attribute, if present, must match the assignedURN of the registry entry identified by regEntryId.

6.4 ExternalData Elements

Purpose

To define or represent an external data instance.

Definition

```
<!ELEMENT ExternalData  (DataName , Comment? )>
<!ATTLIST ExternalData
      relatedRole (%relatedRoleList;)  #REQUIRED
      dataLocation      CDATA          #REQUIRED
      mimeType          CDATA          "text" >

<!ELEMENT DataName  (#PCDATA )>

<!ELEMENT ExternalDataFlat  (DataName , Comment? )>
<!ATTLIST ExternalDataFlat
      regEntryId        IDREF          #REQUIRED
      regEntryURN       CDATA          #IMPLIED
      relatedRole (%relatedRoleList;)  #REQUIRED
      dataLocation      CDATA          #REQUIRED
      mimeType          CDATA          #REQUIRED >
```

Semantic Rules

1. A submission of a proposed new external data instance from a Submitting Organization (SO) to a Registry uses the ExternalData element. An exchange or representation of an existing external data instance in a nested format, where the given item instance is known, uses the ExternalData element. An exchange or representation of an existing association instance as a flat file uses the ExternalDataFlat element. The ExternalData class definition is in Section 3.9 and all XML ENTITY definitions are in Section 6.18.

2. The regEntryId attribute corresponds to the implicit regEntryId attribute of the ExternalData class in the information model. It must identify a RegistryEntry element in the containing XML document.

3. The regEntryURN attribute, if present, must match the assignedURN of the registry entry identified by regEntryId.

4. A DataName element must have at least 1 visible character, and must uniquely identify that element within its containing XML document.

5. EDITOR's NOTE: Is the RelatedRole enumeration domain an open list or a closed list? If open, then the relatedRoleList entity must be replaced above.

6.5 Organization Elements

Purpose

To define or represent an organization instance.

Definition

```
<!ELEMENT OrganizationSubmit  ( Comment? )>
<!ATTLIST OrganizationSubmit
      orgFullName       CDATA    #REQUIRED
      commonName        CDATA    #IMPLIED
      orgURL            CDATA    #IMPLIED
      parentOrgURN      CDATA    #IMPLIED
      addrLine1         CDATA    #IMPLIED
      addrLine2         CDATA    #IMPLIED
      addrLine3         CDATA    #IMPLIED
      city              CDATA    #IMPLIED
      stateProv         CDATA    #IMPLIED
      postalCode        CDATA    #IMPLIED
      country           CDATA    #REQUIRED
      email             CDATA    #IMPLIED
      telephone         CDATA    #IMPLIED
      fax               CDATA    #IMPLIED  >

<!ELEMENT OrganizationInstance  ( Comment? )>
<!ATTLIST OrganizationInstance
      orgId             ID       #REQUIRED
      orgURN            CDATA    #REQUIRED
      orgFullName       CDATA    #IMPLIED
      commonName        CDATA    #REQUIRED
      hasSOstatus       CDATA    #IMPLIED
      hasROstatus       CDATA    #IMPLIED
      hasRAstatus       CDATA    #IMPLIED
      orgURL            CDATA    #IMPLIED
      parentOrgURN      CDATA    #IMPLIED
      parentOrgId       IDREF    #IMPLIED
      addrLine1         CDATA    #IMPLIED
      addrLine2         CDATA    #IMPLIED
      addrLine3         CDATA    #IMPLIED
      city              CDATA    #IMPLIED
      stateProv         CDATA    #IMPLIED
      postalCode        CDATA    #IMPLIED
      country           CDATA    #REQUIRED
      email             CDATA    #IMPLIED
      telephone         CDATA    #IMPLIED
      fax               CDATA    #IMPLIED  >
```

Semantic Rules

1. A submission for registration of an organization as a Submitting Organization (SO) to a Registry uses the OrganizationSubmit element. An exchange or representation of an organization already recognized by the Registration Authority (RA) uses the OrganizationInstance element. Organization class definition is in Section 3.4.

2. The parentOrgURN and parentOrgId attributes represent the parentOrg attribute of the Organization class.

3. The parentOrgURN attribute of the OrganizationSubmit element, if present, must identify the orgURN of an organization already known to the RA.

4. The parentOrgId attribute of the OrganizationInstance, if present, must identify an OrganizationInstance element in the containing XML document.

5. The orgId attribute corresponds to the orgId attribute of the Organization class in the information model. It need be unique only within a given XML document.

6.6 Contact Elements

Purpose

To define or represent a contact instance.

Definition

```
<!ELEMENT Contact  (Comment? )>
<!ATTLIST Contact
      contactName       CDATA             #REQUIRED
      email             CDATA             #REQUIRED
      telephone         CDATA             #IMPLIED
      fax               CDATA             #IMPLIED
      orgURN            CDATA             #IMPLIED
      orgRole        (%orgRoleList;)      "SO"
      availability   (%contactAvailList;) "pub"
      contactRole    (%contactRoleList;)  "all"   >

<!ELEMENT ContactNested  (Comment? )>
<!ATTLIST ContactNested
      contactName       CDATA             #REQUIRED
      orgURN            CDATA             #REQUIRED
      orgRole        (%orgRoleList;)      #REQUIRED
      submitId          IDREF             #IMPLIED
      requestNbr        CDATA             #IMPLIED
      availability   (%contactAvailList;) #REQUIRED
      contactRole    (%contactRoleList;)  #REQUIRED
      email             CDATA             #REQUIRED
      telephone         CDATA             #IMPLIED
      fax               CDATA             #IMPLIED   >

<!ELEMENT ContactInstance  (Comment? )>
<!ATTLIST ContactInstance
      contactId         ID                #REQUIRED
      contactName       CDATA             #REQUIRED
      orgId             IDREF             #REQUIRED
      orgURN            CDATA             #IMPLIED
      orgRole        (%orgRoleList;)      #REQUIRED
      submitId          IDREF             #IMPLIED
      requestNbr        CDATA             #IMPLIED
      availability   (%contactAvailList;) #REQUIRED
      contactRole    (%contactRoleList;)  #REQUIRED
      email             CDATA             #REQUIRED
      telephone         CDATA             #IMPLIED
      fax               CDATA             #IMPLIED   >
```

Semantic Rules

1. A submission of a proposed new contact instance from a Submitting Organization (SO) to a Registry uses the Contact element. An exchange or representation of an existing contact instance in a nested format, where some parent item instance is known, uses the ContactNested element. An exchange or representation of an existing contact instance as a flat file uses the ContactInstance element. The Contact class definition is in Section 3.5.

2. The orgURN attribute corresponds to the orgId attribute of the Contact class in the information model. If present, it is a URN that identifies an organization known to the RA. All XML entities are defined in Section 6.18.

3. The contactId attribute corresponds to the contactId attribute of the Contact class in the information model. It need be unique only within a given XML document.

4. The submitId attribute corresponds to the implicit submitId attribute of the Submission class in the information model. If present, it must identify a SubmissionInstance element in the containing XML document.

5. The requestNbr attribute corresponds to the requestNbr attribute of the Request class in the information model. If present, it must identify a Request element in the containing XML document.

6.7 AlternateName Elements

Purpose

To define or represent an alternate name instance.

Definition

```
<!ELEMENT AlternateName
   ( AltName, Comment? )>
<!ATTLIST AlternateName
   nameContext   (%nameContextList;)    #REQUIRED
   submittingOrg        CDATA              #IMPLIED
   language          CDATA              #IMPLIED
   encoding          CDATA              #IMPLIED >

<!ELEMENT AltName  (#PCDATA )>

<!ELEMENT AlternateNameFlat
   ( AltName, Comment? )>
<!ATTLIST AlternateNameFlat
   regEntryId        IDREF              #REQUIRED
   regEntryURN       CDATA              #IMPLIED
   nameContext   (%nameContextList;)    #REQUIRED
   submittingOrg        CDATA              #IMPLIED
   language          CDATA              #IMPLIED
   encoding          CDATA              #IMPLIED >
```

Semantic Rules

1. A submission of a proposed new alternate name instance from a Submitting Organization (SO) to a Registry uses the AlternateName element. An exchange or representation of an existing alternate name instance in a nested format, where some parent item instance is known, uses the AlternateName element. An exchange or representation of an existing alternate name instance as a flat file uses the AlternateNameFlat element. The AlternateName class definition is in Section 3.8 and all XML ENTITY definitions are in Section 6.18.

2. The regEntryId attribute corresponds to the regEntryId attribute of the RegistryEntry class in the information model. It must identify a RegistryEntry element in the same XML document.

3. The regEntryURN attribute, if present, must match the assignedURN of the registry entry identified by regEntryId.

4. The altName attribute must contain at least 1 visible character.

6.8 Description Elements

Purpose

To define or represent a description instance.

Definition

```
<!ELEMENT Description ( FullDescription, KeywordList?, Abstract? )>
<!ATTLIST Description
      language       CDATA    #REQUIRED
      encoding       CDATA    #REQUIRED
      submittingOrg     CDATA    #IMPLIED  >

<!ELEMENT DescriptionFlat ( FullDescription, KeywordList?, Abstract? )>
<!ATTLIST DescriptionFlat
      regEntryId     IDREF    #REQUIRED
      regEntryURN    CDATA    #IMPLIED
      language       CDATA    #REQUIRED
      encoding       CDATA    #REQUIRED
      submittingOrg  CDATA    #IMPLIED  >

<!ELEMENT FullDescription (#PCDATA)>

<!ELEMENT KeywordList  (#PCDATA)>

<!ELEMENT Abstract  (#PCDATA)>
```

Semantic Rules

1. A submission of a proposed new description instance from a Submitting Organization (SO) to a Registry uses the Description element. An exchange or representation of an existing description instance in a nested format, where some parent item instance is known, uses the Description element. An exchange or representation of an existing description instance as a flat file uses the DescriptionFlat element. The Description class definition is in Section 3.10.

2. The regEntryId attribute corresponds to the regEntryId attribute of the RegistryEntry class in the information model. It must identify a RegistryEntry element in the same XML document.

3. The regEntryURN attribute, if present, must match the assignedURN of the registry entry identified by regEntryId.

4. The Description element must contain at least 1 visible character.

6.9 Contribution Elements

Purpose

To define or represent a constribution instance.

Definition

```
<!ELEMENT Contribution
   ( ContributorName,
     ContributorRole,
     Comment?  )>
<!ATTLIST Contribution
   contributorURL       CDATA    #IMPLIED
   roleCategory         CDATA    #IMPLIED >

<!ELEMENT ContributionFlat  ( Comment? )>
<!ATTLIST ContributionFlat
   regEntryId           IDREF    #REQUIRED
   contributorURL       CDATA    #IMPLIED
   roleCategory         CDATA    #IMPLIED >

<!ELEMENT ContributorName   ( #PCDATA )>

<!ELEMENT ContributorRole   ( #PCDATA )>
```

Semantic Rules

1. A submission of a proposed new contribution instance from a Submitting Organization (SO) to a Registry uses the Contribution element. An exchange or representation of an existing contribution instance in a nested format, where some parent registry entry instance is known, uses the Contribution element. An exchange or representation of existing contribution instances as a flat file uses the ContributionFlat element. The Contribution class definition is in Section 3.11.

2. The regEntryId attribute corresponds to the regEntryId attribute of the RegistryEntry class in the information model. It must identify a registry entry element in the same XML document.

3. The ContributorName element corresponds to the contributorName attribute of the Contribution class in the information model.

4. The ContributorRole element corresponds to the contributorRole attribute of the Contribution class in the information model.

5. The contributorURL attribute corresponds to the contributorURL attribute of the Contribution class in the information model.

6. The ContributorName, ContributorRole, and Comment elements must each contain at least 1 visible character.

6.10 SubmissionInstance Element

Purpose

To represent a submission instance. It does not include the requests that make up the content of a submission.

Definition

```
<!ELEMENT SubmissionInstance  ( Comment? )>
<!ATTLIST SubmissionInstance
   submitId       ID       #REQUIRED
   submitTime     CDATA    #REQUIRED
   submitOrgId    IDREF    #IMPLIED
   submitOrgURN   CDATA    #REQUIRED  >
```

Semantic Rules

1. The SubmissionInstance element is intended for use by Registration Authorities (RA) for Registry information representation and exchange. It should not be used by a Submitting Organization to specify a submission. Instead, the SubmitRequest DTD should be used. The Submission class definition is in Section 3.6.

2. The submitId attribute corresponds to the hidden submitId attribute of the Submission class in the information model. It need be unique only within a given XML document.

3. The submitOrgId attribute corresponds to the submittingOrg attribute of the Submission class in the information model. If present, it must identify an Organization element in the containing XML document.

4. The submitOrgURN attribute corresponds to the submittingOrg attribute of the Submission class in the information model. It must identify an organization known to the RA.

6.11 Request Elements

Purpose

To represent a request instance.

Definition

```
<!ELEMENT RequestNested  ( Comment? )>
<!ATTLIST RequestNested
   requestCode    (%requestCodeList;)  #REQUIRED
   contentXML        CDATA             #IMPLIED  >

<!ELEMENT RequestFlat  ( Comment? )>
<!ATTLIST RequestFlat
   submitId          IDREF             #REQUIRED
   requestNbr        CDATA             #REQUIRED
   requestCode    (%requestCodeList;)  #REQUIRED
   contentXML        CDATA             #IMPLIED  >
```

Semantic Rules

1. The RequestInstance element is intended for use by Registration Authorities for Registry information representation and exchange. It should not be used by a Submitting Organization to specify a request as part of a submission. Instead, the Request element should be used as part of a SubmitRequest DTD. The Request class definition is in Section 3.7 and all XML ENTITY definitions are in Section 6.18.

2. The submitId attribute corresponds to the submitId attribute of the Submission class in the information model. It must reference a SubmissionInstance element in the containing XML document.

3. The requestNbr attribute corresponds to the requestNbr attribute of the Request class in the information model.

6.12 Impact Element

Purpose

To represent an impact instance.

Definition

```
<!ELEMENT Impact  ( Comment? )>
<!ATTLIST Impact
   regEntryId        IDREF          #IMPLIED
   regEntryURN       CDATA          #REQUIRED
   submitId          IDREF          #IMPLIED
   requestNbr        CDATA          #IMPLIED
   impactCode   (%impactCodeList;)  #REQUIRED  >
```

Semantic Rules

1. The Impact element is intended for use by Registration Authorities for Registry information representation and exchange. It is completely determined by the Registration Authority and should not be used by a Submitting Organization as any part of a submission. The Impact class definition is in Section 3.12 and all XML ENTITY definitions are in Section 6.18.

2. The regEntryId attribute corresponds to the regEntryId attribute of the RegistryEntry class in the information model. If present, it must identify a RegistryEntry element in the same XML document.

3. The regEntryURN attribute must match the assignedURN of the registry entry identified by regEntryId.

4. The (submitId, requestNbr) pair, if present, must identify a Request element in the containing XML document.

6.13 ClassifSchemeInstance Element

Purpose

To represent a classification scheme instance. It does not include the levels or items of the classification scheme.

Definition

```
<!ELEMENT ClassifSchemeInstance  ( Comment? )>
<!ATTLIST ClassifSchemeInstance
   schemeURN      CDATA    #REQUIRED
   commonName     CDATA    #IMPLIED
   schemeId       IDREF    #IMPLIED  >
```

Semantic Rules

1. The ClassifSchemeInstance element is intended for use by Registration Authorities for Registry information representation and exchange. It cannnot be used by a Submitting Organization to define a classification scheme as part of a submission request. Instead, the DefineClassificationScheme request element must be used. The ClassificationScheme class definition is in Section 3.13.

2. The schemeURN attribute corresponds to the schemeURN attribute of the ClassificationScheme class in the information model. It must be equal to the assignedURN of a registry entry in the Registry.

3. The commonName attribute, if present, must be equal to the commonName of the registry entry identified by the schemeURN.

4. The schemeId attribute corresponds to the implicit regEntryId attribute of the ClassificationScheme class in the information model. If present, it must match the regEntryId of a registry entry element in the containing XML document.

6.14 ClassificationLevel Elements

Purpose

To define, identify, or exchange a classification level instance. It does not include the item values defined for that level.

Definition

```
<!ELEMENT ClassificationLevel  ( Comment? )>
<!ATTLIST ClassificationLevel
   levelCode   CDATA    #REQUIRED
   levelName   CDATA    #IMPLIED
   levelNbr    CDATA    #IMPLIED >

<!ELEMENT ClassificationLevelFlat  ( Comment? )>
<!ATTLIST ClassificationLevelFlat
   schemeURN   CDATA    #REQUIRED
   schemeId    IDREF    #IMPLIED
   levelCode   CDATA    #REQUIRED
   levelName   CDATA    #IMPLIED
   levelNbr    CDATA    #IMPLIED  >
```

Semantic Rules

1. A submission of a proposed new classification level instance from a Submitting Organization (SO) to a Registry uses the ClassificationLevel element. An exchange or representation of an existing classification level instance in a nested format, where the parent classification scheme is known, uses the ClassificationLevel element. An exchange or representation of an existing classification level instance as a flat file uses the ClassificationLevelFlat element. The ClassificationLevel class definition is in Section 3.13.

2. If a ClassificationLevel element is part of a DefineClassificationScheme element, then the levelCode and levelName attributes define a level code and a level name for the indicated level of the classification scheme. The levelNbr, if present, is superfluous and will be ignored by the registry.

3. If a ClassificationLevel element is not part of a DefineClassificationScheme element, then the levelName attribute, if present, must be the levelName of the level identified by the levelCode.

4. If a ClassificationLevel element is not part of a DefineClassificationScheme element, then the levelNbr attribute, if present, must be the levelNbr of the level identified by the levelCode.

5. The schemeURN attribute corresponds to the schemeURN attribute of the ClassificationScheme class in the information model. It must be equal to the assignedURN of a registry entry in the Registry.

6. The schemeId attribute corresponds to the regEntryId attribute of the RegistryEntry class in the information model. If present, it must match the regEntryId of some registry entry element in the containing XML document.

6.15 ClassificationNode Elements

Purpose

To define, identify, or represent the itemValue and itemName of a classification node instance.

Definition

```
<!ELEMENT ClassificationItem ( Comment? )>
<!ATTLIST ClassificationItem
   itemValue      CDATA    #REQUIRED
   itemName       CDATA    #IMPLIED
   levelNbr       CDATA    #IMPLIED
   levelCode      CDATA    #IMPLIED >

<!ELEMENT ClassificationItemFlat  ( Comment? )>
<!ATTLIST ClassificationItemFlat
   schemeURN      CDATA    #REQUIRED
   schemeId       IDREF    #IMPLIED
   regEntryId      ID      #REQUIRED
   nodeId           ID     #REQUIRED
   itemValue      CDATA    #REQUIRED
   parentId       IDREF    #REQUIRED
   itemName       CDATA    #IMPLIED
   levelNbr       CDATA    #IMPLIED
   levelCode      CDATA    #IMPLIED >
```

Semantic Rules

1. A submission of a proposed new classification item instance from a Submitting Organization (SO) to a Registry uses the ClassificationItem element. An exchange or representation of an existing classification item instance in a nested format, where the parent classification scheme is known, uses the ClassificationItem element. An exchange or representation of an existing classification item instance in a flat file uses the ClassificationItemFlat element. The ClassificationNode class definition is in Section 3.13.

2. If a ClassificationItem element is part of a DefineClassificationScheme element, then the itemValue and itemName attributes define an item value and an item name for the parent classification scheme.

3. If a ClassificationItem element is not part of a DefineClassificationScheme element, then the itemName must match the itemName for the classification item determined by the itemValue.

4. The levelCode and levelNbr attributes, if present in a ClassificationItem element, identify the level of the classification item in the parent classification scheme.

5. The levelCode and levelNbr attributes, if present in a ClassificationItemFlat element, identify the level of the classification item in the classification scheme identified by the schemeURN.

6. The itemName attribute, if present, must match the itemName of the classification item identified by itemValue.

7. The regEntryId attribute, if present, corresponds to the regEntryId attribute of the RegistryEntry class in the information model. It must identify a RegistryEntry element in the same XML document.

8. The regEntryURN attribute must match the assignedURN of the registry entry identified by regEntryId.

6.16 RegistryMetadata Elements

Purpose

To define or represent the public metadata, in a nested format, for a single registry entry.

Definition

```
<!ELEMENT RegistryMetadataSubmit
(  RegistryEntrySubmit ,
   Association* ,
   Classification* ,
   ExternalData* ,
   AlternateName* ,
   Contribution* ,
   Description*  )>

<!ELEMENT RegistryMetadataInstance
(  RegistryEntryInstance ,
   Association* ,
   Classification* ,
   ExternalData* ,
   AlternateName* ,
   Contribution* ,
   Description*  )>
```

Semantic Rules

1. The association instances represented by the Association elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

2. The classification instances represented by the Classification elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

3. The external data instances represented by the ExternalData elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

4. The alternate name instances represented by the AlternateName elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

5. The contribution instances represented by the Contribution elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

6. The description instances represented by the Description elements are all related to the registry entry instance represented by the RegistryEntrySubmit or the RegistryEntryInstance element.

6.17 Repository Element

Purpose

To represent or exchange zero or more registered objects.

Definition

```
<!ELEMENT Repository ( RegisteredObject* ) >
```

Semantic Rules

1. A registered object is only materialized when an object is submitted to a Registry/Repository for storage or safekeeping or when it is extracted from a repository for repository exchange or as the response to a query.

2. The representation of a registered object as part of a submission is specified in the RegisterObject request defined in Section 8.18.

3. The representation of a registered object as part of a Query response is specified in the GetRegisteredObject DTD defined in Section 7.1.

6.18 XML Entity Definitions

assocRoleList -- to identify a value of the AssociationRole enumeration domain (see Section 5.7).

```
<!ENTITY % assocRoleList
      "val | req | cnt | sup | rep | rel" >
```

contactAvailList -- to identify a value of the ContactAvailability enumeration domain (see Section 5.9).

```
<!ENTITY % contactAvailList
      "pub | pri | pro " >
```

contactRoleList -- to identify a value of the ContactRole enumeration domain (see Section 5.10).

```
<!ENTITY % contactRoleList
      "admn | all | tech" >
```

defnSourceList -- to identify a value of the DefinitionSource enumeration domain (see Section 4.1).

```
<!ENTITY % defnSourceList
      " ebXML | LTSC_LOM | IMS | OASIS " >
```

impactCodeList -- to identify a value of the ImpactCode enumeration domain (see Section 4.4).

```
<!ENTITY % impactCodeList
   " AAS | ACF | ACT | AAN | ACB | ARO | ARE | AED | ASO
   | ADS | DAS | DCF | DCT | DAN | DCB | DRO | DRE | DED
   | DSO | DDS | UAS | UCF | UCT | UAN | UCB | URO | URE
   | UED | USO | UDS " >
```

contextTypeList -- to identify a value of the NameContext enumeration domain (see Section 5.8).

```
<!ENTITY % nameContextList
      "cpp | code | java | lang | long | short | sql | urn" >
```

orgRoleList -- to identify a value of the OrganizationRole enumeration domain (see Section 4.2).

```
<!ENTITY % orgRoleList
      " SO | RO | RA " >
```

feeStatusList -- to identify a value of the FeeStatus enumeration domain (see Section 5.5).

```
<!ENTITY % feeStatusList
      "debit  |  free  |  pswd" >
```

propRightsList -- to identify a value of the PropertRights enumeration domain (see Section 5.6)

```
<!ENTITY % propRightsList
             "none | ANY" >
```

objectTypeList -- to identify a value of the ObjectType enumeration domain (see Section 5.1).

```
<!ENTITY % objectTypeList
      "defn | inst | rpkg | schm | mime | othr" >
```

regStatusList -- to identify a value of the RegistrationStatus enumeration domain (see Section 5.3).

```
<!ENTITY % regStatusList
      "sub | urw | reg | sup | dep | rep | wth | exp " >
```

relatedRoleList -- to identify a value of the RelatedRole enumeration domain (see Section 5.11).

```
<!ENTITY % relatedRoleList
       "Changelog | CvrLetter | DistribHP | DocSet |
        DocSetInfo | DSSLSS | DSSLSSInfo |
        EmailInfo | Example | ExpSet | ExpSetInfo | FAQ | Other |
        PublicText | ReadMe | RefMan | RegInfo | RelDataGrp |
        SchemaHP | SGMLDeclar | SGMLOpnCat | StyleSinfo |  ToolInfo |
        UserGuide | WhitePaper | XSLSS | XSLSSinfo " >
```

requestCodeList -- to identify a value of the RequestCode enumeration domain (see Section 4.3).

```
<!ENTITY % requestCodeList
   " addAssoc | addClassif | addAltName | addContrib | addExtData |
     addDescrip | defClassSchm | defRegPkg | delAssoc | delClassif |
     delAltName | delContrib | delExtData | delDescrip | modClassif |
     modRegPkg | modRegEntry | regObj | regSO | refRegObj |
     repRegObj | subRegObj | wdrRegObj ">
```

fileTypeList -- to identify a value of the FileType enumeration domain (see Section 5.2).

```
<!ENTITY % fileTypeList
   " xmlDTD    | sgmlDTD   | xmlSchema | xdrSchema |
     soxSchema | rdfSchema | sgmlElement  |
     xmlElement   | sgmlAttrib| xmlAttrib |
     sgmlAttSet   | xmlAttSet | sgmlAttVal   |
     xmlAttVal | sgmlParm  | xmlParm | charEntSet |
     sgml | zip | xml | html-1 | html-2 | html-3 | html-4 |
     html-iso | xhtml " >
```

stabilityList -- to identify a value of the Stability enumeration domain (see Section 5.4).

```
<!ENTITY % stabilityList
      "comp | dynm | stat">
```

7. Registry Services

7.1 GetRegisteredObject DTD's

Purpose

To obtain one or more registered objects, and some associated metadata, by submitting a RegistryEntryQuery to the registry/repository that holds the desired objects.

NOTE: Initially, the RegistryEntryQuery is a single AssignedURN!

Definitions

Get Request DTD

```
<!ELEMENT GetRegisteredObject
   (  RegistryEntryQuery,
      RecursiveAssocOption?,
      WithShortDescription? )>

<!ELEMENT RecursiveAssocOption ( AssociationRole+ )>
<!ATTLIST RecursiveAssocOption
   depthLimit  CDATA    #IMPLIED >

<!ELEMENT AssociationRole EMPTY >
<!ATTLIST AssociationRole
   role     CDATA     #REQUIRED  >

<!ELEMENT WithShortDescription EMPTY >
```

Get Result DTD

```
<!ELEMENT GetRegisteredObjectResult
   ( RegisteredObject*,  StatusResult  )>  [Should StatusResult be
separate?]

<!ELEMENT RegisteredObject
   (     ClassificationScheme
      |  RegistryPackage
      |  UnknownObject
      |  WithdrawnObject
      |  RemoteObject      )>
<!ATTLIST RegisteredObject
   assignedURN          CDATA   #REQUIRED
   commonName           CDATA   #REQUIRED
   objectURL            CDATA   #REQUIRED
   objectType           CDATA   #REQUIRED
   fileType             CDATA   #IMPLIED
   registrationStatus   CDATA   #REQUIRED
   stability            CDATA   #REQUIRED
   shortDescription     CDATA   #IMPLIED  >

<!ELEMENT UnknownObject  (#PCDATA) >
<!ATTLIST UnknownObject
   byteEncoding    CDATA    #IMPLIED  >

<!ELEMENT WithdrawnObject EMPTY >

<!ELEMENT RemoteObject EMPTY >
```

Semantic Rules

1. If the RecursiveOption element is not present , then set Limit=0. If the RecursiveOption element is present, interpret its depthLimit attribute as an integer literal. If the depthLimit attribute is not present, then set Limit = -1. If a depthLimit value is presnt, but it cannot be interpreted as a positive integer, then stop execution and raise the exception: *invalid depth limit*; otherwise, set Limit=N, where N is that positive integer.

2. Set Depth=0. Let Result denote the set of RegisteredObject elements to be returned as part of the GetRegisteredObjectResult. Initially Result is empty.

3. If the WithShortDescription element is present, then set WSD="yes"; otherwise, set WSD="no".

4. Execute the RegistryEntryQuery according to the Semantic Rules of RegistryEntryQuery specified in Section 7.4.1. Let R be the set of RegistryEntryReference elements returned by the RegistryEntryQResult and let S be the set of status elements returned in the StatusResult. If any status element in S is an exception condition, then stop execution and return the same StatusResult.

5. Execute Semantic Rules 6 and 7 with X as the set of RegistryEntry instances referenced by R. If Depth is now equal to Limit, then return the content of Result as the set of RegisteredObject elements in the GetRegisteredObjectResult element; otherwise, continue with Semantic Rule 8.

6. Let X be a set of RegistryEntry instances. For each registry entry E in X, do the following:

    a) If E.objectURL references a registered object in this registry/repository, then create a new RegisteredObject element, with values for its attributes derived as specified in Semantic Rule 7.

        1) If E.objectType="scheme", then put the referenced ClassificationScheme DTD as the subelement of this RegisteredObject.

        2) If E.objectType="rpkg", then put the referenced RegistryPackage DTD as the subelement of this RegisteredObject.

        3) Otherwise, i.e., if the object referenced by E has an unknown internal structure, then put the content of the registered object as the #PCDATA of a new UnknownObject subelement of this RegisteredObject.

    b) If E.objectURL references a registered object in some other registry/repository, then create a new RegisteredObject element, with values for its attributes derived as specified in Semantic Rule 7, and create a new RemoteObject element as the subelement of this RegisteredObject.

    c) If E.objectURL is void, i.e. the object it would have referenced has been withdrawn, then create a new RegisteredObject element, with values for its attributes derived as specified in Semantic Rule 7, and create a new WithdrawnObject element as the subelement of this RegisteredObject.

7. Let E be a registry entry and let RO be the RegisteredObject element created in Semantic Rule 6. Set the attributes of RO to the values derived from the corresponding attributes of E. If WSD="yes", include the value of the shortDescription attribute; otherwise, do not include it. Insert this new RegisteredObject element into the Result set.

8. Let R be defined as in Semantic Rule 4. Execute Semantic Rule 9 with Y as the set of RegistryEntry instances referenced by R. Then continue with Semantic rule 10.

9. Let Y be a set of RegistryEntry instances. Let NextLevel be an empty set of RegistryEntry instances. For each registry entry E in Y, and for each AssociationRole A of the RecursiveAssocOption, do the following:

    a) Let Z be the set of associated items E' linked to E under association instances having E as the given item, E' as the associated item, and A as the association role.

b) Add the elements of Z to NextLevel.

10. Let X be the set of new registry entries that are in NextLevel but are not yet represented in the Result set.

    Case:

    a) If X is empty, then return the content of Result as the set of RegisteredObject elements in the GetRegisteredObjectResult element.

    b) If X is not empty, then execute Semantic Rules 6 and 7 with X as the input set. When finished, add the elements of X to Y and set Depth=Depth+1. If Depth is now equal to Limit, then return the content of Result as the set of RegisteredObject elements in the GetRegisteredObjectResult element; otherwise, repeat Semantic Rules 9 and 10 with the new set Y of registry entries.

11. If any exception, warning , or other status condition results during the execution of the above, then return appropriate status elements as the StatusResult of the GetRegisteredObjectResult element created in Semantic Rule 5 or Semantic Rule 10.

7.2 GetRegistryEntry DTD's

Purpose

To obtain selected registry metadata associated with one or more registry entries, by submitting a RegistryEntryQuery to the registry/repository that holds the registry entries.

NOTE: Initially, the RegistryEntryQuery is a single AssignedURN!

Definition

Request DTD

```
<!ELEMENT GetRegistryEntry
   (     RegistryEntryQuery,
    ( |  WithClassifications
      |  WithAssociations
      |  WithExternalData
      |  WithAlternateNames
      |  WithContributions
      |  WithDescriptions )*   )>

<!ELEMENT WithClassifications  ( ClassificationFilter? )>
<!ELEMENT WithAssociations  ( AssociationFilter? )>
<!ELEMENT WithExternalData  ( ExternalDataFilter? )>
<!ELEMENT WithAlternateNames  ( AlternateNameFilter? )>
<!ELEMENT WithContributions  ( ContributionFilter? )>
<!ELEMENT WithDescriptions  ( DescriptionFilter? )>
```

Response DTD

```
<!ELEMENT GetRegistryEntryResult
   ( RegistryMetadataInstance*,  StatusResult )> [Should Status Result be
separate?]
```

Semantic Rules

1. Execute the RegistryEntryQuery according to the Semantic Rules of RegistryEntryQuery specified in Section 7.4.1. Let R be the set of RegistryEntryReference elements returned by the RegistryEntryQResult and let S be the set of status elements returned in the StatusResult. If any status element in S is an exception condition, then stop execution and return the same StatusResult element in the GetRegistryEntryResult.

2. If the set R is empty, then do not return a RegistryMetadataInstance subelement in the GetRegistryEntryResult. Instead, raise the warning: *no resulting registry entry*. Add this warning to the StatusResult returned by the RegistryEntryQResult and return this enhanced StatusResult with the GetRegistryEntryResult

3. For each registry entry E referenced by an element of R, use the attributes of E to create a new RegistryEntryInstance element as defined in Section 6.1. Then create a new RegistryMetadataInstance element as defined in Section 6.16.

4. If no With option is specified, then the resulting RegistryEntryInstance element has no Classification, Association, ExternalData, AlternateName, Contribution, or Description subelements. The set of RegistryMetadataInstance elements, with the StatusResult from the RegistryEntryQResult, is returned as the GetRegistryEntryResult.

5. If WithClassifications is specified, then for each E in R do the following: If a ClassificationFilter is not present, then let C be any classification instance linked to E; otherwise, let C be a classification instance linked to E that satisfies the ClassificationFilter (Section 9.3). For each such C, create a new Classification element as defined in Section 6.3. Add these Classification elements to their related RegistryEntryInstance as defined in Section 6.16.

6. If WithAssociations is specified, then for each E in R do the following: If an AssociationFilter is not present, then let A be any association instance linked to E; otherwise, let A be an association instance linked to E that satisfies the AssociationFilter (Section 9.2). For each such A, create a new Association element as defined in Section 6.2. Add these Association elements to their related RegistryEntryInstance as defined in Section 6.16.

7. If WithExternalData is specified, then for each E in R do the following: If an ExternalDataFilter is not present, then let D be any external data instance linked to E; otherwise, let D be an external data instance linked to E that satisfies the ExternalDataFilter (Section 9.4). For each such D, create a new ExternalData element as defined in Section 6.4. Add these ExternalData elements to their related RegistryEntryInstance as defined in Section 6.16.

8. If WithAlternateNames is specified, then for each E in R do the following: If an AlternateNameFilter is not present, then let N be any alternate name instance linked to E; otherwise, let N be an alternate name instance linked to E that satisfies the AlternateNameFilter (Section 9.5). For each such N, create a new AlternateName element as defined in Section 6.7. Add these AlternateName elements to their related RegistryEntryInstance as defined in Section 6.16.

9. If WithContributions is specified, then for each E in R do the following: If a ContributionFilter is not present, then let C be any contribution instance linked to E; otherwise, let C be a contribution instance linked to E that satisfies the ContributionFilter (Section 9.7). For each such C, create a new Contribution element as defined in Section 6.9. Add these Contribution elements to their related RegistryEntryInstance as defined in Section 6.16.

10. If WithDescriptions is specified, then for each E in R do the following: If a DescriptionFilter is not present, then let D be any description instance linked to E; otherwise, let D be a description instance linked to E that satisfies the DescriptionFilter (Section 9.6). For each such D, create a new Description element as defined in Section 6.8. Add these Description elements to their related RegistryEntryInstance as defined in Section 6.16.

11. If any warning or exception condition results, then add the code and the message to the StatusResult that came from the RegistryEntryQResult.

12. Return the set of RegistryMetadataInstance elements and the revised StatusResult as the GetRegistryEntryResult.

7.3 SubmitRequest DTD

Purpose

To submit a collection of one or more requests to a Registry, to identify one or more contacts able to address issues related to that submission, and optionally, to identify one or more specific contacts for each request.

Definition

```
<!ELEMENT SubmitRequest ( AuthenticationToken?, Request+, Contact+ )>
<!ATTLIST SubmitRequest
      submitOrgURN   CDATA   #REQUIRED >

<!ELEMENT AuthenticationToken  (#PCDATA) >

<!ELEMENT Request
  (  (
         AddAssociation
       | AddClassification
       | AddAlternateName
       | AddContribution
       | AddDescription
       | AddExternalData
       | DefineClassificationScheme
       | DefineRegistryPackage
       | DeleteAssociation
       | DeleteClassification
       | DeleteAlternateName
       | DeleteContribution
       | DeleteDescription
       | DeleteExternalData
       | ModifyClassificationScheme
       | ModifyRegistryPackage
       | ModifyRegistryEntry
       | RegisterObject
       | RegisterSubmittingOrg
       | ReaffirmRegisteredObject
       | ReplaceRegisteredObject
       | SupercedeRegisteredObject
       | WithdrawRegisteredObject    ),
    Contact*,
    Comment? )>
```

Semantic Rules

1. A SubmitRequest DTD flows from a Submitting Organization (SO) to a Registration Authority (RA).

2. The RA treats the submission atomically - it is either completely successful or it fails with no effect on the Registry.

3. The AuthenticationToken may or may not be required by the RA. It's content is specified separately by the RA and is not part of this specification.

4. If the SubmitRequest element contains a RegisterSubmittingOrg subelement, then the content of assignedURN is superfluous; otherwise, the submitOrgURN identifies an SO known to the RA. If no RegisterSubmittingOrg subelement is present, and if the assignedURN is not known to the RA, then the RA returns an error: *unknown submitting organization*.

5. If successful, the SubmitRequest results in a new instance of the Submission class, and each Request element results in a new instance of the Request class. Each Contact in SubmitRequest determines a Contact instance linked to the submisssion and each Contact in a Request determines a contact linked to that request instance.

6. Each Comment in a Request element is intended as further information from the SO to the RA regarding that request instance; it is not intended for access by registry users, although the RA may include some portion of the Comment as the comment attribute of the Request instance.

7.4 RegistryQuery DTD's

Purpose

To propose a query to a Registry/Repository implementation, with the expectation of receiving back a query result and a status result. The query result for each query type is a set of references to registry instances of the implied class. The status result is a success indication or a collection of warnings and/or exceptions.

NOTE: A Registry/Repository may conform at the lowest level by supporting only a RegistryEntryQuery that consists of a single AssignedURN. Other query support is required at higher levels of conformance.

Definition

Query DTD

```
<!ELEMENT RegistryQuery
 (    RegistryEntryQuery
   |  ContactQuery
   |  RequestQuery
   |  ImpactQuery
   |  OrganizationQuery    )>
```

QueryResult DTD

```
<!ELEMENT RegistryQueryResult
 (
      RegistryEntryQResult
   |  ContactQResult
   |  RequestQResult
   |  ImpactQResult
   |  OrganizationQResult     )>

<!ELEMENT RegistryEntryQResult ( RegistryEntryReference* )>

<!ELEMENT RegistryEntryReference EMPTY >
<!ATTLIST RegistryEntryReference
   assignedURN    CDATA    #REQUIRED
   objectURL      CDATA    #IMPLIED
   regEntryId      ID      #IMPLIED  >

<!ELEMENT ContactQResult ( ContactReference* )>

<!ELEMENT ContactReference EMPTY >
<!ATTLIST ContactReference
   contactName    CDATA    #REQUIRED
   orgURN         CDATA    #REQUIRED
   email          CDATA    #REQUIRED
   contactID       ID      #IMPLIED  >

<!ELEMENT RequestQResult ( RequestReference* )>

<!ELEMENT RequestReference EMPTY >
<!ATTLIST RequestReference
   submitTime     CDATA    #REQUIRED
   requestNbr     CDATA    #REQUIRED
   requestCode    CDATA    #REQUIRED
   requestId       ID      #IMPLIED  >

<!ELEMENT ImpactQResult ( ImpactReference* )>

<!ELEMENT ImpactReference EMPTY >
```

```
<!ATTLIST ImpactReference
   submitTime     CDATA    #REQUIRED
   requestNbr     CDATA    #REQUIRED
   submitId        ID      #IMPLIED
   regEntryURN    CDATA    #REQUIRED
   objectURL      CDATA    #IMPLIED
   regEntryID      ID      #IMPLIED
   impactCode     CDATA    #REQUIRED
   impactId        ID      #IMPLIED  >
```

StatusResult

```
<!ELEMENT StatusResult
   ( Success | ( Exception | Warning )+ >

<!ELEMENT Success EMPTY >

<!ELEMENT Exception ( #PCDATA )>
<!ATTLIST Exception
   code     CDATA    #REQUIRED >

<!ELEMENT Warning ( #PCDATA )>
<!ATTLIST Warning
   code     CDATA    #REQUIRED >
```

Semantic Rules

1. The semantic rules for each RegistryQuery alternative are specified in subsequent Subsections.

2. [NOT COMPLETE] -- Specify distinctions among Warnings and Exceptions!

3. If any exception or warning results, then it is returned as the appropriate alternative of the StatusResult element.

7.4.1 RegistryEntryQuery

Purpose

To identify a set of registry entry instances by a query over selected registry metadata.

Definition

```
<!ELEMENT RegistryEntryQuery
  (   AssignedURN+
    | MetadataFilter
    | RegistryEntrySQL
    | RegistryEntryXML
    | RegistryEntryOQL     )>

<!ELEMENT AssignedURN  EMPTY >
<!ATTLIST AssignedURN
   URN    CDATA    #REQUIRED >

<!ELEMENT MetadataFilter
  (   RegistryEntryFilter?,
      (AssociationGivenFilter | AssociationAssocFilter )?,
      ClassificationFilter?,
      ExternalDataFilter?,
      AlternateNameFilter?,
      DescriptionFilter?,
      ContributionFilter?,
      SubmittingOrgFilter?,
      ImpactFilter?,
      RequestFilter?,
      ContactFilter?       )>

<!ELEMENT AssociationGivenFilter ( Association Filter )>

<!ELEMENT AssociationAssocFilter ( Association Filter )>

<!ELEMENT SubmittingOrgFilter ( Organization Filter )>

<!ELEMENT RegistryEntrySQL ( #PCDATA )>

<!ELEMENT RegistryEntryXML ( #PCDATA )>

<!ELEMENT RegistryEntryOQL ( #PCDATA )>
```

Semantic Rules

1. If a list of AssignedURN elements is specified as a RegistryEntryQuery, then:

   a) Each AssignedURN should identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If any registry entry does not exist, then raise the warning: *assigned urn does not exist*; otherwise, let E identify the registry entry.

   b) For each E, create a new RegistryEntryReference element as defined in Section 7.4, with the assignedURN and objectURL attributes of E as the corresponding attributes of the new element. Optionally, an implementation may include a persistent object identifier as the value of the regEntryId attribute.

   c) Return the set of RegistryEntryReference elements as the RegistryEntryQResult.

   d) Return the set of warnings as the StatusResult associated with the RegistryEntryQResult.

2. If a MetadataFilter element is specified as a RegistryEntryQuery, then:

   a) Let RE denote the set of all persistent RegistryEntry instances in the Registry. The following steps will eliminate instances in RE that do not satisfy the conditions of the specified filters.

   b) If a RegistryEntryFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a registry entry in RE. If x does not satisfy the RegistryEntryFilter as defined in Section 9.1, then remove x from RE.

   c) If an AssociationGivenFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let AF be the set of Association instances that satisfy the AssociationFilter of the AssociationGivenFilter element as defined in Section 9.2. If x is not the given item of some association instance in AF, then remove x from RE.

   d) If an AssociationAssocFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let AF be the set of Association instances that satisfy the AssociationFilter of the AssociationAssocFilter element as defined in Section 9.2. If x is not the associated item of some association instance in AF, then remove x from RE.

   e) If a ClassificationFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let CF be the set of Classification instances that satisfy the ClassificationFilter as defined in Section 9.3. If x is not the parent of some classification in CF, then remove x from RE.

   f) If an ExternalDataFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let EDF be the set of ExternalData instances that satisfy the ExternalDataFilter as defined in Section 9.4. If x is not the parent of some external data instance in EDF, then remove x from RE.

   g) If an AlternateNameFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let ANF be the set of AlternateName instances that satisfy the AlternateNameFilter as defined in Section 9.5. If x is not the parent of some alternate name in ANF, then remove x from RE.

   h) If a DescriptionFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let DF be the set of Description instances that satisfy the DescriptionFilter as defined in Section 9.6. If x is not the parent of some description in DF, then remove x from RE.

   i) If a ContributionFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let CF be the set of Contribution instances that satisfy the ContributionFilter as defined in Section 9.7. If x is not the parent of some contribution instance in CF, then remove x from RE.

   j) If a SubmittingOrgFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let OF be the set of Organization instances that satisfy the OrganizationFilter of the SubmittingOrgFilter element as defined in Section 9.8. If the submittingOrg attribute of x does not reference some organization instance in OF, then remove x from RE.

   k) If an ImpactFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let IF be the set of Impact instances that satisfy the ImpactFilter as defined in Section 9.9. Let RegistryEntry(IF) be the set of RegistryEntry instances that are linked to some impact instance of IF. If x is not a member of RegistryEntry(IF), then remove x from RE.

   l) If a RequestFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let RF be the set of Request instances that satisfy the RequestFilter as defined in Section 9.10. Let ImpactedBy(RF) be the set of RegistryEntry instances that are linked to some element of RF through an Impact instance. If x is not a member of ImpactedBy(RF), then remove x from RE.

   m) If a ContactFilter is not specified, or if RE is empty, then continue below; otherwise, let x be a remaining registry entry in RE. Let CF be the set of Contact instances that satisfy the ContactFilter as defined in Section

9.11. Let FromSubmissions(CF) be the set of Submission instances whose set of submission contacts intersects any element of CF. Let Requests(FromSubmissions(CF)) be the set of Request instances that have one of the identified submissions as a parent. Let ImpactedBy(Requests(FromSubmissions(CF))) be the set of RegistryEntry instances that are linked to one of the identified requests through an Impact instance. If x is not a member of ImpactedBy(Requests(FromSubmissions(CF))), then remove x from RE.

n) If RE is empty, then raise the warning: *query result is empty*, and return a RegistryEntryQResult with no RegistryEntryReference elements. Otherwise, for each remaining x in RE, create a new RegistryEntryReference element as defined in Section 7.4, with the assignedURN and objectURL attributes of x as the corresponding attributes of the new element. Optionally, an implementation may include a persistent object identifier as the value of the regEntryId attribute.

o) Return the set of RegistryEntryReference elements as the RegistryEntryQResult.

p) Return the set of warnings as the StatusResult associated with the RegistryEntryQResult.

3. If a RegistryEntrySQL element is specified as a RegistryEntryQuery, then the PCDATA contained in the RegistryEntrySQL element shall conform to an SQL <query expression> as defined in International Standard ISO/IEC 9075 - Database Language SQL. A Registry/Repository implementation may require that the <query expression> be constrained by the rules for Minimal SQL as specified in Annex A1.3.

   a) The <from clause> of a <query expression> contained in a RegistryEntryQuery may be restricted to a single <table reference> referencing exactly one of the following views defined in Annex A1.2:

REGISTRY_ENTRY	REGENTRY_LJ_GIVENITEM	REGENTRY_LJ_ASSOCITEM
REGENTRY_LJ_EXTDATA	REGENTRY_LJ_ALTNAME	REGENTRY_LJ_DESCRIPTION
REGENTRY_LJ_CONTRIBUTION	REGENTRY_LJ_SUBMITORG	REGENTRY_LJ_IMPACT
REGENTRY_LJ_CONTACT	REGENTRY_LJ_LEVELVALUEPAIR	REGENTRY_LJ_LVPEXTENDED

   b) The <select list> of a <query expression> contained in a RegistryEntryQuery shall include the assignedURN column from REGISTRY_ENTRY, and that column shall uniquely identify a row of the resulting table R.

   c) For each row x of R, create a new RegistryEntryReference element as defined in Section 7.4. Set the attributes of this RegistryEntryReference with the attributes of the registry entry identified by the assignedURN column of x. A correct value for the objectURL attribute of the RegistryEntryReference is required even if objectURL was not included as a column in the <select list> of the <query expression>. Optionally, an implementation may include a persistent object identifier as the value of the regEntryId attribute.

   d) Set the values of the StatusResult with any exceptions or warnings returned by the SQL processor. The SQLSTATE shall be returned as the code attribute of the Exception or Warning subelement of StatusResult, and associated exception or warning messages shall be returned as the PCDATA of each element.

   e) Return the set of RegistryEntryReference elements as the RegistryEntryQResult and return the StatusResult as specified by the communications protocol being used.

4. If a RegistryEntryXML element is specified as a RegistryEntryQuery, then the PCDATA contained in the RegistryEntrySQL element shall conform to an XML Query as defined by the XML Query technical committee in W3C. [NOT FINISHED].

5. If a RegistryEntryOQL element is specified as a RegistryEntryQuery, then the PCDATA contained in the RegistryEntrySQL element shall conform to an Object query as defined by ODMG. [NOT FINISHED].

7.4.2 ContactQuery

Purpose

To identify a set of contact instances by a query over selected registry metadata.

Definition

```
<!ELEMENT ContactQuery         [NOT FINISHED!]
  (   MetadataFilter
    | ContactSQL
    | ContactEntryXML
    | ContactEntryOQL      )>

<!ELEMENT ContactSQL ( #PCDATA )>

<!ELEMENT ContactXML ( #PCDATA )>

<!ELEMENT ContactOQL ( #PCDATA )>
```

Semantic Rules

[NOT FINISHED!]

7.4.3 RequestQuery

Purpose

To identify a set of request instances by a query over selected registry metadata.

Definition

```
<!ELEMENT RequestQuery         [NOT FINISHED!]
  (   MetadataFilter
    | RequestSQL
    | RequestXML
    | RequestOQL     )>


<!ELEMENT RequestSQL ( #PCDATA )>

<!ELEMENT RequestXML ( #PCDATA )>

<!ELEMENT RequestOQL ( #PCDATA )>
```

Semantic Rules

```
[NOT FINISHED!]
```

7.4.4 ImpactQuery

Purpose

To identify a set of impact instances by a query over selected registry metadata.

Definition

```
<!ELEMENT ImpactQuery     [NOT FINISHED!]
  (   MetadataFilter
    | ImpactSQL
    | ImpactXML
    | ImpactOQL      )>


<!ELEMENT ImpactSQL ( #PCDATA )>

<!ELEMENT ImpactXML ( #PCDATA )>

<!ELEMENT ImpactOQL ( #PCDATA )>
```

Semantic Rules

```
[NOT FINISHED!]
```

7.4.5 OrganizationQuery

Purpose

To identify a set of organization instances by a query over selected registry metadata.

Definition

```
<!ELEMENT OrganizationQuery       [NOT FINISHED!]
  (   MetadataFilter
    | OrganizationSQL
    | OrganizationXML
    | OrganizationOQL      )>


<!ELEMENT OrganizationSQL ( #PCDATA )>

<!ELEMENT OrganizationXML ( #PCDATA )>

<!ELEMENT OrganizationOQL ( #PCDATA )>
```

Semantic Rules

[NOT FINISHED!]

7.5 ClassificationScheme DTD

Purpose

To define or represent a complete classification scheme as a hierarchy of nodes.

Definition

```
<!ELEMENT ClassificationScheme
  (  Comment?,
     ClassificationLevel*,
     ClassificationNode+   ) >
<!ATTLIST ClassificationScheme
   schemeName   CDATA   #IMPLIED  >

<!ELEMENT ClassificationNode
  (( ClassificationItem, ClassificationNode*) | USER_INPUT )>

<!ELEMENT USER_INPUT EMPTY >
```

Semantic Rules

1. The nested hierarchy of ClassificationNode elements determines the partial ordering of a classification scheme over those nodes. The mathematical definition of classification scheme is in Section 2.4. The itemValue and itemName attributes identify the itemValue and itemName of each node.

2. The ClassificationLevel elements, if present, must be equal in number to the number of levels in the classification scheme derived from the nested hierarchy of ClassificationNode's. The levelCode and levelName attributes identify the levelCode and levelName of each level.

3. The schemeName, if present, identifies the commonName of the classification scheme.

4. If USER_INPUT is specified as a ClassificationNode sub-element, then the itemValue in any classification that references this classification scheme can be any value that satisfies the datatype for itemValue as an attribute of the LevelValuePair class defined in Section 3.3.

5. The ClassificationLevel element is defined in Section 6.14.

6. The ClassificationItem element is defined in Section 6.15.

EDITOR's NOTE: It is desirable that one be able to define a new classification scheme from parts of existing classification schemes. For example, an existing GeographyClassifScheme might be used as the basis of a new classification scheme that classifies commercial vendors by the geography of their sales regions and the geographic locations of their raw resources. This capability is not yet included in the ClassificationScheme DTD.

7.6 RegistryPackage DTD

Purpose

To represent a registered registry package as a collection of registry entry URN's and other optional identifiers.

Definition

```
<!ELEMENT  RegistryPackage (PackageMember*)>
<!ATTLIST RegistryPackage
   packageURN    CDATA   #REQUIRED
   packageName   CDATA   #IMPLIED
   pkgID         ID      #IMPLIED >

<!ELEMENT PackageMember EMPTY >
<!ATTLIST PackageMember
   memberURN     CDATA   #REQUIRED
   memberURL     CDATA   #IMPLIED
   memberID      ID      #IMPLIED >
```

Semantic Rules

1. The packageURN identifies a registry entry whose objectType attribute is *registry package*.

2. The packageName, if present, is the commonName of the registry entry identified by packageURN.

3. The objectURL of the registry entry identified by packageURN points to a registry package in a Repository managed by the RA.

4. Each memberURN identifies a registry entry in a Registry managed by the RA.

5. Each registry entry identified by a memberURN participates as the assocItem in an association instance where the associationRole is *Contains* and the givenItem is the registry entry identified by the packageURN.

6. The pkgId, if present, is an object reference to the registry entry identified by the packageURN.

7. Each memberId, if present, is an object reference to the registry entry identified by the member URN.

7.7 RegistryContentFlat DTD

Purpose

To represent a complete and consistent subset of a Registry as a collection of flat tables.

Definition

```
<!ELEMENT RegistryContentFlat
      (
            RegistryEntryInstance* ,
            AssociationFlat* ,
            ExternalDataFlat* ,
            AlternateNameFlat* ,
            DescriptionFlat* ,
            ClassifSchemeInstance* ,
            ClassificationLevelFlat* ,
            ClassificationItemFlat* ,
            ClassificationFlat* ,
            OrganizationInstance* ,
            ContactInstance* ,
            SubmissionInstance* ,
            RequestFlat* ,
            Impact*     )>
<!ATTLIST RegistryContentFlat
   RegistryURN    CDATA #REQUIRED >
```

Semantic Rules

1. [NOT COMPLETE]

2. [NOT COMPLETE]

7.8 RegistryContentNested DTD

Purpose

To represent a complete and consistent subset of a Registry as a collection of hierarchically structured elements.

Definition

```
<!ELEMENT RegistryContentNested
   (  RegistryMetadataInstance* ,
      OrganizationAndContacts* ,
      ClassificationScheme* ,
      SubmissionHistory*,
      Impact*     )>
<!ATTLIST RegistryContentNested
      RegistryURN  CDATA   #REQUIRED >

<!ELEMENT OrganizationAndContacts
      ( OrganizationInstance, ContactNested+ )>

<!ELEMENT SubmissionHistory
      ( SubmissionInstance, ContactNested+, RequestSummary+ )>

<!ELEMENT RequestSummary
      ( RequestNested, ContactNested* )>
```

Semantic Rules

1. [NOT COMPLETE]

2. [NOT COMPLETE]

8. Request Elements

8.1 AddAssociation

Purpose

To add one or more new associations for an existing registry entry.

Definition

```
<!ELEMENT AddAssociation ( Association+ )>
<!ATTLIST AddAssociation
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist.*

2. Each Association element shall satisfy the semantic rules for an Association. If any does not, then raise theexception: *invalid association.*

3. For each Association element, using the contained information of that element, the RA shall create an association instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Insert Association* (IAS) impact on that registry entry.

5. For each Association element, if its AssociatedItem subelement references a registry entry in the same Registry, then the RA shall insert a new Impact instance to indicate that this request had an *Insert Association* (IAS) impact on that registry entry. NOTE: If the AssociatedItem references a registry entry in some other Registry, do we want to send a message to that Registry?

8.2 AddClassification

Purpose

To add one or more new classifications to an existing registry entry.

Definition

```
<!ELEMENT AddClassification ( Classification+ )>
<!ATTLIST AddClassification
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist.*

2. Each Classification element shall satisfy the semantic rules for a Classification. If any does not, then raise theexception: *invalid classification.*

3. For each Classification element, using the contained information of that element, the RA shall create a classification instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Insert Classification* (ICF) impact on that registry entry.

8.3 AddAlternateName

Purpose

To add one or more new alternate names to an existing registry entry.

Definition

```
<!ELEMENT AddAlternateName ( AlternateName+ )>
<!ATTLIST AddAlternateName
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each AlternateName element shall satisfy the semantic rules for an AlternateName. If any does not, then raise theexception: *invalid alternate name.*

3. For each AlternateName, using the contained information of that element, the RA shall create an alternate name instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Insert Alternate Name* (IAN) impact on that registry entry.

8.4 AddContribution

Purpose

To add one or more new contribution instances to an existing registry entry.

Definition

```
<!ELEMENT AddContribution ( Contribution+ )>
<!ATTLIST AddContribution
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each ExternalData element shall satisfy the semantic rules for ExternalData. If any does not, then raise theexception: *invalid contribution.*

3. For each Contribution element, using the contained information of that element, the RA shall create a contribution instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Add Contribution* (ACB) impact on that registry entry.

8.5 AddDescription

Purpose

To add one or more new Description instances to an existing registry entry.

Definition

```
<!ELEMENT AddDescription ( Description+ )>
<!ATTLIST AddDescription
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each Description element shall satisfy the semantic rules for a Description. If any does not, then raise theexception: *invalid description*.

3. For each Description, using the contained information of that element, the RA shall create a description instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had an *Add Description* (ADS) impact on that registry entry.

8.6 AddExternalData

Purpose

To add one or more new external data items to an existing registry entry.

Definition

```
<!ELEMENT AddExternalData ( ExternalData+ )>
<!ATTLIST AddExternalData
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each ExternalData element shall satisfy the semantic rules for ExternalData. If any does not, then raise theexception: *invalid external data.*

3. For each ExternalData element, using the contained information of that element, the RA shall create an external data instance linked to the registry entry instance referenced by assignedURN.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Add External Data* (AED) impact on that registry entry.

8.7 DefineClassificationScheme

Purpose

To define and register a single new classification scheme.

Definition

```
<!ELEMENT DefineClassificationScheme
    ( ClassificationSchemeMetadata, ClassificationScheme )>

<!ELEMENT ClassificationSchemeMetadata  ( RegistryMetadataSubmit ) >
```

Semantic Rules

1. The ClassificationScheme element shall satisfy the semantic rules of the ClassificationScheme DTD. It defines the node hierarchy for the resulting classification scheme. If it contains any illegal values, then raise theexception: *invalid classification scheme hierarchy*.

2. The RegistryMetadataSubmit element shall satisfy semantic rules as if it were a subelement of a RegisterObject element. If it does not, then the RA returns the identified error.

3. The RA shall create a new registry entry as if a RegisterObject request were executed with this RegistryMetadataSubmit element.

4. For each ClassificationLevel subelement of ClassificationScheme, the RA creates a new instance of the ClassificationLevel class.

5. For each ClassificationNode subelement directly contained in ClassificationScheme, the RA creates a new instance of the ClassificationItem class with 0 as its ParentId.

6. For each ClassificationNode subelement directly contained in another ClassificationNode element, the RA creates a new instance of the ClassificationItem class, with the nodeId of its parent element as its parentId.

7. For each ClassificationLevel subelement of the ClassificationScheme element, the RA creates a new instance of the ClassificationLevel class. The levelNbr attribute of each level is its numerical position in the list of ClassificationLevel elements, beginning with 1.

8.8 DefineRegistryPackage

Purpose

To define and register a single new package of registry entries.

Definition

```
<!ELEMENT DefineRegistryPackage
    ( RegistryPackageMetadata,
      RegistryPackageSubmit   )>

<!ELEMENT RegistryPackageSubmit ( PackageMemberSubmit* )>

<!ELEMENT PackageMemberSubmit
    ( assignedURN | PkgItemRef | PackageMemberMetadata)>

<!ELEMENT assignedURN (#PCDATA)>

<!ELEMENT PkgItemRef EMPTY >
<!ATTLIST PkgItemRef
    regEntryId  IDREF  #REQUIRED >

<!ELEMENT RegistryPackageMetadata  ( RegistryMetadataSubmit ) >

<!ELEMENT PackageMemberMetadata  ( RegistryMetadataSubmit ) >
```

Semantic Rules

1. The Submitting Organization (SO) is defining a single, complete package of registry entries. The package may be empty, or it may consist of references to previously registered items, or it may consist of references to registration requests in the same submission, or it may consist of new registration requests for its elements, or it may be a combination of these alternatives.

2. The SO owns the package, even if it references registry entries owned by other submitting organizations.

3. The RegistryMetadataSubmit element directly contained in RegistryPackageMetadata shall satisfy semantic rules as if it were a RegistryMetadataSubmit subelement of a RegisterObject element. If it does not satisfy these semantic rules, then the RA returns the identified error; otherwise, the RA shall create a new registry entry as if a RegisterObject request were executed with this RegistryMetadataSubmit.

4. If a PackageMember is an assignedURN, then the assignedURN shall be identical to the assignedURN of some existing RegistryEntry instance in some registry managed by the RA. If the assigned URN does not exist, then raise theexception: *assigned urn does not exist.* NOTE: We may want to allow the possibility that the assignedURN is in some remote registry, but this may put too much of a checking burden on the RA; for now lets allow an RA to refuse this reference if it does not identify a registry entry in the same registry.

5. If a PackageMember is a PkgItemRef, then the regEntryId attribute shall reference the regEntryId of some RegistryEntrySubmit element in the containing XML document. If this attribte does not reference a metadata element, then raise theexception: *illegal package item reference*. NOTE: A SubmitRequest document could validate to the SubmitRequest DTD if the IDREF references any ID in the same document; this rule ensures that it references a proposed registry entry.

6. A RegistryMetadataSubmit element directly contained in a PackageMemberSubmit element shall satisfy semantic rules as if it were a subelement of a RegisterObject element. If it does not satisfy these semantic rules, then the RA returns the identified error; otherwise, the RA shall create a new registry entry as if a RegisterObject request were executed with each RegistryMetadataSubmit.

7. For each PackageMember, the RA creates a new instance of Association with the newly created package as the GivenItem, with *Contains* as the associationRole, and with the referenced or newly created package element as the AssociatedItem.

8.9 DeleteAssociation

Purpose

To delete one or more associations from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteAssociation
      ( (associationRole, AssociatedItemURN)* )>
<!ATTLIST DeleteAssociation
     assignedURN  CDATA  #REQUIRED>

<!ELEMENT associationRole (#PCDATA)>

<!ELEMENT AssociatedItemURN (#PCDATA)>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each associationRole and AssociatedItemURN pair, if present, shall identify a unique association instance linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced association does not exist.*

3. Case:

    a. If no associationRole and AssociatedItemURN pair exists, then the RA shall delete every association linked to the registry entry identified by the assignedURN as the GivenItem.

    b. Otherwise, the RA shall delete each association identified by a associationRole and AssociatedItemURN pair.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had a *Delete Association* (DAS) impact on that registry entry.

5. The RA shall insert a new Impact instance for each registry entry identified by the AssociatedItemURN in any deleted association instance to indicate that this request had *a Delete Association* (DAS) impact on that registry entry.

8.10 DeleteClassification

Purpose

To delete one or more classifications from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteClassification ( SchemeURN* )>
<!ATTLIST DeleteClassification
     assignedURN  CDATA  #REQUIRED>

<!ELEMENT SchemeURN (#PCDATA)>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. The SchemeURN element shall identify a collection of one or more classification instances linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced classification does not exist*.

3. The RA shall delete each classification instance in the collection identified by SchemeURN.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had a *Delete Classification* (DCF) impact on that registry entry.

8.11 DeleteAlternateName

Purpose

To delete one or more alternate names from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteAlternateName ( (AltName, NameContext)* )>
<!ATTLIST DeleteAlternateName
     assignedURN  CDATA  #REQUIRED>

<!ELEMENT NameContext (#PCDATA)>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. Each AltName and NameContext pair, if present, shall identify a unique alternate name instance linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced alternate name does not exist*.

3. Case:

    a. If no AltName and NameContext pair exists, then the RA shall delete every alternate name linked to the registry entry identified by the assignedURN.

    b. Otherwise, the RA shall delete each alternate name identified by an AltName and NameContext pair.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had a *Delete Alternate Name* (DAN) impact on that registry entry.

8.12 DeleteContribution

Purpose

To delete one or more contribution instances from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteContribution ( (ContributorName, ContributorRole)* )>
<!ATTLIST DeleteContribution
     assignedURN   CDATA  #REQUIRED
     roleCategory  CDATA  #IMPLIED >
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist.*

2. The ContributorName and ContributorRole pair, if present, shall identify a unique contribution instance linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced contribution does not exist.*

3. Case:

    a. If a ContributorName and ContributorRole pair exists, then the value of the roleCategory attribute, if present, is superfluous because the name and role pair identifies a unique contribution instance. The RA shall delete that contribution instance.

    b. If no ContributorName and ContributorRole pair exists, and if no roleCategory attribute is present, then the RA shall delete every contribution instance linked to the registry entry identified by the assignedURN.

    c. If no ContributorName and ContributorRole pair exists, and if a roleCategory attribute has a specific value, then the RA shall delete every contribution instance having that roleCategory value that is linked to the registry entry identified by the assignedURN.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had a *Delete Contribution* (DCB) impact on that registry entry.

8.13 DeleteDescription

Purpose

To delete one or more descriptions from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteDescription ( Language* )>
<!ATTLIST DeleteDescription
     assignedURN  CDATA  #REQUIRED>

<!ELEMENT Language (#PCDATA)>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. The Language element shall identify a unique description instance linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced description does not exist*.

3. The RA shall delete the referenced description.

4. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had a *Delete Description* (DDS) impact on that registry entry.

8.14 DeleteExternalData

Purpose

To delete one or more external data items from the metadata for an existing registry entry.

Definition

```
<!ELEMENT DeleteExternalData ( DataName* )>
<!ATTLIST DeleteExternalData
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. The DataName element, if present, shall identify a unique external data instance linked to the registry entry identified by the assignedURN. If it does not, then raise theexception: *referenced external data does not exist*.

3. Case:

    a. If no DataName element exists, then the RA shall delete every external data instance linked to the registry entry identified by the assignedURN.

    b. Otherwise, the RA shall delete each external data instance identified by a DataName.

4. The RA shall insert a new Impact instance for the registry entry referenced by assignedURN to indicate that this request had a *Delete External Data* (DED) impact on that registry entry.

8.15 ModifyClassificationScheme

Purpose

To modify the definition of an existing classification scheme.

Definition

```
<!ELEMENT ModifyClassificationScheme
    ( ( AddLevels | AddNodes | DeleteLevels | DeleteNodes)+ )>
<!ATTLIST ModifyClassificationScheme
     assignedURN CDATA #REQUIRED>

<!ELEMENT AddLevels ( ClassificationLevel+ )>

<!ELEMENT AddNodes ( ClassificationNode+ )>

<!ELEMENT DeleteLevels ( LevelRef )>

<!ELEMENT DeleteNodes ( ( LevelRef, ItemRef)+ )>

<!ELEMENT LevelRef EMPTY >
<!ATTLIST LevelRef
    levelCode     CDATA "leaf" >

<!ELEMENT ItemRef EMPTY >
<!ATTLIST ItemRef
    itemValue     CDATA #REQUIRED >
```

Semantic Rules

1. [NOT COMPLETE]

2. [NOT COMPLETE]

8.16 ModifyRegistryPackage

Purpose

To modify the contents of an existing package.

Definition

```
<!ELEMENT ModifyRegistryPackage ( ( AddItem | DeleteItem )+ )>
<!ATTLIST ModifyRegistryPackage
     assignedURN CDATA #REQUIRED>

<!ELEMENT AddItem ( PackageMember+ )>

<!ELEMENT DeleteItem ( assignedURN+ )>
```

Semantic Rules

1. [NOT COMPLETE]

2. [NOT COMPLETE]

8.17 ModifyRegistryEntry

Purpose

To modify the attributes of an existing registry entry.

Definition

```
<!ELEMENT ModifyRegistryEntry ( RegistryEntrySubmit )>
<!ATTLIST ModifyRegistryEntry
     assignedURN CDATA #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*. If the assigned URN does identify a registry entry, then let RE represent that entry.

2. Each subelement of the RegistryEntrySubmit shall satisfy the semantic rules associated with that element. If they do not satisfy these semantic rules, then raise theexception: *invalid registry entry subelements.*

3. Each attribute of the RegistryEntrySubmit shall satisfy the semantic rules of RegistryEntry. If they do not satisfy these semantic rules, then raise theexception: *invalid registry entry attributes*.

4. The content of each subelement of the RegistryEntrySubmit element shall replace the value of the corresponding attribute of RE.

5. The value of each attribute of the RegistryEntrySubmit element shall replace the value of the corresponding attribute of RE.

8.18 RegisterObject

Purpose

To provide all of the necessary metadata to register a new object in a Registry. The object itself may or may not be included, depending on whether the Registry is also acting as a Repository for that object.

Definition

```
<!ELEMENT RegisterObject ( RegistryMetadataSubmit, ObjectFile? )>

<!ELEMENT ObjectFile ( #PCDATA )>
<!ATTLIST ObjectFile
   mimeType      CDATA   #IMPLIED
   toCharMethod  CDATA   #IMPLIED  >
```

Semantic Rules

1. The RegistryMetadataSubmit element directly contained in the RegisterObject element contains the metadata that describes the object to be registered. If it contains any illegal values, then raise theexception: *registry metadata submit failure*.

2. If the registry is acting in tandem with a repository for specific types of documents, and if submission of the object to be registered is required by the published rules and procedures of the registry/repository, then the ObjectFile element shall be present. If ObjectFile is not present, or if the objectURL attribute of the RegistryEntrySubmit element in the RegistryMetadataSubmit element does not identify a file location accessible to the RA, then the RA may return the error: *object file required*.

3. The mimeType attribute of the ObjectFile identifies the MIME type (Section 3.14) of the ObjectFile.

4. If the ObjectFile is a binary file, then the toCharMethod attribute identifies the method used to convert bytes to characters that can be transported as XML PCDATA. The specification of the meaning of any value for this attribute is beyond the scope of this specification.

5. If the registry is acting in tandem with a repository for specific types of documents, and if the ObjectFile, or the file referenced by objectURL, does not validate to a published schema definition for objects to be registered, then the RA may return the error: *object does not validate*.

6. The RA shall create a new registry entry using the information from the RegistryEntrySubmit subelement of the RegistryMetadataSubmit element.

7. The RA shall insert a new Impact instance for the newly created registry entry to indicate that this request had an *Insert Registry Entry* (IRE) impact on that registry entry.

8. For every Association subelement of the RegistryMetadataSubmit element, the RA shall execute the rules of AddAssociation, with the URN of the newly created registry entry as the assignedURN.

9. For every Classification subelement of the RegistryMetadataSubmit element, the RA shall execute the rules of AddClassification, with the URN of the newly created registry entry as the assignedURN.

10. For every ExternalData subelement of the RegistryMetadataSubmit element, the RA shall execute the rules of AddExternalData, with the URN of the newly created registry entry as the assignedURN.

11. For every AlternateName subelement of the RegistryMetadataSubmit element, the RA shall execute the rules of AddAlternateName, with the URN of the newly created registry entry as the assignedURN.

12. For every Description subelement of the RegistryMetadataSubmit element, the RA shall execute the rules of AddDescription, with the URN of the newly created registry entry as the assignedURN.

8.19 RegisterSubmittingOrg

Purpose

To request certification from a Registration Authority to be a Submitting Organization for a Registry managed by that Registration Authority.

Definition

```
<!ELEMENT RegisterSubmittingOrg
      ( OrganizationSubmit, Contact+ )>
```

Semantic Rules

1. The RegisterSubmittingOrg element provides information about the intended Submitting Organization (SO). That information is evaluated by the Registration Authority (RA) against its published policies and procedures for certification as an SO.

2. If the information provided is not satisfactory, then the RA returns an error: *certification not successful*.

3. If the orgFullName and country attributes match an already registered SO, then the RA returns an error: *organization already certified*. [NOTE: we may want to use this element to allow an SO to modify some of its contact information, but right now this error assumes that such modifications are accomplished by a different communication element.]

4. [NOT COMPLETE] Add Impact Rules, Set Organization class attributes.

8.20 ReaffirmRegisteredObject

Purpose

To reaffirm the registration of a registered object and suggest a new expirationDate.

Definition

```
<!ELEMENT ReaffirmRegisteredObject   EMPTY >
<!ATTLIST ReaffirmRegisteredObject
     assignedURN      CDATA  #REQUIRED
     expirationDate   CDATA  #IMPLIED  >
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*. If the assigned URN does identify a registry entry, then let RE represent that entry.

2. The expirationDate, if present, is a suggestion by the SO of a new expirationDate for the registered object referenced by RE. The new value is set by the RA depending on its own policies and this suggestion.

3. [NOT COMPLETE] Add Impact Rules, Set Organization class attributes.

8.21 ReplaceRegisteredObject

Purpose

To replace an existing registered object with a new object having new metadata, leaving a modified version of the metadata for the old object in the Registry.

Definition

```
<!ELEMENT ReplaceRegisteredObject
      ( RegistryMetadataSubmit, ObjectFile? )>
<!ATTLIST ReplaceRegisteredObject
     assignedURN  CDATA  #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist*.

2. If the SO submitting this request is not also the SO of the registry entry instance referenced by assignedURN, then raise theexception: *insufficient privileges*.

3. If the stability attribute of the registry entry instance referenced by assignedURN does not allow replacement, then raise theexception: *replacement not allowed*.

4. The RegistryMetadataSubmit element and the ObjectFile element shall satisfy the semantic rules of RegisterObject, as if they were part of a RegisterObject element. If they do not, then the RA returns the identified error. The RA shall create a new registry entry as if a RegisterObject request were executed with these elements.

5. The RA shall create a new association instance, with the newly created registry entry as the parent GivenItem, with assignedURN as the AssocItemURN, with the regEntryId of the assignedURN as the AssocItemId, and with the associationRole set to *supercedes*.

6. The RA shall insert two new Impact instances, one for the newly created registry entry and one for the registry entry referenced by assignedURN, to indicate that this request had an *Insert Association* (IAS) impact on each registry entry.

7. The registry entry instance referenced by assignedURN shall have its registrationStatus attribute changed to *Replaced*, its objectURL attribute shall be modified to point to the location of the replacement object, and the statusChgDate attribute shall be changed to indicate the Datetime that this registry action takes place.

8. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Update Registry Entry* (URE) impact on that registry entry.

8.22 SupercedeRegisteredObject

Purpose

To supercede an existing registered object with a new object having new metadata, leaving the old object in its Repository, and a modified version of the metadata for the old object in the Registry.

Definition

```
<!ELEMENT SupercedeRegisteredObject ( RegistryMetadataSubmit )>
<!ATTLIST SupercedeRegisteredObject
     assignedURN CDATA #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist.*

2. If the SO submitting this request is not also the SO of the registry entry instance referenced by assignedURN, then raise theexception: *insufficient privileges*.

3. The RegistryMetadataSubmit element and the ObjectFile element shall satisfy the semantic rules of RegisterObject, as if they were part of a RegisterObject element. If they do not, then the RA returns the identified error. The RA shall create a new registry entry as if a RegisterObject request were executed with these elements.

4. The RA shall create a new association instance, with the newly created registry entry as the parent GivenItem, with assignedURN as the AssocItemURN, with the regEntryId of the assignedURN as the AssocItemId, and with the associationRole set to supercedes.

5. The RA shall insert two new Impact instances, one for the newly created registry entry and one for the registry entry referenced by assignedURN, to indicate that this request had an *Insert Association* (IAS) impact on each registry entry.

6. The registry entry instance referenced by assignedURN shall have its registrationStatus attribute changed to *Superceded*, and the statusChgDate attribute shall be changed to indicate the Datetime that this registry action takes place. NOTE: Unlike for ReplaceRegisteredObject, the objectURL attribute of the registry entry referenced by assignedURN does not change.

7. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Update Registry Entry* (URE) impact on that registry entry.

8.23 WithdrawRegisteredObject

Purpose

To withdraw a registered object, leaving a modified version of its metadata in the Registry.

Definition

```
<!ELEMENT WithdrawRegisteredObject EMPTY>
<!ATTLIST WithdrawRegisteredObject
     assignedURN CDATA #REQUIRED>
```

Semantic Rules

1. The assignedURN attribute shall identify an existing RegistryEntry instance in some registry managed by the Registration Authority (RA). If it does not, then raise theexception: *assigned urn does not exist.*

2. If the SO submitting this request is not also the SO of the registry entry instance referenced by assignedURN, then raise theexception: *insufficient privileges*.

3. If the stability attribute of the registry entry instance referenced by assignedURN does not allow withdrawal, then raise theexception: *withdrawal not allowed*.

4. The registry entry instance referenced by assignedURN shall have its registrationStatus attribute changed to *Withdrawn*, and the statusChgDate attribute shall be changed to indicate the Datetime that this registry action takes place.

5. The RA shall insert a new Impact instance for the registry entry instance referenced by assignedURN to indicate that this request had an *Update Registry Entry* (URE) impact on that registry entry.

6. The RA shall delete any External Data instances that link to the registry entry referenced by assignedURN.

7. The RA shall delete any association instances that have the registry entry referenced by assignedURN as the GivenItem in that association.

8. The RA shall insert a new Impact instance for each registry entry that has any remaining association instance where the registry entry referenced by assignedURN is the AssocItem in that instance to indicate that this request had *a Delete Association* (DAS) impact on that registry entry. Note: these association instances themselves are not modified in any way; they still point to the registry entry of the now withdrawn object.

8.24 Request by unique identifier

An interoperable network of registries requires at least one agreed-upon protocol for obtaining resources and their metadata. Fortunately, such a protocol was specified in the course of URN development work in the IETF. Resolution of requests by URL and URN are discussed in RFC 2483, URI Resolution Services Necessary for URN Resolution. This an experimental specification appears to be well suited to the purposes of the OASIS Registry and Repository Technical Committee. Its typology of requests, results, errors, and security considerations is well considered.

As the OASIS Registry and Repository Technical Committee is willing to limit the protocols supported to HTTP, the syntax proposed in RFC 2169, “A Trivial Convention for using HTTP in URN Resolution” (THTTP) is specified here, with revisions to bring it in line with the later RFC 2483, to wit, replacement of the L2* and N2* requests with a generic I2* request. Thus emended, section 2.0 of RFC 2169 reads:

> The general approach used to encode resolution service requests in THTTP is quite simple:
>
> GET /uri-res/<service>?<uri> HTTP/1.0
>
> For example, if we have the URN "urn:foo:12345-54321" and want a URL, we would send the request:
>
> GET /uri-res/I2L?urn:foo:12345-54321 HTTP/1.0
>
> The request could also be encoded as an HTTP 1.1 request. This would look like:
>
> GET /uri-res/I2L?urn:foo:12345-54321 HTTP/1.1
> Host: <whatever host we are sending the request to>

Responses from the HTTP server follow standard HTTP practice. Status codes, such as 200 (OK) or 404 (Not Found) shall be returned. The normal rules for determining cachability, negotiating formats, etc. apply.

To use this syntax in general, one would follow the pattern (cast as a URL rather than a full HTTP request):

http://someregistry.org/<function>?argument

To obtain an entity such as the Docbook DTD (the URN is imaginary):

http://someregistry.org/I2R?urn:x-oasis:dtds:Docbook-v3.1

To obtain the composite metadata document for the Docbook DTD (the URN is again imaginary):

http://someregistry.org/I2C?urn:x-oasis:dtds:Docbook-v3.1

RFC 2483 defines an “I2C” request (section 4.5), for resolution of a URL or URN to a description of a resource (URC). As this is a generic request, the OASIS Registry and Repository Technical Committee chooses not to require registries conformant with this specification to return an entity's registration document in response to this request; registries are free to supply whatever their preferred metadata is, which may extend that specified here. (An RA may set its own policy with respect to what metadata it will accept beyond that specified here.) Instead, an additional request, I2X, is specified as returning the entity's registration document as defined here. Some information, such as personal contact information, may be withheld by the RA, perhaps on the basis of the identity of the requestor, if such is its policy.

The “I2CS” request, section 4.6, allows a request for multiple documents; in the absence of agreement on XML packaging, it is not clear that it
would be useful to implement it at the present time.

9. Registry Filters

9.1 RegistryEntryFilter

Purpose

To select a set of RegistryEntry instances from the set of all persistent RegistryEntry instances.

Definition

```
<!ELEMENT RegistryEntryFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the RegistryEntry class.

3. A RegistryEntry instance is said to satisfy the RegistryEntryFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.2 AssociationFilter

Purpose

To select a set of Association instances from the set of all persistent Association instances.

Definition

```
<!ELEMENT AssociationFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Association class.

3. An Association instance is said to satisfy the AssociationFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.3 ClassificationFilter

Purpose

To select a set of Classification instances from the set of all persistent Classification instances.

Definition

```
<!ELEMENT ClassificationFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be an attribute of the Classification class or an attribute of the LevelValuePair class; however attributes from the one class may not be compared with attributes from the other class.

3. NOTE: Also need a way to distinguish among the two "comment" attributes!

4. A Classification instance is said to satisfy the ClassificationFilter if the RegistryPredicate evaluates to *true* for the classification instance linked to at least one of its dependent LevelValuePair instances.

9.4 ExternalDataFilter

Purpose

To select a set of ExternalData instances from the set of all persistent ExternalData instances.

Definition

```
<!ELEMENT ExternalDataFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the ExternalData class.

3. An ExternalData instance is said to satisfy the ExternalDataFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.5 AlternateNameFilter

Purpose

To select a set of AlternateName instances from the set of all persistent AlternateName instances.

Definition

```
<!ELEMENT AlternateNameFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the AlternateName class.

3. An AlternateName instance is said to satisfy the AlternateNameFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.6 DescriptionFilter

Purpose

To select a set of Description instances from the set of all persistent Description instances.

Definition

```
<!ELEMENT DescriptionFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Description class.

3. A Description instance is said to satisfy the DescriptionFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.7 ContributionFilter

Purpose

To select a set of Contribution instances from the set of all persistent Contribution instances.

Definition

```
<!ELEMENT ContributionFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Contribution class.

3. A Ccontribution instance is said to satisfy the ContributionFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.8 OrganizationFilter

Purpose

To select a set of Organization instances from the set of all persistent Organization instances.

Definition

```
<!ELEMENT OrganizationFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Organization class.

3. An Organization instance is said to satisfy the OrganizationFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.9 ImpactFilter

Purpose

To select a set of Impact instances from the set of all persistent Impact instances.

Definition

```
<!ELEMENT ImpactFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Impact class.

3. An Impact instance is said to satisfy the ImpactFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.10 RequestFilter

Purpose

To select a set of Request instances from the set of all persistent Request instances.

Definition

```
<!ELEMENT RequestFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Request class.

3. A Request instance is said to satisfy the RequestFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.11 ContactFilter

Purpose

To select a set of Contact instances from the set of all persistent Contact instances.

Definition

```
<!ELEMENT ContactFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Contact class.

3. A Contact instance is said to satisfy the ContactFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.12 SubmissionFilter

Purpose

To select a set of Contact instances from the set of all persistent Contact instances.

Definition

```
<!ELEMENT SubmissionFilter ( RegistryPredicate )>
```

Semantic Rules

1. The RegistryPredicate element is defined in Section 9.13.

2. Any attribute referenced in the RegistryPredicate shall be a class attribute of the Submission class.

3. A Submission instance is said to satisfy the SubmissionFilter if the RegistryPredicate evaluates to *true* for the class attributes of that instance.

9.13 RegistryPredicate

Purpose

To specify a predicate over attibutes defined by classes in the Registry/Repository Information Model. The predicate uses 2-value logic to return a *true* or *false* result.

Definition

```
<!ELEMENT RegistryPredicate  [NOT FINISHED!] )>
```

Semantic Rules

[NOT FINISHED]

10. Conformance

An implementation may claim conformance to this specification at any of several different levels, including RegistryOnly, RegistryRepositoryBasic, and RegistryRepositoryQuery. In addition, the two repository levels may be specified with or without Validation. Optionally, a conforming implementation at any level of conformance may claim conformance to additional alternatives for SQL Query, XML Query, or OQL Query.

10.1 RegistryOnly

If an implementation claims conformance at the RegistryOnly level, then it must suppport the RegistryEntry, Association, Classification, ExternalData, AlternateName, Contribution, Description, Organization, Contact, and Submission classes via implementation of the XML SubmitRequest DTD with explicit support for the following registry services:

- RegisterSubmittingOrganization, RegisterObject without an ObjectFile, ReaffirmRegisteredObject, ReplaceRegisteredObject, SupercedeRegisteredObject, WithdrawRegisteredObject, ModifyRegistryEntry, AddAssociation, AddClassification, AddAlternateName, AddExternalData, AddProductionCreddit, AddDescription, DeleteAssociation, DeleteClassification, DeleteAlternateName, DeleteExternalData, DeleteContribution, DeleteDescription, and

- GetRegistryEntry (AssignedURN+)

10.2 RegistryRepositoryBasic

If an implementation claims conformance at the RegistryRepositoryBasic level, then it must satisfy the requirements for RegistryOnly conformance. In addition, it must support the RegisteredObject, RegistryPackage, and ClassificationScheme classes via implementation of the following additional registry services:

- RegisterObject with ObjectFile specified, DefineRegistryPackage, DefineClassificationScheme, ModifyRegistryPackage, and

- GetRegisteredObject(AssignedURN+)

10.3 RegistryRepositoryQuery

If an implementation claims conformance at the RegistryRepositoryQuery level, then it must satisfy the requirements for RegistryRepositoryBasic conformance. In addition, it must support full implementation of the Impact class and implementation of the following registry services:

- ModifyClassificationScheme, GetSchemeSubtree [NOT SPECIFIED YET], and
  NOTE: GetSchemeSubtree will allow options to specify the number of levels to retrieve, from one to all.

- RegistryQuery() with support for each of the following alternatives:

  RegistryEntryQuery (MetadataFilter)
  ContactQuery (MetadataFilter)
  RequestQuery (MetadataFilter)
  ImpactQuery (MetadataFilter)
  OrganizationQuery (MetadataFilter)

10.4 with Validation option

An implementation may claim conformance at any RegistryRepository level either with or without Validation. If with Validation is specified, then the implementation must support validation of any registered object whose objectType is *Instance*, whose fileType is *text/xml*, whose registry entry has a *Validates To* association with another registered object

whose objectType is *Definition*, and whose fileType is *xml/dtd* or *xml/schema*. In addition the implementation shall state any other objectType's for which it supports validation.

10.5 with Query options

An implementation may claim conformance at any RegistryRepository level either with or without any of the following options:

- with SQL Query
- with OQL Query
- with XML Query

If any of these options is specified, then the implementation shall support each of the requirements for that alternative as specified in Section 7.4.

10.6 Normative policy requirements

This specification declares the need for certain policies, but may not specify their contents.

Intellectual Property Notices. The registry and repository shall have published policies relating to their provision of intellectual property notices for entities in the repository; that is, whether the interface to the registry or repository warns of the existence of copyright notices, asserted licenses, or other intellectual property restrictions or encumbrances, or leaves it to the user to discover them.

Integrity. The registry and repository shall have published policies relating to their use of methods to guarantee the integrity of entities in repository and metadata in the registry; for example, does the repository employ digital signatures to ensure against corruption? if transformations of registered entities are served are they signed as well?

Security. The registry and repository shall have security policies sufficient to engender confidence in the registry and repository.

Disaster Recovery. The complete content of both the registry and repository shall be backed up offsite, and the backup tested. Some plan shall be made for reconstituting the registry and repository from the backup should the original site be rendered inoperable.

Persistence. The registry and repository shall have published policies relating to its plans for continuing in operation and the outcomes to be expected should it cease operation or should business relationships with the owners of its content change. A point of departure for describing archival longevity is the "Reference Model for an Open Archival Information System" (OAIS) which is a draft ISO standard. It shall be possible for an SO to request that an entity be kept available for a given length of time, also that it be withdrawn after a given length of time.

Retraction. It shall be possible for an SO to request the retraction of an entity.

Privacy. The registry and repository shall have published policies relating to the privacy of users and the sale or other distribution of usage information.

Quality Control. ISO/IEC 11179 defines a data element status value, "certified" (Part 6, p. 9) for a "recorded data element [that] has met the quality requirements specified in this and other parts of ISO/IEC 11179." If the registry provides this or other quality control checking, it shall provide metadata about what specifications an entity conforms to and who did the testing to determine that conformance. (XML validity vs. well-formedness falls under this heading.)

Limitation of Legal Liability. A registry shall have a statement of limitation of legal liability (disclaiming responsibility for the use of information in the repository, for example).

Notice of Quality of Service. A registry shall have a statement of the quality of service it can be expected to provide.

11. Terminology and Relevant Specs

Glossed here are relevant terms, including acronyms, with entries for some specifications relevant to the registry and the repository.

FPI - Formal Public Identifier, defined in the SGML Standard, ISO/IEC 8879, section 10.2, and further in ISO/IEC 9070.

ISO/IEC 8879 - The SGML Standard, “Information processing—Text and office systems—Standard Generalized Markup Language (SGML)”.

ISO/IEC 9070 - Further specifies PIs and FPIs, first defined in the SGML Standard, “Information technology—SGML support facilities—Registration procedures for public text owner identifiers”.

ISO/IEC 11179 - ISO/IEC 11179 is online at http://www.sdct.itl.nist.gov/~ftp/l8/11179/ . The home page of the relevant committee is http://sdct-sunsrv1.ncsl.nist.gov/~ftp/l8/sc32wg2/projects/11179content/content-home.htm with a link to an HTML representation of the stanadard. It is proposed to replace Part 3 of 11179 with ANSI X3.285, “Metamodel for the Management of Shareable Data”, which you can find in HTML at http://www.lbl.gov/~olken/X3L8/drafts/Metamodel/MetaModel_ToC.html and in Word and PDF format (filenames beginning dpX3-285) at ftp://sdct-sunsrv1.ncsl.nist.gov/x3l8/x3l8docs/x3.285/docs/ .

OAIS - “Reference Model for an Open Archival Information System”, a draft ISO standard.

PI - Public Identifier, defined in the SGML Standard, ISO/IEC 8879, section 10.1.6, and further in ISO/IEC 9070.

RA - Registration Authority (ISO/IEC 11179).

Repository - a location or set of distributed locations where documents pointed at by a registry reside, and from which they can be retrieved by conventional (http, ftp) means, perhaps with an additional authentication/permissions layer.

RO - Responsible Organization (ISO/IEC 11179).

SO - Submitting Organization (ISO/IEC 11179).

UML - Unified Modelling Language, an Object Management Group specification for visual modelling of object-oriented systems, see UML Resource Page.

URC - Uniform Resource Characteristics, a general term for any metadata about a resource identified by a URL or URN.

URL - Uniform Resource Locator.

URN - Uniform Resource Name. This is a list of IETF (and other) documents relating to URNs, originally drawn up by Murray Altheim of Sun Microsystems and updated by Terry Allen. The documents he thinks most important are marked with an asterisk.

Charter of the current IETF WG
Some history
Requests For Comments
*Uniform Resource Identifiers (URI): Generic Syntax (RFC 2396)
*URN syntax
*Resolution of Uniform Resource Identifiers using the Domain Name System (RFC 2168)
A Trivial Convention for using HTTP in URN Resolution (RFC 2169)
Architectural Principles of Uniform Resource Name Resolution (RFC 2276)
Using Existing Bibliographic Identifiers as Uniform Resource Names (RFC 2288)
Internationalized Uniform Resource Identifiers (IURI)
*URI Resolution Services Necessary for URN Resolution (RFC 2483)
*A URN Namespace for IETF Documents (RFC 2648)

Internet Drafts
*Resolution of Uniform Resource Identifiers using the Domain Name System
*The Naming Authority Pointer (NAPTR) DNS Resource Record
URN Namespace Definition Mechanisms
Requirements for Human Friendly Identifiers
An Architecture for Supporting Human Friendly Identifiers

XMI - XML Metadata Interchange, an Object Management Group specification of XML formats for interchange of UML models, see UML Resource Page.

XML - Extensible Markup Language, an application profile (that is, an application) of SGML, specified by the W3C, Extensible Markup Language (XML) 1.0.

XML-related entity - In this document, “XML-related entity” means any XML or SGML entity necessary for the processing of an XML document, or documentation of such an entity.

Annex 1 - Database Language SQL Representations

A1.1 - Information Model via SQL Views

If the Registry/Repository Information Model is implemented in a relational database, then the UML classes of the model, or some combination of classes, may be implemented as SQL base tables. These base tables may be private to the implementation with no access to ouside users. However, the implementation may choose to make the information described by each of the UML classes in the model accessible to public users as SQL view tables.

The following SQL view tables are in a one-to-one correspondence with the UML classes of the Information Model. The rows of each table contain the persistent instances in the registry of the corresponding UML class. The non-reference attributes of each class are exposed as columns in the tables and the REF columns are also exposed as columns using the appropriate identifier from the referenced class, i.e. orgURN for an Organization instance, regEntryURN for a RegistryEntry instance, etc. The WHERE clause of each view definition is not visible because it is private to the implementation, and may vary considerably from one implementation to another.

If implementations of the Registry/Repository choose to provide an SQL interface to registry information, they should do so using the following view table names and view column names:

```
create view ALTERNATE_NAME as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Alt.AltName as "altName",
 Alt.NameContext as "nameContext",
 Org.OrgURN as "submittingOrg",
 Alt.Comments as "comment",
 Alt.Language as "language",
 Alt.Encoding as "encoding"
FROM
 REGISTRY_ENTRY_TBL  Entry,
 ALTERNATE_NAME_TBL  Alt,
 ORGANIZATION_TBL  Org
WHERE  privileged-information;


create view ASSOCIATION as
SELECT
 GivenEntry.AssignedURN as
"givenItemURN",
 AssocEntry.AssignedURN as
"assocItemURN",
 Assoc.AssociationRole as
"associationRole",
 Assoc.Comments as "comment"
FROM
 ASSOCIATION_TBL Assoc,
 REGISTRY_ENTRY_TBL GivenEntry,
 REGISTRY_ENTRY_TBL AssocEntry
WHERE  privileged-information;


create view CLASSIFICATION as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Scheme.AssignedURN as "schemeURN",
 Org.OrgURN as "submittingOrgURN",
 Classif.Comments as "comment"
FROM
 CLASSIFICATION_TBL  Classif,
 REGISTRY_ENTRY_TBL  Entry,
 REGISTRY_ENTRY_TBL  Scheme,
 ORGANIZATION_TBL
WHERE  privileged-information;


create view CLASSIFICATION_LEVEL as
SELECT
 Object.SourceURN as "schemeURN",
 Lev.LevelNbr as "levelNbr",
 Lev.LevelCode as "levelCode",
 Lev.LevelName as "levelName",
 Lev.Comments as "comment"
FROM
 CLASSIFICATION_LEVEL_TBL  Lev,
 REGISTERED_OBJECT_TBL  Object
WHERE  privileged-information;


create view CLASSIFICATION_NODE as
SELECT
 Object.SourceURN as "schemeURN",
 Node.NodeId as "nodeId",
 Node.ItemValue as "itemValue",
 Node.ItemName as "itemName",
 Node.ParentId as "parentId",
 Node.LevelNbr as "levelNbr",
 Node.Comments as "comment"
FROM
 CLASSIFICATION_NODE_TBL  Node,
 REGISTERED_OBJECT_TBL  Object
WHERE  privileged-information;


create view CLASSIFICATION_SCHEME as
SELECT
 Object.SourceURN as "schemeURN",
 Scheme.Comments as "comment"
FROM
 CLASSIFICATION_SCHEME_TBL  Scheme,
```

```
 REGISTERED_OBJECT_TBL  Object
WHERE  privileged-information;
create view CONTACT as
SELECT
 Con.ContactName as "contactName",
 Org.OrgURN as "orgURN",
 Con.InternalAddr as
"internalAddress",
 Con.OrgRole as "orgRole",
 Con.Availability as "availability",
 Con.ContactRole as "contactRole",
 Con.Email as "email",
 Con.Telephone as "telephone",
 Con.Fax as "fax",
 Sub.SubmitId as "submitTimestamp",
 Req.RequestNbr as "requestNbr",
 Entry.AssignedURN as "regEntryURN",
 Con.Comments as "comment"
FROM
 CONTACT_TBL  Con,
 SUBMISSION_TBL  Sub,
 REQUEST_TBL  Req,
 ORGANIZATION_TBL  Org,
 REGISTRY_ENTRY_TBL
WHERE  privileged-information;


create view CONTRIBUTION as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Cont.ContributorName as
"contributorName",
 Cont.ContributorRole as
"contributorRole",
 Cont.RoleCategory as
"roleCategory",
 Cont.ContributorURL as
"contributorURL",
 Cont.Comments as "comment"
FROM
 REGISTRY_ENTRY_TBL Entry,
 CONTRIBUTION_TBL Cont
WHERE  privileged-information;


create view DESCRIPTION as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Des.Language as "language",
 Des.Encoding as "encoding",
 Org.OrgURN as "submittingOrg",
 Des.Abstract as "abstract",
 Des.KeywordList as "keywordList",
 Des.FullDescription as
"fullDescription"
FROM
 REGISTRY_ENTRY_TBL  Entry,
 DESCRIPTION_TBL  Des,
 ORGANIZATION_TBL  Org
WHERE  privileged-information;


create view EXTERNAL_DATA as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Ext.DataName as "dataName",
 Ext.DataLocation as "dataLocation",
 Ext.RelatedRole as "relatedRole",
 Ext.MimeType as "mimeType",
 Ext.SizeBytes as "sizeBytes",
 Ext.Comments as "comment"
FROM
 REGISTRY_ENTRY_TBL Entry,
 EXTERNAL_DATA_TBL Ext
WHERE  privileged-information;


create view IMPACT as
SELECT
 Imp.SubmitId as "submitTimestamp",
 Imp.RequestNbr as "requestNbr",
 Entry.AssignedURN as "regEntryURN",
 Imp.ImpactCode as "impactCode",
 Imp.Comments as "comment"
FROM
 REGISTRY_ENTRY_TBL  Entry,
 IMPACT_TBL  Imp
WHERE  privileged-information;


create view LEVELVALUEPAIR as
SELECT
 Entry.AssignedURN as "regEntryURN",
 Scheme.AssignedURN as "schemeURN",
 Org.OrgURN as "submittingOrgURN",
 Lvp.LevelCode as "levelCode",
 Lvp.ItemValue as "itemValue",
 Lvp.Comments as "comment"
FROM
 LEVELVALUEPAIR_TBL  Lvp,
 CLASSIFICATION_TBL  Classif,
 REGISTRY_ENTRY_TBL  Entry,
 REGISTRY_ENTRY_TBL  Scheme,
 ORGANIZATION_TBL  Org
WHERE  privileged-information;


create view ORGANIZATION as
SELECT
 Org.OrgURN as "orgURN",
 Org.OrgFullName as "orgFullName",
 Org.CommonName as "commonName",
 Org.HasSOstatus as "hasSOstatus",
 Org.HasROstatus as "hasROstatus",
 Org.HasRAstatus as "hasRAstatus",
 Parent.OrgUrn as "parentOrg",
 Org.AddrLine1 as "addrLine1",
 Org.AddrLine2 as "addrLine2",
 Org.AddrLine3 as "addrLine3",
 Org.City as "city",
 Org.StateProv as "stateProv",
 Country.ShortName as "country",
 Org.PostalCode as "postalCode",
 Org.Email as "email",
 Org.Telephone as "telephone",
```

```
 Org.Fax as "fax",
 Org.Comments as "comment"
FROM
 ORGANIZATION_TBL  Org,
 ORGANIZATION_TBL  Parent,
 COUNTRY_CODE_TBL  Country
WHERE  privileged-information;


create view REGISTRY_ENTRY as
SELECT
 Reg.AssignedURN as "assignedURN",
 Reg.CommonName as "commonName",
 Reg.Version as "version",
 Reg.ObjectURL as "objectURL",
 Reg.DefnSource as "defnSource",
 Reg.ObjectType as "objectType",
 Reg.FileType as "fileType",
 Reg.RegistrationStatus as
"registrationStatus",
 Reg.StatusChgDate as
"statusChgDate",
 Reg.Stability as "stability",
 Reg.FeeStatus as "feeStatus",
 Reg.PropertyRights as
"propertyRights",
 Reg.ShortDescription as
"shortDescription",
 Reg.ExpirationDate as
"expirationDate",
 SubmitOrg.OrgURN as
"submittingOrg",
 RespOrg.OrgURN as "responsibleOrg",
 Reg.Comments as "comment"
FROM
 REGISTRY_ENTRY_TBL  Reg,
 ORGANIZATION_TBL  SubmitOrg,
 ORGANIZATION_TBL  RespOrg
WHERE  privileged-information;


create view REGISTRY_PACKAGE as
SELECT
 Object.SourceURN as "packageURN",
 Rpkg.Comments as "comment"
FROM
 REGISTRY_PACKAGE_TBL  Rpkg,
 REGISTERED_OBJECT_TBL  Object
WHERE  privileged-information;


create view REQUEST as
SELECT
 Req.SubmitId as "submitTimestamp",
 Req.RequestNbr as "requestNbr",
 Req.RequestCode as "requestCode",
 Req.ContentXML as "contentXML",
 Req.Comments as "comment"
FROM
 REQUEST_TBL  Req
WHERE  privileged-information;;


create view SUBMISSION as
SELECT
 Sub.SubmitTime as "submitTime",
 Org.OrgURN as "submittingOrg",
 Sub.Comments as "comment"
FROM
 SUBMISSION_TBL  Sub,
 ORGANIZATION_TBL  Org
WHERE  privileged-information;
```

A1.2 - Additional Registry Views

The following views represent typical requirements of Registry users. They should be supported by all Registry/Repository implementations that claim to support an SQL interface. The view definitions may use features from upper conformance levels of SQL as definitional mechanisms, but queries against those views may still be limited to Minimal SQL facilites as specified in Annex A1.3.

```
create view REGENTRY_LJ_GIVENITEM as
SELECT
  Entry.*,  Given.* (List out and
rename)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  ASSOCIATION as Given ON
  Entry.assignedURN =
Given.givenItemURN;


create view REGENTRY_LJ_ASSOCITEM as
SELECT
  Entry.*,  Assoc.* (List out and
rename)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  ASSOCIATION as Assoc ON
  Entry.assignedURN =
Assoc.assocItemURN;


create view REGENTRY_LJ_EXTDATA as
SELECT
  Entry.*,  Data.* (List out and
rename)
FROM
```

```
  REGISTRY_ENTRY as Entry LEFT JOIN
  EXTERNAL_DATA as Data ON
  Entry.assignedURN =
Data.regEntryURN;


create view REGENTRY_LJ_ALTNAME as
SELECT
  Entry.*,  Name.* (List out and
rename)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  ALTERNATE_NAME as Name ON
  Entry.assignedURN =
Name.regEntryURN;


create view REGENTRY_LJ_DESCRIPTION
as
SELECT
  Entry.*,  Desc.*  (List out)
FROM
  REGISTRY_ENTRY as Eentry LEFT JOIN
  DESCRIPTION as Desc ON
  Entry.assignedURN =
Desc.regEntryURN;
create view REGENTRY_LJ_CONTRIBUTION
as
SELECT
  Entry.*, Cont.*  (List out!)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  CONTRIBUTION as Cont ON
  Entry.assignedURN =
Cont.regEntryURN;


create view REGENTRY_LJ_SUBMITORG as
SELECT
  Entry.*,  Org.*  (List out!)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  ORGANIZATION as Org ON
  Entry.submittingOrg = Org.orgURN;


create view REGENTRY_LJ_IMPACT as
SELECT
  Entry.*,  Imp.*  (List out!)
FROM REGISTRY_ENTRY as RE LEFT JOIN
IMPACT as IM ON RE.assignedURN =
IM.regEntryURN


create view REGENTRY_LJ_CONTACT as
SELECT
  Entry.*,  Contact.*  (List out!)
FROM
  REGISTRY_ENTRY as Entry LEFT JOIN
  CONTACT as Contact ON
  Entry.assignedURN =
Contact.regEntryURN;


create view LVP_EXPANSION as
SELECT
  Pair.*
FROM
   LEVELVALUEPAIR  Pair,
   CLASSIFICATION_NODE  Node
WHERE
[Do Full Recursion on Nodes to get
all parent pairs represented in the
result];
create view REGENTRY_LJ_LVPEXTENDED
as
SELECT
  Entry.*,  Classif.*, Pairs.* (List
out!)
FROM
  (REGISTRY_ENTRY as Entry LEFT JOIN
  CLASSIFICATION as Classif ON
  Entry.assignedURN =
Classif.regEntryURN)
  LEFT JOIN LVP_EXPANSION as Pairs
ON
  (Classif.schemeURN =
Pairs.schemeURN AND

Classif.regEntryURN=Pairs.regEntryUR
N);
```

A1.3 - Minimal SQL

[Must be updated to reference ISO/IEC 9075:1999 instead of ISO/IEC 9075:1992]

The International Standard for Database Language SQL, ISO/ISO 9075:1992, specifies three levels of conformance for SQL language and SQL implementations:

Entry SQL,
Intermediate SQL, and
Full SQL.

Each of these existing SQL conformance levels requires the facilities of a full-function SQL processor, i.e. schema definition, data manipulation, transaction management, and access control. New conformance alternatives are needed for non-SQL processors that wish to claim conformance to only a portion of the SQL language. Such processors may be able to provide very sophisticated data retrieval capabilities, but may not be able to allow update of data instances or creation of new schema objects. Since existing SQL levels cut across both the schema definition and data manipulation facilities in the SQL standard, it is necessary to consider each SQL level separately as applied to schema definition or data manipulation.

Consider the SQL leveling rules separately for schema definition and data manipulation. Use the term Schema Definition Language (SDL) to identify SQL language features defined in Clause 11, "Schema definition and manipulation", in the SQL'92 standard, and use the term Data Manipulation Language (DML) to identify SQL language features defined in Clause 13, "Data Manipulation". One is then able to discuss the following alternatives for partial support of the SQL language:

Entry DML	Entry SDL
Intermediate DML	Intermediate SDL
Full DML	Full SDL

There is an additional requirement to specify new Minimal DML and Minimal SDL levels to be used exclusively in the definition of application interfaces that do not pretend to support all of the functionality of a full-function SQL processor. These Minimal definitions are intended for use only by non-SQL processors and cannot be used to claim conformance to the SQL standard as an SQL processor.

Minimal DML will support SQL operations on a single table, with no joins and no subqueries, and with severe limitations on derived columns and set functions. Minimal SDL will support specification of only the simplest views and the simplest SQL tables, using only character string, integer, decimal, and real data types, with no table constraints and with only very limited column constraints.

Levels of conformance in the SQL standard are specified by Leveling Rules in each clause of the specification. Using the style of the SQL standard, the following subsections specify restrictions that apply for Minimal SDL and Minimal DML in addition to any restrictions for Entry SQL. All Clause and Subclause references, and all syntactic terms delimited by angle brackets (i.e. <...>) are from ISO/IEC 9075:1992.

Minimal Schema Definition Language

1. A <schema element> contained in a <schema definition> shall be a <table definition> or a <view definition>.

2. A <table element> contained in a <table definition> shall be a <column definition>.

3. A <column constraint> shall not be a <unique specification>, a <references specification>, or a <check constraint definition>; thus a <column constraint> may only specify NOT NULL.

4. In some cases, an SQL/ERI Server implementation at the Minimal SDL level or below may choose not to provide support for SQL null values; if every column of every accessible table is constrained to be NOT NULL, then the implementation may require that every <column definition> in a new <table definition> have an explicit or implicit NOT NULL constraint.

5. The <data type> of a <column definition> shall not specify NUMERIC, FLOAT, or DOUBLE PRECISION; thus a <column definition> may only specify DECIMAL, REAL, INTEGER, SMALLINT, and fixed length CHARACTER string <data type>s. [Note: Must relax this to include VarChar character strings!!]

6. A <view definition> shall not specify WITH CHECK OPTION.

7. The <query expression> contained in a <view definition> shall satisfy the restrictions specified by the Minimal Data Manipulation Language leveling rules below.

Minimal Data Manipulation Language

1. A <query expression> shall be a <query specification>.

2. A <derived column> in the <select list> of a <query specification> shall be a <value expression primary> that is either a <column reference> or a <set function specification>, and the <derived column> shall not contain an <as clause>.

3. A <set function specification> that is a <derived column> in the <select list> of a <query specification> shall be either COUNT(*) or a <general set function> whose directly contained <value expression> is a <column reference>.

4. A <table expression> shall not contain a <group by clause> or a <having clause>.

5. The <from clause> contained in a <table expression> shall contain exactly one <table reference>, and that <table reference> shall be a single <table name> without an associated <correlation name>. A <table name> may be qualified to include a <schema name>.

6. A <search condition> contained in an <SQL data statement> shall not contain any <subquery>. Any <predicate> contained in a <search condition> shall be a <comparison predicate> without subqueries, a <between predicate>, a <like predicate>, a <null predicate>, or an <in predicate> whose <in predicate value> is a parenthesized list of <value specification>s.

7. A <row value constructor> contained in any <predicate> shall have exactly one <row value constructor element> that is a <value expression>.

8. A <value expression> in a <search condition> shall be either a <numeric value expression> or a <string value expression> that is a <character primary>.

9. A <value expression primary> in a <search condition> shall be either a <column reference> or an <unsigned value specification>; thus it may not be a <set function specification> or a <scalar subquery>.

10. A <numeric primary> shall not be a <numeric value function>.

11. A <character primary> shall not be a <character value function>.

12. A <sort key> in a <declare cursor> shall be a <column name>; thus it may not be an <unsigned integer>.

Index