

Candidate Oasis Information Model

A contribution from NIST intended as one section of a
Multi-section OASIS Registry/Repository Specification.

Other sections of the specification, e.g. interfaces and
XML interoperability messages, could be defined in
terms of these definitions and logical structures.

Draft
May 31, 2000

X OASIS Information Model.....	5
X.1 Concepts.....	5
Registry Item	5
Association	6
Related Data.....	7
Organization	8
Contact.....	8
Classification	9
Classification Scheme.....	9
Package	9
Alternate Name.....	10
Alternate Description.....	10
Submission.....	10
Query	10
Request	11
Impact	11
X.2 Registry Model.....	11
X.2.1 Registry Model - Public View	13
X.2.2 Registry Model – Contacts View.....	14
X.2.3 Registry Model – Classification View.....	15
X.2.4 Registry Model – Administration View.....	16
X.2.5 Registry Model – Enumeration Entity View	17
X.3 Classifications.....	18
Simple 1-level classification scheme.....	18
Multi-level naming classification scheme	18
Multi-level coded classification scheme.....	19
Mapping to the information model.....	20
Candidate Classification scheme DTD.....	21
Example scheme DTD	21
Example classification XML.....	22
X.4 Attribute Types	23
X.5 Enumeration Entities.....	26
DefinitionSource.....	26
PrimaryClassification	26
SubClassification	27
RelatedDataType	27
RegistrationStatus.....	28
Stability.....	28
PaymentStatus.....	29
AssociationType	29
ContactAvailability.....	29
ContactRole	29
NameContext.....	30
RegistryImpact.....	30

RegistryRequest.....	31
X.6 Entity Semantics	32
X.6.1 REGISTRY_ITEM.....	32
Structure.....	32
Semantic Rules	32
X.6.2 ASSOCIATION.....	37
Structure.....	37
Semantic Rules	37
X.6.3 RELATED_DATA.....	39
Structure.....	39
Semantic Rules	39
X.6.4 CLASSIFICATION.....	40
Structure.....	40
Semantic Rules	40
X.6.5 ORGANIZATION.....	41
Structure.....	41
Semantic Rules	41
X.6.6 CONTACT.....	43
Structure.....	43
Semantic Rules	43
X.6.7 ALTERNATE_NAME.....	45
Structure.....	45
Semantic Rules	45
X.6.8 ALTERNATE_DESCRIPTION.....	46
Structure.....	46
Semantic Rules	46
X.6.9 SUBMISSION.....	47
Structure.....	47
Semantic Rules	47
X.6.10 REQUEST.....	48
Structure.....	48
Semantic Rules	48
X.6.11 IMPACT.....	49
Structure.....	49
Semantic Rules	49
X.6.12 CLASSIFICATION_SCHEME.....	50
Structure.....	50
Semantic Rules	50
X.6.13 CLASSIFICATION_LEVEL.....	51
Structure.....	51
Semantic Rules	51
X.6.14 CLASSIFICATION_ITEM.....	52
Structure.....	52
Semantic Rules	52
X.6.15 REPOSITORY.....	53

X OASIS Information Model

X.1 Concepts

The OASIS Registry/Repository specification is concerned with the administration, maintenance, presentation, and exchange of metadata related to a *registered object*. The types of objects that can be registered are determined by standardization organizations that sponsor registry/repository specifications. For example, OASIS is interested in registration of XML and SGML objects and other documents, e.g. examples and documentation, that are closely related to registered objects. EBXML is interested in the registration of XML business objects and IMS is interested in the registration of learning objects. The OASIS Registry/Repository specification hopes to be broad and inclusive enough to accommodate the common base requirements of each of these registration efforts; more specific requirements of each registry sponsor can be supported as extensions to the base model.

Each registered object is associated with two pieces of information: 1) an electronic file consisting of a specific digital instance of the registered object, and 2) metadata for naming, describing, and locating the registered object and identifying its associations and relationships with other objects. The metadata for a registered object is maintained by a Registry, and the file containing a registered object is maintained by a Repository. The Registry and Repository are tied together in that the metadata for a registered object in the Registry includes a globally unique locator for a file, in some Repository, that contains the registered object.

If the registered object is a free, publicly available object, then the locator is an anonymous URL or FTP reference that can be used to retrieve the file containing the object; otherwise, the locator 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.

The metadata for a registered object is represented by several general structures, described conceptually as follows:

Registry Item

A Registry Item contains information that identifies, names, and describes each registered object, gives its administrative and payment status, defines its persistence and mutability, classifies it according to pre-defined classification categories, declares its file representation type, and identifies the submitting and responsible organizations. A registry item is a root entity in a Registry in the sense that many other metadata items in the Registry are directly dependent on it. For example, deletion of a registry item results in deletion of all other metadata items that depend on it.

Each registry item has a non-public, local identifier created by the Registration Authority. This identifier is used to maintain relationships of this item with other items in the same registry. In the Registry Information Model, this identifier is said to be of type `REGISTRY_ITEM`. In

addition, each registry item has 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 is intended for use both by humans and by software systems; it is unique within all registries that claim conformance to the OASIS specification. In order to avoid ambiguities in presentation of information, the local identifier and the global name must be in a one-to-one correspondence with one another.

Each registry item is classified by a primary classification, and an optional sub-classification, each defined by the sponsoring organization. For example, for OASIS the primary classification identifies the registered object as either an XSGML definition, an XSGML instance, a Package of registered objects, or some Other document type. The OASIS sub-classification identifies the type XSGML object, e.g. DTD, schema, element, etc. The details and constraints of these and other information items for a Registry Item are given in the Entity Semantics section below.

Association

An Association maintains pairwise relationships among registry items. One item in the pair is called the given item and the other item in the pair is called the associated item. The given item is a registry item in the same Registry as the association instance. The associated item is a registry item in any conformant Registry, possibly one not managed by the same Registration Authority. The associated item is uniquely referenced by its global name. If an associated item is also a registry item in the same Registry as the association instance, then the association instance holds the value of its local identifier.

Each pairwise relationship identifies the role played by the given item in relationship to the associated item. Initially the OASIS specification supports *uses*, *supersedes*, *contains*, and *related* roles.

- A given item *uses* an associated item if the object identified by the given item references the object identified by the associated item or if the object identified by the given item is considered to be incomplete unless the object identified by the associated item is available to it.
- A given item *supersedes* an associated item if both items remain in the Registry but it is intended that the object identified by the given item replace the object identified by the associated item whenever practical. A registry item may remain in the Registry even though the object referenced by it may be withdrawn.
- A given item *contains* an associated item if the object referenced by the given item is a package that contains a pointer to the associated item. The concept of a package is defined below.
- A given item is *related* to an associated item if the given item is documentation or some other kind of important support for the associated item. The given item and the associated item must both be registered in the same registry.

The details and constraints of associations and roles for the OASIS specification are given in the Entity Semantics section below.

Examples:

- 1) A registered XML document *uses* the registered XML DTD that defines its structure.
- 2) A registered XML DTD *uses* a registered XML element or attribute that the DTD references by the global name of the associated element or attribute.
- 3) A registered XML DTD *uses* a registered package consisting of other registered XML DTD's and their associated documentation and style sheets if it references the package by its global name and references package contents by their common names.
- 4) A registered XML element *supersedes* another registered XML element if it has the same common name but a different version ID and thus a different global name.
- 5) A registered diagram of some MIME type is *related* to a registered XML DTD if the diagram is a graphic visualization of the DTD.

Related Data

Related Data is a conceptual notion used to collect together references to data objects that are related to a registered object. This category of metadata is reserved for things like graphic visualizations, example sets, white papers, usage scenarios, extended documentation, etc. Sometimes related data objects will be very important, e.g. usage documentation, and will themselves be registered objects. At other times, related data objects will be less important support information for a registered item and will not themselves be registered.

In most cases, related data objects will not be registered, so their own metadata will not be maintained by a registration authority. Yet they are valuable supplements to a registered object and should be optionally available. In these cases the registration authority will only keep a list of the name and type of the related data object, with just enough additional information so that they can be presented as options on a web page. The related objects and their associated metadata will be created, held, and maintained by the submitting organization.

If a related data instance is not registered, then an instance of it consists of a human readable name, a URL, a standardized classification as a supporting object, a MIME file representation type, and a short human readable comment.

If a related data instance is registered, then its full documentation is carried as metadata in the Registry, including its MIME filetype and its standardized classification as a related data object. As with any registered item, its associations and relationships with other objects will be

maintained by the Registry. In particular, its *related* role to some other item will be captured by an association instance.

Organization

The Organization concept 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. The process of application and approval is not defined in this specification, but the end result is that a Submitting Organization and responsible contacts within that organization are known to each Registration Authority to which they intend to make submissions.

Similarly, each Registration Authority and responsible contacts within that authority must be known to some central accrediting authority before that organization can be recognized as a legitimate Registration Authority. Once an organization has been accredited as a Registration Authority, then under proper authentication that organization will be trusted by other Registration Authorities for the sharing of Registry or Repository data.

Finally, a Responsible Organization is the organization responsible for creation and revision of the registered object. In many cases, the Responsible Organization will be an accredited standards development organization (e.g. as defined by ISO or ANSI), but it may also be some Submitting Organization. Every Responsible Organization should be able to supply a set of development and voting procedures whereby an initial specification gets approved, as well as development and voting procedures for corrections, amendments, revisions, and withdrawals. The Registration Authority is responsible for additions and changes to the metadata associated with a registered object, and for ensuring that revisions and withdrawals of registered objects abide by initial stability and persistence declarations.

Each organization has a non-public local identifier created by the Registration Authority that is used to maintain relationships among organizations, contacts, and registered objects. In the Registry Information Model, this identifier is said to be of type ORGANIZATION. In addition, each organization has 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 is intended for use both by humans and by software systems; it is unique within all registries that claim conformance to the OASIS specification. In order to avoid ambiguities in presentation of information, the local identifier and the global name must be in a one-to-one correspondence with one another.

Contact

A Contact is the identification of a person, role, or other entity within an organization that has some relationship to a registered object. A contact will consist of a common name, 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 relationships among organizations, contacts, and registered objects. In the Registry Information Model, this identifier is said to be of type CONTACT. Each contact includes a mandatory reference to some organization. It is not necessary to maintain a global, unique name for contacts because the global name for the organization together with the common name for the contact will be sufficient to uniquely identify the contact.

Classification

A classification is a partition of a given collection of items into mutually exclusive and collectively exhaustive sub-collections. A classification depends upon a pre-existing specification of a hierarchy of values, names, and codes called a classification scheme. Registry items in a Registry may be classified by as many classification schemes as deemed appropriate by the Submitting Organization.

Classification Scheme

A classification scheme is a hierarchy of values that can be referenced by a classification. A classification scheme can vary from a simple set of values to a complex multi-level hierarchy. An example of a simple single-level classification is the set {Freshman, Sophomore, Junior, Senior} used to partition a collection of students. An example of a more complicated classification scheme is one based on the hierarchy of all living things with named levels for Kingdom, Phylum, Class, Order, Family, Genus and Species. The Classifications section below defines the various types of classification schemes and specifies XML representations for defining classification schemes and exchanging classifications of items dependent upon that scheme.

Package

A Package is a conceptual notion used to identify a set of registered objects. It is defined to be a registered object that is a set of pointers to other registered objects. Using this definition, a package can represent a hierarchy of registered objects, where non-terminal nodes of the hierarchy are other packages and terminal nodes are package or non-package objects. A package is a terminal node in a package hierarchy if and only if the package is empty. A registered object may be pointed to by several different packages. A package relationship between a registered package and some other registered object pointed to by a package element is represented by the *contains* role in an association instance.

Since the representation of a registered object is defined to be a file, the file representing a package object is an XML document that conforms to the OASIS Package DTD defined elsewhere in this specification. Only the owner of a package can modify its contents. Rules for package definition and package maintenance are given in the Entity Semantics section below.

Alternate Name

Alternate Name is a conceptual category used to group together alternate names used for local or global reference to a registered object. Especially significant are names used in special circumstances or contexts, e.g. short names consisting of all visible characters from a limited character set for local identifiers in a specific programming language context, or globally unique qualified names that satisfy the hierarchical qualification structure of a specific context. Alternate names can also be used for names in different human languages with characters encoded in unusual character sets.

An alternate name instance consists of the alternate name, a context for using that name, and a human readable comment that further explains how the name should be used. An alternate name instance is tagged with a language code and a character set code that apply to all other information in that instance.

Alternate Description

Alternate Description is a conceptual category used to group together alternate descriptions, in different human languages, of a registered object. An alternate description instance is tagged with a human language code and a character set code that apply to the text of the description.

Submission

A submission is a collection of requests, in the form of a message, sent from a Submitting Organization to a Registration Authority. For administrative accountability the Registration Authority logs each submission into the Registry information model, timestamps it with the date and time received, and stores it for potential subsequent scrutiny. Any Registry entities created, deleted, or modified by a submission should be traceable back to the submission instance and to its Submitting Organization.

Each submission instance has a non-public, local identifier created by the Registration Authority that is used to maintain its relationships among organizations, registry items, contacts, and other registry entities. In the Registry information model, this identifier is said to be of type SUBMISSION. Each submission includes a mandatory reference to a submitting Organization and to some Contact within that organization.

Query

A query is a message from a public user of a registry database to a registry, asking that certain information be returned. A request is generally sent in the form of an XML document that validates to one of the XML query DTD's defined elsewhere in this specification. The response to a query will validate to the associated XML response DTD.

A public user only has access to public information in the registry. This includes all of the information in the Public View of the registry, portrayed by the diagram in Section X.2.1, as well as the Contact, Organization, and Impact information presented in Section X.2.2, the

classification hierarchy information presented in Section X.2.3, and the code information presented in Section X.2.5. It does not include the administrative information maintained by the registration authority for its own purposes.

Request

A request is a message sent from a Submitting Organization to a Registration Authority asking that certain additions or modifications be made to the Registry. A request is generally sent in the form of an XML document that validates to one of the request DTD's defined elsewhere in this specification. A request instance will consist of a request code to identify the type of request as well as the XML content of a specific request.

The kinds of requests that will supported by an OASIS conformant Registry are still under development, but will likely include the following:

- register a new object,
- register a package of objects,
- add to or modify the associations of a registered object,
- add to or modify the classifications of a registered object,
- add to or modify the alternate names or alternate descriptions of a registered object,
- certify a company or organization as a Submitting Organization,
- modify the content of a previous submission,
- withdraw a previously registered object,
- supersede or replace an existing registered object.

Impact

An impact is a many-to-many relationship between Requests and Registry Items. Each request may have an impact on one or more registered objects. A supersede request impacts both the superseded item and the new item that supersedes it. The addition of an association impacts both the given item and the associated item.

The Impact concept will maintain a historical record of all SO-initiated modifications of a registry item from first creation to final deletion.

X.2 Registry Model

The OASIS Registry information model is an Entity-Attribute-Relationship (EAR) model that defines the logical structure of an OASIS Registry. Other parts of the OASIS specification, especially query messages sent to a Repository and answer responses from a Repository, will be defined in terms of this model. Unless otherwise indicated, each concept described above is an Entity in the model, each information item associated with a concept is an Attribute, and each association between concepts is a Relationship.

In the figures of this section, a rectangle represents an entity and lines between entity rectangles represent relationships. The attributes that define an entity are listed as rows within the entity rectangle. For each attribute, the first column of the row is its name, the second column is its datatype, and the third column is an indication of whether or not it is mandatory.

Each entity in the EAR model represents both the data structure of a single entity instance and a container that holds a collection of persistent entity instances. The attributes of an entity determine the structure of a single instance and the relationships between entities determine pairwise associations among individual entity instances as well as constraints on collections of persistent instances.

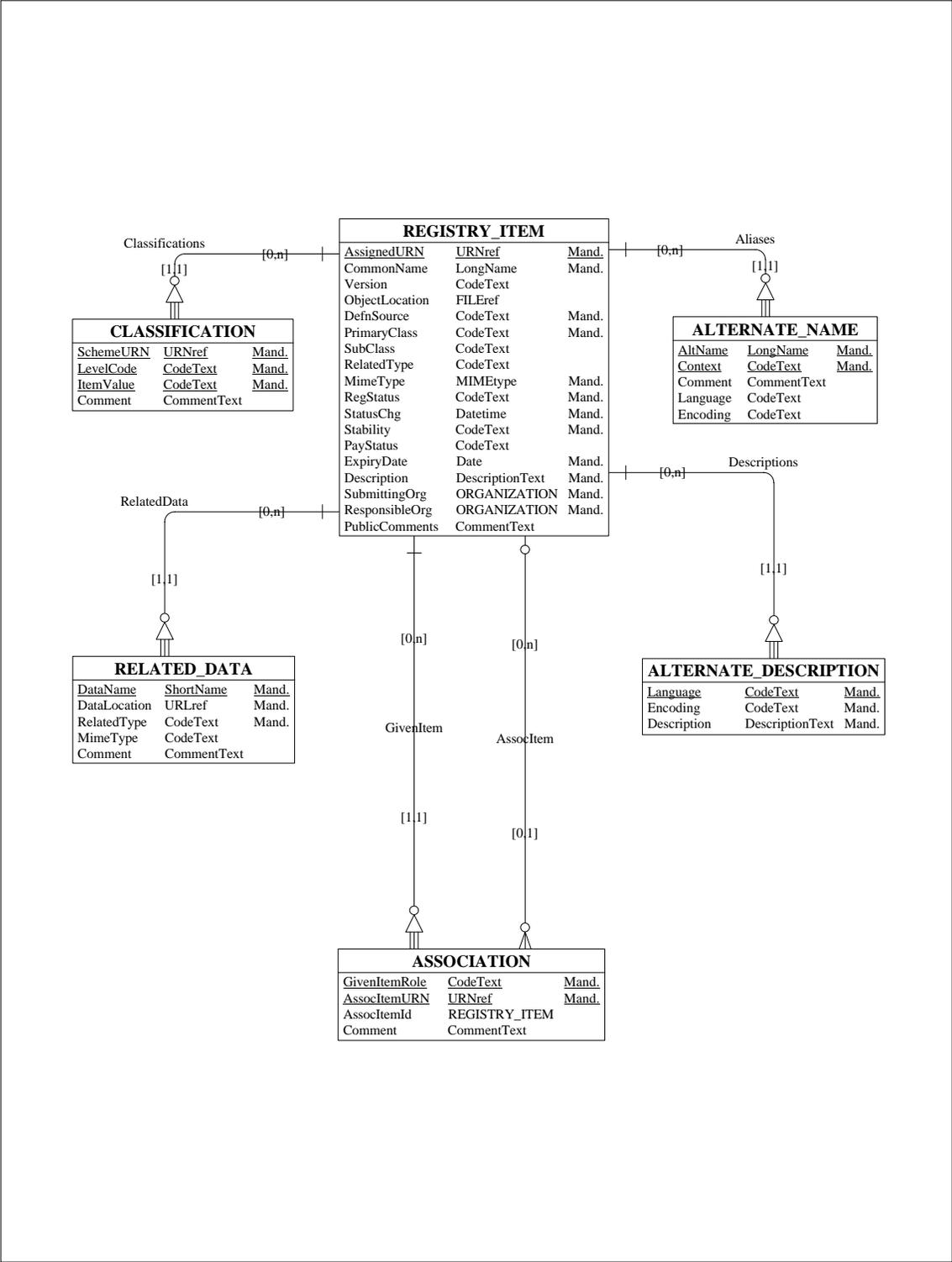
Every relationship is bi-directional, always interpretable as from one entity to another, and vice versa. Each direction can have a *cardinality* of zero-to-many, one-to-many, zero-to-one, or one-to-one. We do not allow relationships to be many-to-many; instead, we define a new entity with two different relationships that satisfy the simpler assumptions, e.g. the Association entity. A zero-to-one or zero-to-many cardinality of a directional relationship indicates *optionality*, i.e. an instance at the from-side may not map to any instance at the to-side. In the diagrams, a small circle at the to-side of a directional relationship represents such optionality, as does the notation [0,1] or [0,n] at the from-side. A zero-to-many or one-to-many cardinality of a directional relationship indicates *multiplicity*, i.e. an instance at the from-side may map to multiple instances at the to-side. In the diagrams, triple feet at the to-side of a directional relationship represent such multiplicity, as does the notation [0,n] or [1,n] at the from-side. If a directional relationship is one-to-many, then an instance at the from-side requires the existence of one or more instances at the to-side. If a directional relationship is one-to-one, then an instance at the from-side requires exactly one instance at the to-side; this means that the from-side instance has a *dependency* on that to-side instance.

Dependency can be handled in several different ways. Suppose x and y are entity instances and that y has a dependency on x. If x is deleted, what happens to y? If y is also deleted, then we say that the dependency has a *cascade delete* effect. If the deletion of x is prohibited, then we say that the dependency has a *cascade restrict* effect. If y is somehow modified to remove its dependency on x, then we say that the relationship has a *cascade modify* effect. In the diagrams, a triangle at the from-side of a directional relationship represents a dependency with a *cascade delete* effect.

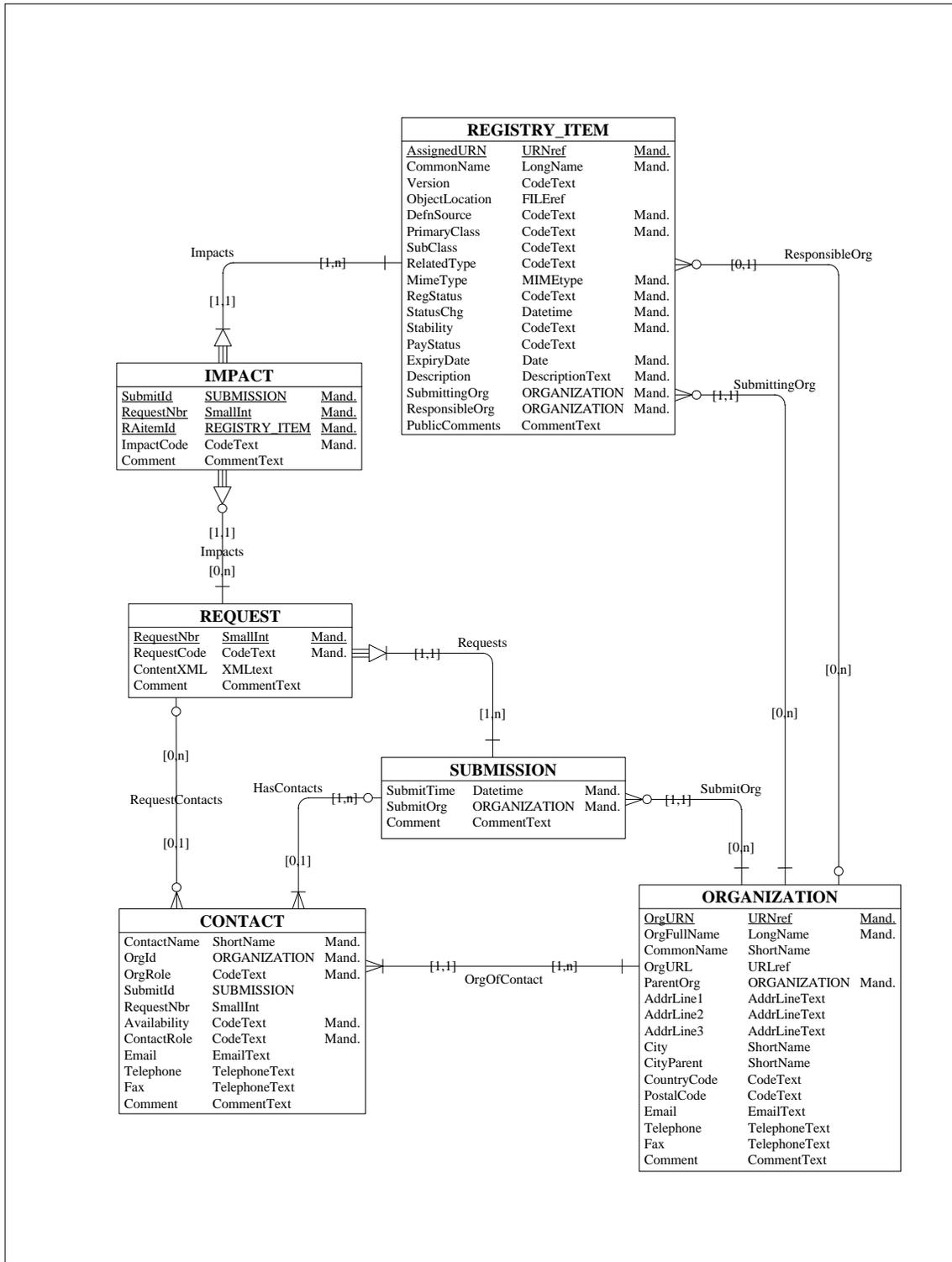
Underlined Attributes indicate *uniqueness*. If an entity is not dependent upon any other entity, then the underlined attributes uniquely determine an instance of that entity. If an entity is dependent upon some other entity, then for each instance of the other entity, the underlined attributes determine a unique instance of the given entity.

This restricted use of the Entity-Attribute-Relationship model ensures that its usage is simple enough to be consistently transferable to representations in a number of other data modeling paradigms, including relational, object-oriented, UML, or MOF. The diagrams on the next few pages give a high-level view of the EAR model of the OASIS Registry. Later sections add more detail to attribute types and specify the Semantic Rules that must be satisfied.

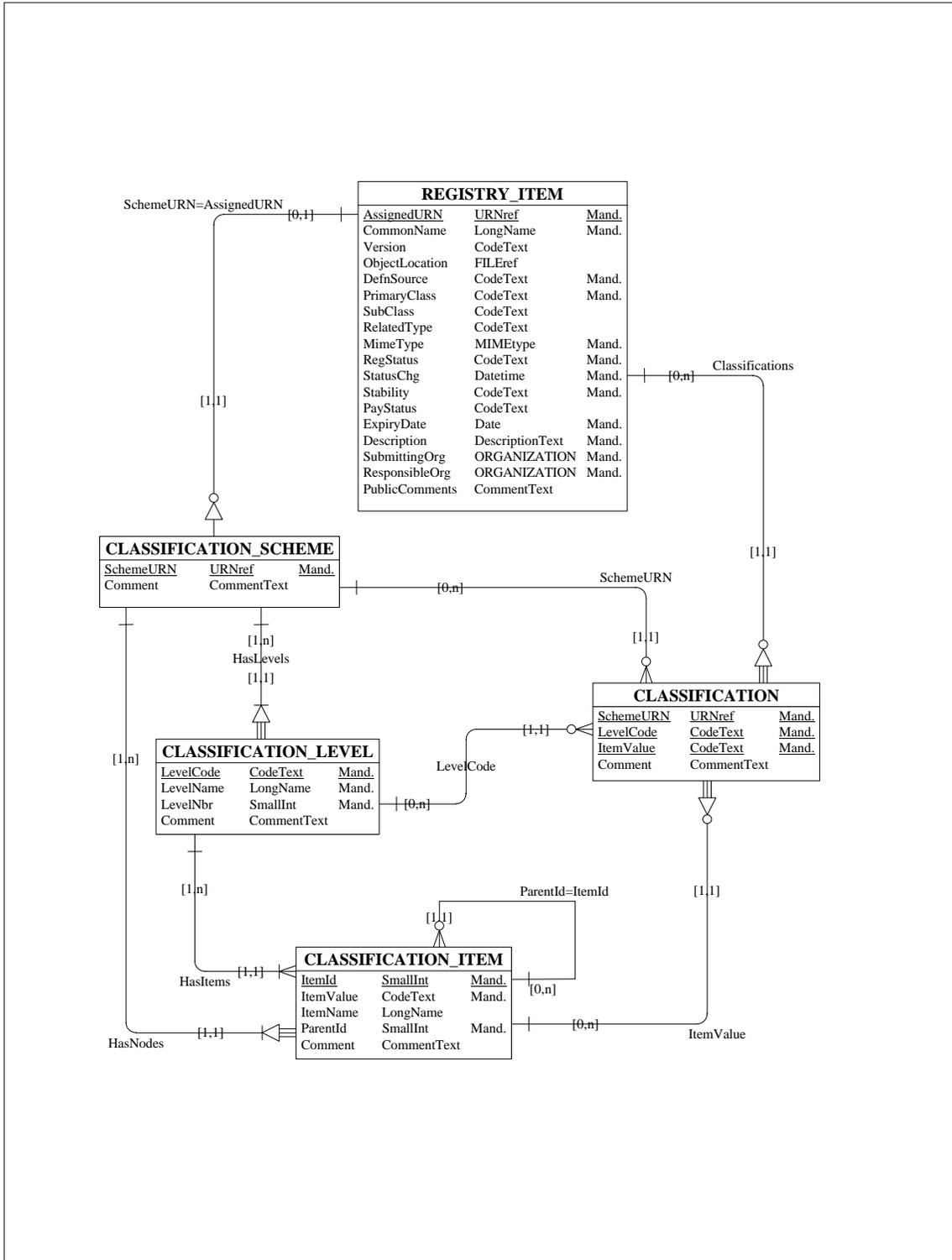
X.2.1 Registry Model - Public View



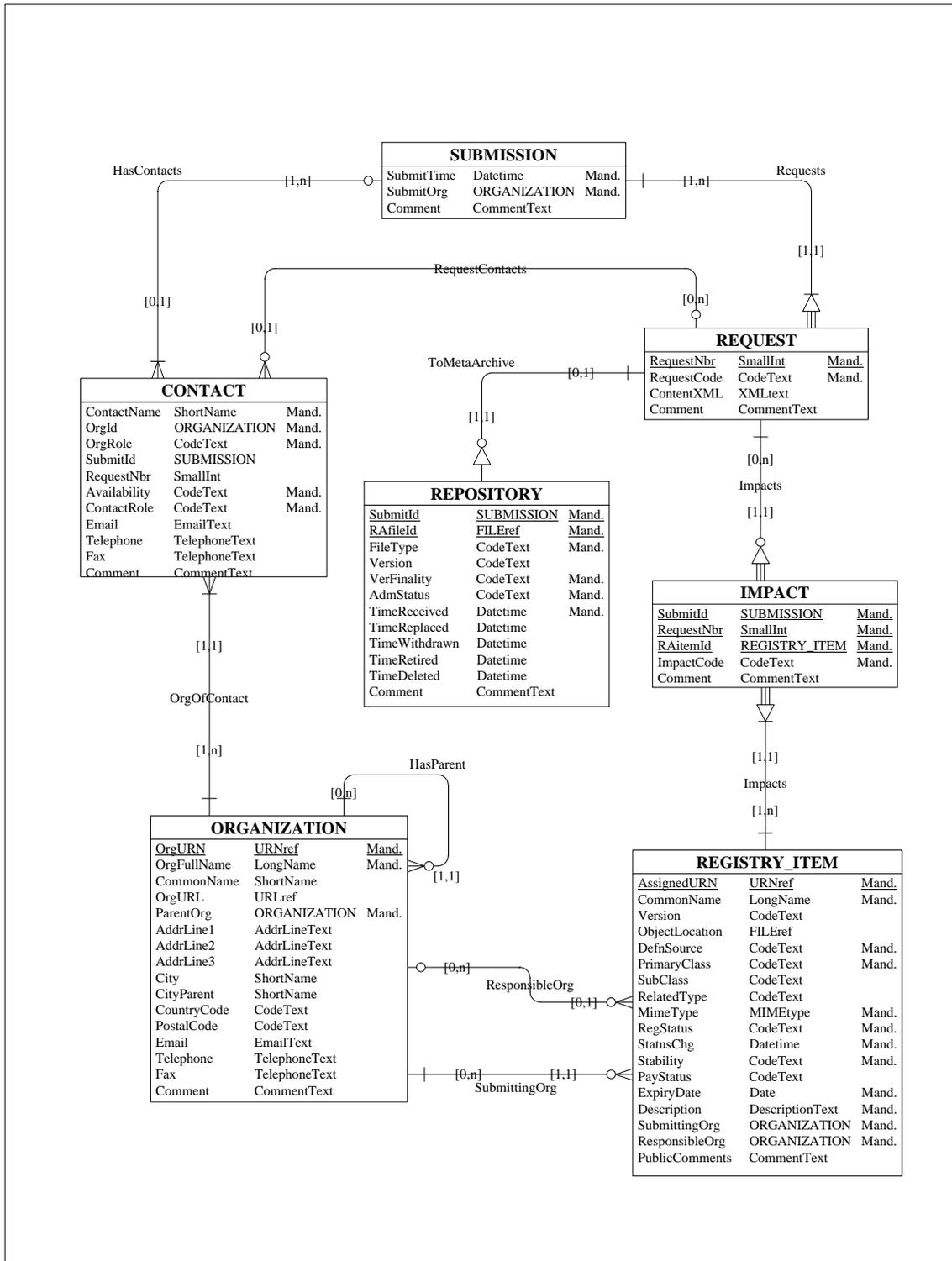
X.2.2 Registry Model – Contacts View



X.2.3 Registry Model – Classification View



X.2.4 Registry Model – Administration View



X.2.5 Registry Model – Enumeration Entity View

DefinitionSource		
SourceCode	CodeText	Mand.
SourceName	LongName	Mand.
Description	TXT	

RegistrationStatus		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

AssociationType		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

RelatedDataType		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

PrimaryClassification		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

SubClassification		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

PaymentStatus		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

Stability		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

ContactAvailability		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

ContactRole		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

OrganizationRole		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

NameContext		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

RegistryRequest		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

RegistryImpact		
Source	CodeText	Mand.
Code	CodeText	Mand.
Name	LongName	Mand.
Description	CommentText	

New Enumerated Entity types above.
Original lists below.

REG_STATUS_LIST		
RSLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

DOC_GENRE_LIST		
DGLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

ADMIN_STATUS_LIST		
ASLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

SEMANTIC_ENTITY_LIST		
SELcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

LITERARY_GENRE_LIST		
LGLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

OTHER_XSGML_LIST		
OXLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

SUBMISSION_PURPOSE_LIST		
SPLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

VERSION_FINALITY_LIST		
VFLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

STYLE_SHEET_LIST		
SSLcode	EntityCode_Domain	Mand.
OASISname	OASIS_Name_Domain	Mand.
Description	DescriptionText	

X.3 Classifications

A classification is a partition of a given collection of items into mutually exclusive and collectively exhaustive sub-collections. A classification depends upon a pre-existing specification of a hierarchy of values, names, and codes called a classification scheme. Registry items in a Registry may be classified by as many classification schemes as deemed appropriate by the Submitting Organization.

A classification scheme can be one of several different styles, or some combination of those styles. The following paragraphs describe three separate styles of classification schemes.

Simple 1-level classification scheme

A simple 1-level classification scheme is a list of distinct values that can be used to partition a collection of objects. The name of the classification scheme can be viewed as the root of the hierarchy, with each of the distinct values considered as a node at the first level. An example of such a simple classification scheme is the StudentStatus attribute of a student population, which can be used to partition the students into Freshmen, Sophomores, Juniors, Seniors, and Special students.

Each node of a classification scheme will allow distinctions to be made between ItemValue and ItemName. An ItemValue will always be considered as a reference to be used in place of the ItemName. With this capability, we could refine the StudentStatus classification scheme to specify the item values FR, SO, JR, SR, SP as references and replacements for the longer item names. A classification scheme defines only one item value for each item name.

Multi-level naming classification scheme

A multi-level naming classification scheme uses scoped names to identify the nodes in a classification hierarchy. Each name is meaningful only if the name of its parent node is known. Names need be unique only as children of the parent node, so in order to uniquely identify a node it is necessary to know the names of each node in a path from the root to the given node.

An example of a multi-level naming classification scheme is the scheme used by biologists to classify all living things. The scheme consists of seven levels: Kingdom, Phylum, Class, Order, Family, Genus, and Species, each with a list of recognized values. However, it is not required that names within each level be unique; there could be a species for tree that is the same as the species of some animal, because trees and animals are in different kingdoms, and thus in different branches of the hierarchy. A complete classification of all living things would thus require a value for each of the seven levels.

A classification scheme can be used for classification of a population taken from just one of its nodes. For example, since primate is an instance of Order, a classification of all primates could consist of just three values, one for each of the levels Family, Genus, and Species. Similarly, a classification of all modern-day-trees could consist of values for just Genus and Species.

In using a naming classification scheme for classification of a given population, it is usually necessary to identify the following items: 1) a globally unique name for a classification scheme, 2) a unique identifier for each level within the scheme, and 3) a value for each level. This is easily accomplished by defining a classification to be a set of ordered triples, {(SchemeURN, LevelCode, ItemValue)}, where SchemeURN is the unique name of a classification scheme, LevelCode is a short name used to reference a longer and possibly more descriptive LevelName, and ItemValue is a short name or value used to represent a longer ItemName. Or, to eliminate the repetition of SchemeURN, and to allow optional inclusion of the full names, one could define a classification as the following XML element definition:

```
<!ELEMENT classification (levelValuePair+)>
<!ATTLIST classification
    schemeURN CDATA #REQUIRED
    schemeName CDATA #IMPLIED >

<!ELEMENT levelValuePair (comment-text?)
<!ATTLIST levelValuePair
    levelCode CDATA #IMPLIED "Leaf"
    itemValue CDATA #REQUIRED
    levelName CDATA #IMPLIED
    levelNbr CDATA #IMPLIED
    itemName CDATA #IMPLIED >

<!ELEMENT comment-text (#PCDATA)>
```

Multi-level coded classification scheme

A multi-level coded classification scheme uses a string of codes to represent a path down the classification scheme hierarchy. Each node has a code that is unique under its parent; then, in an N-level hierarchy, a sequence of node codes from Level 1 to Level N uniquely determines a definition path through the tree. As above, each code is considered to be an ItemValue that represents an ItemName. As in the named classification scheme, item names do not need to be unique. However, the codes are chosen so that it is convenient to represent the path from the root to the given node as a short string of codes.

In a coded classification scheme, the LevelCode is defined to be the sequence of item codes from the root to the given node. Then a classification using this scheme need only supply a value for a single node. The values for each item in the path can be inferred from the sequence, thereby identifying the name for each item in the path.

An example of a coded classification scheme is one for newspaper articles that uses a 3-level scheme with 2 digits to identify Level-1 and three digits each to identify Level-2 and Level-3. The coded value "15052003" thus represents a named classification path as Sport (15), followed by Ski Jumping (052), followed by K180 Flying Jump (003).

In coded classifications, if the structure of the code path is known, i.e. 2:3:3 digits for the three levels, then the level name is unimportant. An application receiving the levelValue “15052003” would know to break it up into three codes 15, 52, and 3 to retrieve the three item names for the item values “15”, “15052”, and “15052003”. If the code structure is not known, then it could be implied by using a separator between the codes in the path, e.g. “15:052:003”, or one could begin with the leaf node and successively find each parent node until reaching the root. In any classification scheme, we will always assume that if any node of the scheme is known, then the sequence of parent nodes, and their associated level names, can always be determined.

In coded classifications where the code structure is not known, it is convenient to have a default code for each level, e.g. Level1, Level2, etc., so that the person or application receiving the coded item value knows how many itemNames to look for.

Mapping to the information model

The diagram in Section X.2.3, “Classification View”, gives the information model for representing classification schemes and classifications. The CLASSIFICATION entity holds the classifications for any registered item. Each classification will identify a classification scheme by a SchemeURN, even though that URN may be from some other registry. Since each classification scheme referenced by a classification is registered in the local registry, the URN should identify an instance of the ALTERNATE_NAME entity, which points to a unique instance of REGISTRY_ITEM, which in turn identifies a unique instance of CLASSIFICATION_SCHEME. The description of the scheme and all of its metadata can then be retrieved from the registry.

The CLASSIFICATION_LEVEL entity identifies each level of the classification scheme. If it is a simple 1-level scheme, then there is only one level and its LevelName is unimportant. By default, the level name and level code of the single level of a simple classification scheme is “Leaf”, since each value will represent a leaf node in the 1-level hierarchy. Similarly, if no level names are defined for a multi-level coded classification scheme, then default level names will be Level1, Level2, etc., and the default level codes will be the same as the level numbers, i.e. 1,2,3, etc. If a single ItemValue is sufficient to identify the whole path leading from the root node to the identified node, then the keyword “Leaf” can be used as an alias for whatever the assigned level code of that item might be. For example, in the newspaper article example above, the 3rd level leaf value “15052003” could be transmitted as the triple (“NewsArticle”, “Leaf”, “15052003”) and the three associated level names could be derived from the scheme definition.

The CLASSIFICATION_ITEM entity identifies each node in the classification hierarchy. Since the item names need not be unique, the local registry will define a local ItemId for each node, so that the child-to-parent relationships of the nodes can be easily represented. Then given a “Leaf” node value, and no other information, the registry can follow the child-to-parent hierarchy from the given node to the root of the hierarchy to retrieve the entire name sequence for that node.

A constraint on classifications is that each referenced classification scheme must identify a registered item in the same Registry as the classification instance. Then the locator of the identified registry item for the classification scheme can be used to locate a specific, OASIS

conformant classification hierarchy in this or in some other Repository. The classification hierarchy must be defined by and transmitted by an XML document that validates to the XML ClassificationScheme DTD defined in this specification.

Candidate Classification scheme DTD

```
<!ELEMENT ClassificationScheme
  (
    comment-text?,
    scheme-level*,
    scheme-node+
  )>
<!ATTLIST ClassificationScheme schemeURN CDATA #REQUIRED>

<!ELEMENT scheme-level (comment-text?)>
<!ATTLIST scheme-level
  levelCode CDATA #REQUIRED
  levelName CDATA #IMPLIED
  levelNbr CDATA #IMPLIED >

<!ELEMENT scheme-node
  (
    scheme-item,
    scheme-node*
  ) >

<!ELEMENT scheme-item (comment-text?)>
<!ATTLIST scheme-item
  itemValue CDATA #REQUIRED
  itemName CDATA #IMPLIED
  levelNbr CDATA #IMPLIED
  levelCode CDATA #IMPLIED
  levelName CDATA #IMPLIED >

<!ELEMENT comment-text (#PCDATA)>
```

Example scheme DTD

```
<!--
A simple named and coded classification
for Student Status in a University
```

```
FR   Freshman
SO   Sophomore
JR   Junior
SR   Senior
SP   Special
```

```
Subclassification of Special student
```

D Degree seeking
N Non-degree seeking
-->

```
<ClassificationScheme schemeURN="gov:nist:StudentStatus">
<comment-text>
A named and coded classification scheme for student status.
With Special student status further subclassified.
</comment-text>
<scheme-level levelNbr="1" levelCode="Primary"/>
<scheme-level levelNbr="2" levelCode="Secondary"/>
<scheme-node>
<scheme-item itemValue="FR" itemName="Freshman"/>
</scheme-node>
<scheme-node>
<scheme-item itemValue="SO" itemName="Sophomore"/>
</scheme-node>
<scheme-node>
<scheme-item itemValue="JR" itemName="Junior"/>
</scheme-node>
<scheme-node>
<scheme-item itemValue="SR" itemName="Senior"/>
</scheme-node>
<scheme-node>
<scheme-item itemValue="SP" itemName="Special"/>
  <scheme-node>
    <scheme-item itemValue="D" itemName="Degree"/>
  </scheme-node>
  <scheme-node>
    <scheme-item itemValue="N" itemName="Non-degree"/>
  </scheme-node>
</scheme-node>
</ClassificationScheme>
```

Example classification XML

```
<classification schemeURN="gov:nist:StudentStatus">
<itemLevelPair
  levelCode="Primary" itemValue="SP"/>
<itemLevelPair
  levelCode="Secondary" itemValue="N"/>
</classification>
```

X.4 Attribute Types

Attribute values in the information model will be one of the following types:

- Entity References
- Base Types

Some attribute values will be references to entity instances and some will be primitive types representable as character strings, numbers, dates, or dates and times. We have already identified entity references to be one of the following types:

REGISTRY_ITEM
ORGANIZATION
CONTACT
SUBMISSION

To this list we add the Enumeration Entities defined below in Section X.5.

The following definitions identify the base types that will be 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 less than or equal to 50 characters. In XML environments, CodeText may not contain XML characters with special meaning. These include the ampersand (&), etc.

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 less than or equal to 50 characters.

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 less than or equal to 150 characters.

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 less than or equal to 50 characters.

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 less than or equal to 50 characters.

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 less than or equal to 50 characters.

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 less than or equal to 250 characters.

DefinitionText -- 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 DefinitionText string is less than or equal to 5000 characters.

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 items 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.

SmallInt -- A non-negative integer with value less than 2^{16} .

URNref -- a character string that conforms to the format of a Uniform Resource Name (URN) as specified by IETF RFC 1241. The length of a URNref string is less than or equal to 150 characters.

(See <http://www.ietf.cnri.reston.va.us/rfc/rfc2141.txt?number=2141>)

URLref -- a character string that conforms to the format of a Uniform Resource Locator (URL) as specified by W3C. The length of a URLref string is less than or equal to 150 characters.

(See http://www.w3.org/Addressing/URL/5_BNF.html)

FTPref -- 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 FTPref string is less than or equal to 150 characters.

(See http://www.w3.org/Addressing/URL/5_BNF.html)

FILEref -- a character string that is a URLref or an FTPref.

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). 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>)

LanguageId -- 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. 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>).

X.5 Enumeration Entities

Many of the attributes declared to be of type CodeText will have an additional constraint that the CodeText value match a specific value from a pre-defined list of values. The Registry information model represents such lists as entities with a fixed number of entity instances. We define such entities to be enumeration entities.

The paragraphs of this section define the structure and specify the contents of each enumeration entity for OASIS.

DefinitionSource

SourceCode	SourceName	Description
EBXML		Author of the EBXML Registry/Repository specification.
IEEE_LOM	IEEE Learning Technology - Learning Object Model	Author of the IEEE LOM Registry specification.
IMS		Author of the IMS Registry specification.
OASIS	Organization for the Advancement of Structured Information Standards	Author of the OASIS Registry/Repository specification.

PrimaryClassification

Source	Code	Name	Description
OASIS	defn	Definition	An XSGML definition document.
OASIS	inst	Instance	An XSGML instance document.
OASIS	pkg	Package	A package of OASIS registered items.
OASIS	other	Other	A document related to a registered item.

SubClassification

Source	Code	Name	Description
OASIS	CharEntSet	character-entity-set	
OASIS	RDFschema	rdf-schema	
OASIS	SGMLattrib	sgml-attribute	
OASIS	SGMLattSet	sgml-enumerated-attribute-set	
OASIS	SGMLattVal	sgml-enumerated-attribute-value	
OASIS	SGMLdtd	sgml-dtd	
OASIS	SGMLElem	sgml-element	
OASIS	SGMLparm	sgml-parameter-entity	
OASIS	SOXschema	sox-schema	
OASIS	XDRschema	xdr-schema	
OASIS	XMLattrib	xml-attribute	
OASIS	XMLattSet	xml-enumerated-attribute-set	
OASIS	XMLattVal	xml-enumerated-attribute-value	
OASIS	XMLdtd	xml-dtd	
OASIS	XMLelem	xml-element	
OASIS	XMLparm	xml-parameter-entity	
OASIS	XMLschema	xml-schema	

RelatedDataType

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	

RegistrationStatus

Source	Code	Name	Description
OASIS	exp	Expired	
OASIS	reg	Registered	
OASIS	rep	Replaced	
OASIS	sub	Submitted	
OASIS	sup	Superseded	
OASIS	wth	Withdrawn	

Stability

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.

PaymentStatus

Source	Code	Name	Description
OASIS	debt	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.

AssociationType

Source	Code	Name	Description
OASIS	contains	Contains	Given item is a package that contains the associated item.
OASIS	related	Related	Given item is related to associated item and provides supplemental information for the associated item.
OASIS	supersedes	Supersedes	Given item supersedes associated item.
OASIS	uses	Uses	Given item uses associated item.

ContactAvailability

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

ContactRole

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

NameContext

Source	Code	Name	Description
OASIS	c++	C++ Name	For C++ usage
OASIS	code	Code Name	Without spaces
OASIS	java	Java Name	For Java usage
OASIS	lang	Language Name	In a human language
OASIS	long	Long Name	1 to 150 characters
OASIS	short	Short Name	1 to 50 characters
OASIS	sql	SQL Name	For SQL usage
OASIS	urn	Uniform Resource Name	Globally unique

RegistryImpact

Source	Code	Name	Description
OASIS	DAD	Delete Alternate Description	
OASIS	DAN	Delete Alternate Name	
OASIS	DAS	Delete Association	
OASIS	DCF	Delete Classification	
OASIS	DCT	Delete Contact	
OASIS	DRD	Delete Related Data	
OASIS	DRI	Delete Registry Item	
OASIS	IAD	Insert Alternate Description	
OASIS	IAN	Insert Alternate Name	
OASIS	IAS	Insert Association	
OASIS	ICF	Insert Classification	
OASIS	ICT	Insert Contact	
OASIS	IRD	Insert Related Data	
OASIS	IRI	Insert Registry Item	
OASIS	UAD	Update Alternate Description	
OASIS	UAN	Update Alternate Name	
OASIS	UAS	Update Association	
OASIS	UCF	Update Classification	
OASIS	UCT	Update Contact	
OASIS	URD	Update Related Data	
OASIS	URI	Update Registry Item	

RegistryRequest

Source	Code	Name	Description
OASIS	certSO	Certify Submitting Org	
OASIS	defClf	Define Classification Scheme	
OASIS	defPkg	Define Package	
OASIS	modClf	Modify Classification Scheme	
OASIS	modPkg	Modify Package	
OASIS	modReg	Modify Registration	
OASIS	regObj	Register Object	
OASIS	repReg	Replace Registered Object	
OASIS	supReg	Supersede Registration	
OASIS	wdrReg	Withdraw Registration	
OASIS	x etc.	Other	

X.6 Entity Semantics

X.6.1 REGISTRY_ITEM

Structure

Attribute Name	Attribute Type	Presence
AssignedURN	URNref	Mandatory
CommonName	ShortName	Mandatory
Version	CodeText	
ObjectLocation	FILEref	
DefnSource	CodeText	Mandatory
PrimaryClass	CodeText	Mandatory
SubClass	CodeText	
RelatedType	CodeText	
MimeType	MIMETYPE	Mandatory
RegStatus	CodeText	Mandatory
StatusChg	Datetime	Mandatory
Stability	CodeText	Mandatory
PayStatus	CodeText	Mandatory
ExpiryDate	Date	Mandatory
Description	DescriptionText	Mandatory
SubmittingOrg	ORGANIZATION	Mandatory
ResponsibleOrg	ORGANIZATION	Mandatory
PublicComments	CommentText	

Semantic Rules

1. The RegistryItem entity represents the set of all registered objects in the Registry. Each instance identifies a single registered object. A registry item instance holds only some of the metadata for a registered object; other metadata is held by other entities in the Registry.
2. Each registry item instance is assigned a unique identifier by the Registration Authority (RA). This implicit value is said to be of type REGISTRY_ITEM. It is used to represent relationships of this instance with other information in the Registry.
3. The AssignedURN name is created and assigned by the RA. It is created to be unique within all OASIS conforming Registry/Repository implementations. When a Submitting Organization (SO) makes a submission to the Registry, it provides a local reference name of type CodeText. If possible, the RA uses that name to construct the AssignedURN.

4. The `CommonName` is provided by the SO.
5. The `Version` is provided by the SO. It can have an arbitrary format and is used only to help distinguish one registry item from another having the same common name. The `AssignedURN` will be different for different versions.
6. The `ObjectLocation` 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 download the item, store it in the Repository, and create an anonymous FTP locator as a value for `ObjectLocation`. If the Registry is not also a Repository, then the `ObjectLocation` is provided by the SO and the RA has no further responsibility. Depending on the value of the `PayStatus` attribute, the `ObjectLocation` 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 entity that 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 `PrimaryClass` is provided by the SO and takes its value from the `PrimaryClassification` enumeration entity. If the `DefnSource` is OASIS, then `PrimaryClass` identifies an element of the set {`Definition`, `Instance`, `Package`, `Other`}.

Let G be a registry item whose `DefnSource` is OASIS.

Case:

- a) If the `PrimaryClass` of G identifies `Definition`, then the Submitting Organization declares that the registered object identified by G is an xsgml definition object. The `SubClass` of G will identify the specific kind of definition object it is.
- b) If the `PrimaryClass` of G identifies `Instance`, then the Submitting Organization declares that the registered object identified by G is an instance of some registered xsgml definition object. The `SubClass` of G identifies its type. There must exist a registry item A in the same registry as G satisfying the following conditions: A is the associated item in a *uses* association with G as the given item, `A.PrimaryClass = Definition`, `A.SubClass = G.SubClass`, and `A.RegStatus=Registered`.
- c) If the `PrimaryClass` of G identifies `Package`, then the Submitting Organization declares that the registered object identified by G is an OASIS package, subject to the OASIS specific rules for creating, populating, manipulating, and transmitting packages.
- d) If the `PrimaryClass` of G identifies `Other`, then the Submitting Organization declares that the registered object identified by G is not an OASIS package and has a representation other than as an xsgml definition object or an xsgml instance object.

9. The SubClass is provided by the SO and takes its value from the SubClassification enumeration entity. If the DefnSource is OASIS, then SubClass, if it exists, identifies an element of the set {char-entity-set, rdf-schema, sgml-attribute, sgml-enum-attr-set, sgml-enum-attr-value, sgml-dtd, sgml-element, sgml-param-entity, sox-schema, xdr-schema, xml-attribute, xml-enum-attr-set, xml-enum-attr-value, xml-dtd, xml-element, xml-param-entity, xml-schema}.

Let G be a registry item whose DefnSource is OASIS.

Case:

- a) If the PrimaryClass of G identifies Definition, then G.SubClass exists and G is a definition object of the identified type.
 - b) If the PrimaryClass of G identifies Instance, then G.SubClass exists and G is an instance object of the identified type.
 - c) If the PrimaryClass of G identifies Package or Other, then G.SubClass is undefined.
10. The RelatedType is provided by the SO and takes its value from the RelatedDataType enumeration entity. If the DefnSource is OASIS, then RelatedType, if it exists, identifies an element of the set {reference-manual, user-guide, white-paper, faq, example, example-set, example-set-information, changelog, readme, email-discussion-list-information, tool-information, schema-home-page, distribution-home-page, registration-information, other}.

Let G be a registry item whose DefnSource is OASIS. If G.RelatedType exists, then there must exist a registry item A in the same registry as G where A is the associated item in a *related* association with G as the given item.

11. The MimeType is provided by the SO. If G is a registry item whose DefnSource is OASIS, then:
Case:

- a) If the PrimaryClass of G identifies Definition, Instance, or Package, then G.Representation must be a MIME text type. By default the type is text/plain.
 - b) If the PrimaryClass of G identifies Other, then G.Representation can be any MIME type.
12. The RegStatus is provided by the RA with its value taken from the RegistrationStatus enumeration entity. For OASIS registrations, that entity includes the values {Submitted, Registered, Superseded, Replaced, Withdrawn, Expired}. The StatusChg attribute is the datetime that the RA last approved a change for RegStatus.

Let G be a registry item whose DefnSource is OASIS.

Case:

- a) If the RegStatus of G identifies Submitted, then G and its associated metadata is not visible to anyone other than the SO and the RA.
 - b) If the RegStatus of G identifies Registered, then the RA has approved for registration the registered object identified by G. If the RA is also a Repository, then the approval process includes checking the registered object to ensure that it validates as the xsgml item identified by G.PrimaryClass and G.SubClass and that the G.MimeType declaration is appropriate.
 - c) If the RegStatus of G identifies Superseded, then the RA has received and acted upon a request from the SO that submitted G to supersede it with a new registered object. The original registered object remains available for continued use.
 - d) If the RegStatus of G identifies Replaced, then the RA has received and acted upon a request from the SO that submitted G to replace it with a new registered object. The original registered object is no longer available, although its metadata will remain in the registry for an indefinite period of time. G.ObjectLocation will now locate the new object.
 - e) If the RegStatus of G identifies Withdrawn, then the RA has received and acted upon a request from the SO that submitted G to withdraw it. The original registered object is no longer available, although its metadata will remain in the registry for an indefinite period of time. G.ObjectLocation will now have no value.
 - f) If the RegStatus of G identifies Expired, then G.ExpiryDate has passed with no reaffirmation action from the SO. The original registered object may or may not be available, although the registry item G and other relevant metadata will remain in the registry for an indefinite period of time.
13. The Stability attribute is provided by the SO with its value taken from the Stability enumeration entity. For OASIS registrations, that entity includes the values {Static, Dynamic, Compatible}.

Let G be a registry item whose DefnSource is OASIS.

Case:

- a) If the Stability of G identifies Static, then the SO declares that the registered object identified by G will not be replaced or withdrawn before the ExpiryDate. It is the responsibility of the RA to ensure as best it can that this declaration is not violated.
- b) If the stability of G identifies Dynamic, then the SO declares that the registered object identified by G may change without notice, possibly in incompatible ways.
- c) If the stability of G identifies Compatible, then the SO declares that the registered object defined by G will not be replaced with an incompatible object. If the registered object is a DTD, then new element and attribute declarations may be added to the DTD, but a

document that validates to the original DTD will also validate to its replacement. The notion of *compatibility* needs to be examined for all xsgml definition and instance objects and for OASIS packages!

14. The PayStatus attribute is provided by the SO with its value taken from the Payment_Status enumeration entity. For OASIS registrations, that entity includes the values {Free, Password, Debit}. It is expected that these options will expand as online payment alternatives become standardized.

Let G be a registry item whose DefnSource is OASIS.

Case:

- a) If the PayStatus of G identifies Free, then the URL of G.ObjectLocation follows an anonymous protocol that allows retrieval of the registered object with no password or payment obstructions.
 - b) If the PayStatus of G identifies Password, then access to the object is free, but a password is required. The procedure for obtaining a password is not part of this specification.
 - c) If the PayStatus of G identifies Debit, then it is possible to purchase the registered item on-line and then download it using the URL of G.ObjectLocation. The procedure for handling the debit or credit card transaction is not part of this specification.
15. The ExpiryDate 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 initiates an expiration action.
 16. The Description is provided by the SO.
 17. The SubmittingOrg identifies the organization submitting the registered object. It points to a unique instance of the ORGANIZATION entity. On presentation of this information, the RA substitutes the CommonName of the organization. The SO must be known to the RA before it can make submissions to the Registry/Repository, and they each know of a unique URN for the other. The process for becoming known is not part of this specification.
 18. The ResponsibleOrg identifies the organization responsible for the formal specification of the registered object. It points to a unique instance of the ORGANIZATION entity. The RO may be a formal accredited standards development organization or it may be the SO. On presentation of this information, the RA substitutes the CommonName of the organization.
 19. The PublicComment may be suggested by the SO, but it is supplied by the RA. In most cases the comment will explain some administrative process that cannot be clearly determined from the standardized information. For example, this comment may explain how long the metadata for a replaced or withdrawn object remains available, or how long an expired object remains available before it is deleted.

X.6.2 ASSOCIATION

Structure

Attribute Name	Attribute Type	Presence
GivenItemRole	CodeText	Mandatory
AssocItemURN	URNref	Mandatory
AssocItemId	REGISTRY_ITEM	
Comment	CommentText	

Semantic Rules

1. The Association entity represents a many-to-many relationship from the RegistryItem entity to itself. Each association instance relates a given registry item instance to an associated registry item instance. The given item plays a specified role with the associated item.
2. Each instance of an Association depends upon the given item. This dependency is represented by an implicit value, GivenItemId, of type REGISTRY_ITEM.
3. The GivenItemRole is provided by the SO with its value taken from the AssociationType enumeration entity. For OASIS registrations, that entity includes the values {Uses, Supersedes, Contains, Related}.
4. The AssocItemURN is provided by the SO. It must identify the AssignedURN of some registry item in some OASIS conformant Registry. If the Registry is not the local one, then the RA has no responsibility to ensure that it identifies a relevant item.
5. The AssocItemId is present if and only if the registry item identified by the AssocItemURN is a registry item in the same registry as the association instance. In that situation, AssocItemURN and AssocItemId identify the same registry item.
6. Let G be a registry item whose DefnSource is OASIS and let X be an association instance linked to G with G as the GivenItem.

Case:

- a) If X.GivenItemRole identifies Uses, then the SO declares that G *uses* the registry item identified by X.AssocItemURN. The RA is under no obligation to check whether this declaration is or remains true.

- b) If X.GivenItemRole identifies Supersedes, then the SO declares that G *supersedes* the registry item identified by X.AssocItemURN. If A is that item, then A must be in the same registry as G and A.RegStatus must be Superseded.
 - c) If X.GivenItemRole identifies Contains, then the SO declares that G *contains* the registry item identified by X.AssocItemURN. If A is that registry item, then A must be in the same registry as G and A.PrimaryClass must be Package.
 - d) If X.GivenItemRole is Related, then the SO declares that G is *related* data for the registry item identified by X.AssocItemURN. The RA is under no obligation to check whether this declaration is or remains true.
7. The Comment is provided by the SO. It may explain in more detail the relationship between the given and associated registry items.

X.6.3 RELATED_DATA

Structure

AttributeName	AttributeType	Presence
DataName	ShortName	Mandatory
DataLocation	URLref	Mandatory
RelatedType	CodeText	Mandatory
MimeType	MIMEtype	
Comment	CommentText	

Semantic Rules

1. The RelatedData entity represents the set of non-registered objects that are related to registered objects. Each instance is a pairwise relationship between a single registered item and a single related data item. A registered item may map to many related data items.
2. Each instance of RelatedData depends upon a RegistryItem instance. This dependency is represented by an implicit value, RAitemId, of type REGISTRY_ITEM.
3. The DataName attribute is provided by the SO. It is intended that this be the link name for the DataLocation if related data items are presented visually to a user.
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 link is a valid one.
5. The RelatedType is provided by the SO and takes its value from the RelatedDataType enumeration entity. It may include values not defined by OASIS.
6. The MimeType is provided by the SO. It identifies the MIME type of the related data item. The RA is under no obligation to ensure that the declared MimeType type is consistent with the actual file type of the file referenced by DataLocation.
7. The Comment is provided by the SO. It may further explain the relationship between the related data instance and the registry item it is linked to.

X.6.4 CLASSIFICATION

Structure

AttributeName	AttributeType	Presence
SchemeURN	URNref	Mandatory
LevelCode	CodeText	Mandatory
ItemValue	CodeText	Mandatory
Comment	CommentText	

Semantic Rules

1. The Classification entity identifies all classifications of a registry item.
2. Each Classification instance identifies a node in a classification hierarchy and maps the registry item to that node.
3. Each Classification instance has an implicit value, RAitemId, of type REGISTRY_ITEM that maps that instance to its associated registry item. The classification instance is dependent on that registry item.
4. The SchemeURN is provided by the SO. It must identify the AssignedURN of some registry item in the same Registry as the classification instance and the object referenced by that registry item must be a classification scheme.
5. The LevelCode is provided by the SO, or the RA fills it in by default, depending on the type of classification scheme identified by the SchemeURN. If that classification scheme does not require multiple levels, then the default value for LevelCode is “Leaf”.
6. The ItemValue attribute is provided by the SO. It must be a legal ItemValue as defined by the registered classification scheme.
7. The optional Comment is supplied by the SO. It might give a purpose for using the identified classification scheme, or explain why multiple values are given for the same level, or qualify the classification in some other way.

X.6.5 ORGANIZATION

Structure

AttributeName	AttributeType	Presence
OrgURN	URNref	Mandatory
OrgFullName	LongName	Mandatory
CommonName	ShortName	
OrgURL	URLref	
ParentOrg	ORGANIZATION	
AddrLine1	AddrLineText	
AddrLine2	AddrLineText	
AddrLine3	AddrLineText	
City	ShortName	
StateProv	ShortName	
Country	ShortName	
PostalCode	CodeText	
Email	EmailText	
Telephone	TelephoneText	
Fax	TelephoneText	
Comment	CommentText	

Semantic Rules

1. The Organization entity 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 the organization is linked to its parent.
2. Each Organization instance is represented by an identifier of type ORGANIZATION. This identity is used to link an organization with other entities in the Registry.
3. Each organization instance has a relationship with one or more Contact instances.
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.
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.

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 OrgURL is provided by the SO. It is the locator for the home page of that company or organization on the world wide web.
8. 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.
9. 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.
10. The City is provided by the SO. It identifies the city associated with the AddressLines.
11. The StateProv is provided by the SO. It identifies the state, province, territory, or other political unit between the city and country.
12. The Country is provided by the SO. It identifies the home country of the company or organization.
13. 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.
14. The Email is provided by the SO. It identifies the best email address to use when trying to reach the company or organization headquarters.
15. 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.
16. 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.
17. The Comment is provided by the SO or RA. It conveys any important information not captured by the other attributes.

X.6.6 CONTACT

Structure

AttributeName	AttributeType	Presence
ContactName	ShortName	Mandatory
OrgId	ORGANIZATION	Mandatory
OrgRole	CodeText	Mandatory
SubmitId	SUBMISSION	
RequestNbr	SmallInt	
Availability	CodeText	Mandatory
ContactRole	CodeText	Mandatory
Email	EmailText	
Telephone	TelephoneText	
Fax	TelephoneText	
Comment	CommentText	

Semantic Rules

- 1) The Contact entity 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 represented by an identifier of type CONTACT. This identity is used to link a contact with other entities in the Registry.
- 3) Each contact instance is linked to a single organization.
- 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 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.
- 6) The OrgRole is provided by the SO, with its value taken from the OrganizationRole enumeration entity. For OASIS registrations, that entity identifies one of the values {SO, RA, RO}.
- 7) The SubmitId is provided by the RA. If this contact was provided as part of a submission from the SO to the RA, then it references that submission.

- 8) The RequestNbr is provided by the RA. If this contact was provided as part of a request in a submission from the SO to the RA, then it references that request.
- 9) The Availability code is provided by the SO with its value taken from the ContactAvailability enumeration entity. For OASIS registrations, that entity identifies one of the values {Private, Protected, Public}. The default value is Public.

If the Availability value is:
Case:

- a) Private, then the RA will not reveal the <simple-contact> to any other organization.
 - b) Protected, then the RA may reveal the <simple-contact> to other legitimate registration authorities. [Note: legitimate left purposely undefined for now.]
 - c) Public, then the RA may reveal the <simple-contact> to any user of the registry maintained by the RA.
- 10) The ContactRole is provided by the SO with its value taken from the ContactRole enumeration entity. For OASIS registrations, that entity identifies one of the values {Administrative, Technical, All Issues}. The default value is All Issues.

If the ContactRole value is:
Case:

- a) Administrative, then the contact is prepared to address administrative questions related to registered items.
 - b) Technical, then the contact is prepared to address technical questions related to registered items.
 - c) All Issues, then the contact is prepared to address all issues related to the submitted item.
- 11) 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.
 - 12) 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.
 - 13) 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.
 - 14) 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.

X.6.7 ALTERNATE_NAME

Structure

AttributeName	AttributeType	Presence
AltName	LongName	Mandatory
Context	CodeText	Mandatory
Comment	CommentText	
Language	CodeText	
Encoding	CodeText	

Semantic Rules

1. The AlternateName entity 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, RAitemId, of type REGISTRY_ITEM that maps that instance to its associated registry item. The classification instance is dependent on that registry item.
3. The AltName attributes is provided by the SO.
4. The Context is provided by the SO. It is a code value that identifies an item from the NameContext enumeration entity. The context may identify the name as a URN from some other registry, or identify the name as appropriate for use with a specific programming language, or provide a name for the registered object in some other human language.
5. The Comment is provided by the SO. It may give further details about how the name should be used.
6. The Language attribute is provided by the SO. It is of datatype LanguageId, defined in Section X.4. The default is "en-US".
7. The Encoding attribute is provided by the SO. It is of datatype Encoding, defined in Section X.4. The default is "Latin-1".
8. The intent is that all attribute values for an AlternateName instance be encoded in the encoding scheme identified by the Encoding attribute.

X.6.8 ALTERNATE_DESCRIPTION

Structure

AttributeName	AttributeType	Presence
Language	CodeText	Mandatory
Encoding	CodeText	Mandatory
Description	DescriptionText	Mandatory

Semantic Rules

1. The AlternateDescription entity 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 item, with the human language and character encoding identified. The intent is that each description is a direct translation of the Description attribute of the associated registry item into the language and encoding identified by the Language and Encoding attributes.
2. Each AlternateDescription instance has an implicit value, RAitemId, of type REGISTRY_ITEM that maps that instance to its associated registry item. The alternate description instance is dependent on that registry item.
3. The Language attribute is provided by the SO. It is of datatype LanguageId, defined in Section X.4. The default is "en-US".
4. The Encoding attribute is provided by the SO. It is of datatype Encoding, defined in Section X.4. The default is "Latin-1".
5. The Description is provided by the SO. The intent is that each description is a direct translation of the Description attribute of the associated registry item into the language and encoding identified by the Language and Encoding attributes.

X.6.9 SUBMISSION

Structure

AttributeName	AttributeType	Presence
SubmitTime	Datetime	Mandatory
SubmitOrg	ORGANIZATION	Mandatory
Comment	CommentText	

Semantic Rules

1. The Submission entity 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 entity represents a single submission. If the SO is not known to the RA, then the Submission must consist of a single request, CertifySO, that asks the RA to certify the submitter as an SO for that Registry.
3. Each Submission instance is represented by an identifier of type SUBMISSION. 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 entity.
5. A submission must be related to one or more requests. Each request is represented as an instance of the Request entity.
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 SubmitOrg 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.
8. 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.

X.6.10 REQUEST

Structure

AttributeName	AttributeType	Presence
RequestNbr	SmallInt	Mandatory
RequestCode	CodeText	Mandatory
ContentXML	XMLtext	Mandatory
Comment	CommentText	

Semantic Rules

1. The Request entity represents a set of requests made to a Registration Authority (RA) from Submitting Organizations (SO).
2. 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.
3. 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. In most cases, if there are N requests in a submission package, then the RequestNbr's will vary from 1 to N inclusive. This default may be overridden by the SO, but there is no semantic significance to request order.
4. The RequestCode attribute is provided by the RA after receiving an XML Request DTD from an SO. Its value is the code defined by the RegistryRequest enumeration entity. It may include values not defined by OASIS.
5. 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.
6. The Comment attribute is provided by the SO. It may provide further explanation about the purpose of the request.

X.6.11 IMPACT

Structure

AttributeName	AttributeType	Presence
SubmitId	SUBMISSION	Mandatory
RequestNbr	SmallInt	Mandatory
RAitemId	REGISTRY_ITEM	Mandatory
ImpactCode	CodeText	Mandatory
Comment	CommentText	

Semantic Rules

1. The Impact entity represents a many-to-many relationship between the Request entity and the RegistryItem entity. Each Impact instance relates a single Request instance with a single RegistryItem instance
2. A request, e.g. CertifySO, may not have an impact on any registry item, 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. The Impact entity, together with its parent Submission and Request entities, will maintain a historical record of all SO-initiated modifications of a registry item from first creation to final deletion.
4. Every registry item 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.
5. The SubmitId and RequestNbr attributes are provided by the RA. They identify the submission package and the request that had an impact on the related registry item.
6. The RAitemId attribute is provided by the RA. It identifies the impacted registry item.
7. The ImpactCode attribute is provided by the RA. This code identifies one of the impacts specified by the RegistryItemImpact entity domain.
8. The Comment is provided by the RA. It may explain some unusual consideration.

X.6.12 CLASSIFICATION_SCHEME

Structure

AttributeName	AttributeType	Presence
SchemeURN	URNref	Mandatory
Comment	CommentText	

Semantic Rules

1. The ClassificationScheme entity identifies the classification schemes known to a Registry. Each instance of this entity identifies one classification scheme.
2. Each instance of a ClassificationScheme is in a one-to-one correspondence with an instance of the RegistryItem entity. This dependency is represented by an implicit value, RaitemId, of type REGISTRY_ITEM.
3. The SchemeURN matches the AssignedURN of the associated registry item.
4. The Comment is provided by the SO. It may expand upon the Description provided by the Description attribute of the associated registry item.

X.6.13 CLASSIFICATION_LEVEL

Structure

AttributeName	AttributeType	Presence
LevelCode	CodeText	Mandatory
LevelNbr	SmallInt	Mandatory
LevelName	LongName	
Comment	CommentText	

Semantic Rules

1. The ClassificationLevel entity identifies the levels of a classification scheme. Each instance represents a single level of the scheme.
2. Each instance of a ClassificationLevel depends upon a ClassificationScheme instance. This dependency is represented by the implicit value, SchemeId, of type REGISTRY_ITEM. If the classification scheme is a simple 1-level scheme, then the scheme has a single default level instance with LevelNbr set equal to 1, LevelName = "Level 1", and LevelCode = "1". The SO can override these defaults.
3. The LevelCode is provided by the SO. It identifies the value that will be referenced as the level code in a Classification.
4. The LevelNbr is provided by default by the RA if not explicitly provided by the SO. It is a positive integer that identifies the level in a classification hierarchy. In most cases, if there are N levels, then the LevelNbr's vary from 1 to N inclusive. The SO may override these defaults by explicitly numbering the levels provided in an XML ClassificationScheme DTD.
5. 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 never used in a Classification; instead, the LevelCode is always used.
6. The Comment is provided by the SO. It may further describe the level.

X.6.14 CLASSIFICATION_ITEM

Structure

AttributeName	AttributeType	Presence
ItemId	SmallInt	Mandatory
ItemValue	CodeText	Mandatory
ItemName	LongName	
ParentId	SmallInt	Mandatory
Comment	CommentText	

Semantic Rules

1. The ClassificationItem entity represents the classification hierarchy of a classification scheme. Each instance represents a node in that hierarchy. For convenience, we define a node to represent the entire sub-tree below it. The term item is used to reference an individual node without the sub-tree below it.
2. Each instance of a ClassificationItem depends upon a ClassificationScheme instance and a ClassificationLevel instance. This dependency can be represented by a pair of typed values: SchemeId of type CLASSIFICATION_SCHEME, and LevelNbr of type SmallInt. If the classification scheme is a simple 1-level scheme, then all nodes map to the single default level having LevelNbr=1.
3. The ItemId attribute is a positive integer that identifies each node of the classification hierarchy. The Item Id's are provided locally by the RA, usually numbered consecutively based on an instance of a node in the XML ClassificationScheme DTD. The root of each classification hierarchy is the classification scheme itself. The root is a virtual node, not represented in this entity, that is assumed to have ItemId=0.
4. The ItemValue is provided by the SO. It identifies the value that will be referenced as the item value in a Classification.
5. 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 never used in a Classification; instead, the ItemValue is always used.
6. The ParentId is supplied by the RA. It references the ItemId of the parent node in the classification hierarchy. If the node is at Level 1 in the classification hierarchy, then the ParentId is set to 0, to indicate that its parent node is the virtual root of the hierarchy.
7. The Comment is provided by the SO. It may further describe the item.

X.6.15 REPOSITORY

Structure

AttributeName	AttributeType	Presence
INCOMPLETE		

Semantic Rules

[TO BE COMPLETED]