OASIS ebXML Registry Technical Committee

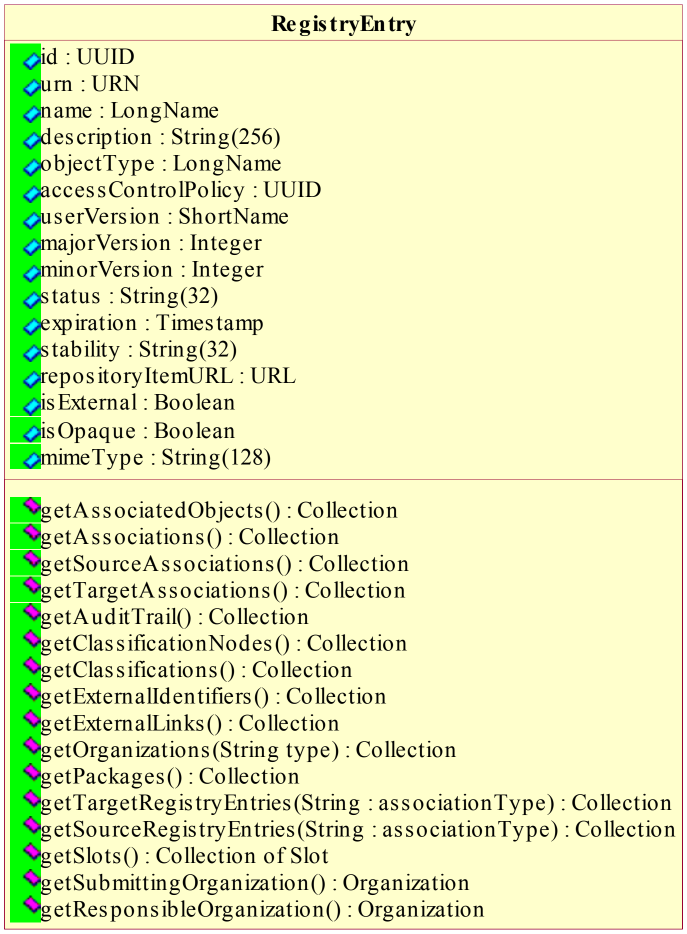
Subject: RegistryEntry: Issues re Attributes and Methods
Version: 1
Author: Len Gallagher
Date: 22 October
Type: ebRIM Issue Paper with Proposals

Introduction

This is an issues paper that focuses on a single Registry class, namely RegistryEntry. I believe that the RegistryEntry class is the most important class in our model because it represents objects that a submitting organization (SO) wishes to have registered. In order to register an object, the SO must create a RegistryEntry instance to describe it. The RegistryEntry instance will hold the name and description of the registered object, its registration characteristics, and pointers to other registry instances that give additional metadata related to the registered object.

What I'd like to do at the upcoming face-to-face meeting is have a discussion restricted to just the attributes and methods of this class. In the figure below I've included all existing attributes and methods, including those inherited from RegistryObject, as well as those proposed by other proposals to be considered at this meeting. If we can come to agreement on the intended purpose of each of these, then we should be able to make rapid progress towards a solution agreeable to all. After discussion of the issues for each attribute or method, I often make one or more proposals to clarify its intended interpretation.

RegistryEntry class Diagram


RegistryEntry
id : UUID
urn : URN
name : LongName
description : String(256)
objectType : LongName
accessControlPolicy : UUID
userVersion : ShortName
majorVersion : Integer
minorVersion : Integer
status : String(32)
expiration : Timestamp
stability : String(32)
repositoryItemURL : URL
isExternal : Boolean
isOpaque : Boolean
mimeType : String(128)
getAssociatedObjects() : Collection
getAssociations() : Collection
getSourceAssociations() : Collection
getTargetAssociations() : Collection
getAuditTrail() : Collection
getClassificationNodes() : Collection
getClassifications() : Collection
getExternalIdentifiers() : Collection
getExternalLinks() : Collection
getOrganizations(String type) : Collection
getPackages() : Collection
getTargetRegistryEntries(String : associationType) : Collection
getSourceRegistryEntries(String : associationType) : Collection
getSlots() : Collection of Slot
getSubmittingOrganization() : Organization
getResponsibleOrganization() : Organization

Attribute Discussion

id:UUID
This attribute is the persistent object identifier for each persistent RegistryEntry instance. It is a fixed-length identifier (i.e. 128 bits) that can be used for internal associations as well as for external references to specific RegistryEntry instances. Its fixed-length nature makes it easy to optimize for those purposes. Its global uniqueness makes it easy to share metadata references among cooperating Registries while retaining their individual uniqueness. However, I think it is problematic to use this "id" as a way to reference the "repository item" described by a RegistryEntry instance. If different Registries have different RegistryEntry instances that point to the same "repository item", then they will want to have a common, shared identifier for it. I think we need a new attribute, "urn" or something analogous, to serve as a human-friendly, globally-unique name that can be used as a surrogate identifier for that "repository item". This issue is discussed further with "urn" just below.

urn:URN
The issue of the desirability of having a new "urn" attribute for RegistryEntry (and Organization) is discussed in the Query Team paper http://lists.oasis-open.org/archives/regrep-query/200109/msg00062.html. This paper contains the outline of a proposal to add a new "urn" attribute to the RegistryEntry and Organization classes. Since this proposal is beyond the scope of just query, I've distributed it to the whole Registry TC. Acceptance of this proposal would allow different Registry implementations to all have different RegistryEntry instances, with different "id" attributes, that still all describe and reference the same "repository item" using the same "urn". For example, Registries in many different industries could each reference NAICS or UNSPSC using the same URN, independently of whether the taxonomies of these schemes are "internal" or "external".
Proposal: Add "urn" as a new RegistryEntry attribute as described by the above referenced paper.

name:LongName (128 characters)
This is the descriptive name supplied by the SO. The RA has no control over this name and in the absence of profanity will normally accept whatever the SO submits. Thus we can make no assumptions about this name. It may or may not be unique within some context. It may or may not include version information. It might even include some terms that are copyright protected.

description:String(256)
This is the description of the "repository item" referenced by the RegistryEntry instance. The RA has no control over this description and in the absence of profanity will normally accept whatever the SO submits.

objectType:LongName(128 characters)
This is an inherited attribute from RegistryObject. In all classes except RegistryEntry and its sub-classes it identifies the type of the class, but in RegistryEntry it is an enumeration that identifies the "type" of the "repository item" described by the RegistryEntry instance. The valid values are presented in Section 6.3.6 of ebRIM. I'd like to see a couple of minor restrictions on the data type of this attribute. Suppose we agree that the valid objectType's are pre-defined by some responsible organization (RO). In fact, the valid objectType's will likely be specified by a taxonomy with "unique" codes (i.e. the "code" attribute of a ClassificationNode instance uniquely identifies that node). This attribute can be thought of as a pointer to a specific node of that classification scheme. As such, I think we can restrict the values to something more stable than an arbitrary String of 128 characters. How about if we restrict such enumerations to shorter strings (e.g. 32 characters) of characters contained in the International Reference Version (IRV) of ISO 646. These characters are recognized by nearly every international character set and will survive most character set transformations. For example, from the ebRIM list, "CPA" and "CPP" could be thought of as a sub-classification of "XMLDocument" since they will validate to a specific schema definition. I suggest we define a new data type, e.g. CodeText, which will be a variable length string, of length greater than 0, with no space characters, and consisting only of IRV characters, possibly with some XML special characters removed. We can then use this new CodeText data type for all attributes that carry a value defined by a responsible organization.
Proposal_1: Define a CodeText data type and use it as the type of this attribute, and of any other attributes in RIM that assume a pre-determined code value defined by some responsible organization.
Proposal_2: Decide how the objectType attribute is supposed to be used and interpreted in classes other than RegistryEntry. If the value is minimal, consider moving the attribute from RegistryObject to RegistryEntry so that its interpretation can always be "a declaration of the type of the repository item described by this RegistryEntry instance".

accessControlPolicy:UUID

This attribute is inherited from RegistryObject. It is a pointer to an AccessControlPolicy instance, discussed in Section 11 of ebRIM. Since Section 11 is rather sparse in its requirements, I'm concerned that a Registry may have a very valid "default" access control policy that applies to every RegistryObject instance. If so, why is the Registry required to carry around this attribute on each instance. Wouldn't it be just as effective to have this UUID be found via a method? Then the value would not have to be stored persistently in each instance.
Proposal: Remove "accesscontrolPolicy" as an attribute of RegistryObject; instead, define a new method, getAccessControlPolicy() returning either: 1) a UUID REF , or 2) an actual AccessControlPolicy instance.

userVersion:ShortName (64 characters)
This is the version supplied by the SO. The RA has no control over this version and in the absence of profanity will normally accept whatever the SO submits. Thus we can make no assumptions about its completeness or relationship to majorVersion and minorVersion, or its representation in any "id" or "urn" identifier. It may or may not be unique within some context.

majorVersion:Integer
This is a version number assigned by the RA when the RegistryEntry instance is first created and it is maintained by the RA when an object is updated. Presumably, the version is derivable from the AuditTrail returned by the getAuditTrail() method, but there are no rules in ebRIM to specify when and/or how this happens. Since this value is derivable from other information that may be left unspecified until we add a robust version control facility, I'd prefer to see it as a method instead of an attribute. If it stays as an attribute, I think we need rules to reconcile its relationship with the AuditTrail!
Proposal: Remove majorVersion as an attribute of RegistryEntry; instead, define a new method for the RegistryEntry class, getMajorVersion(), that returns an integer. Its value is derivable from the AuditTrail using an implementation-defined version control algorithm.

minorVersion:Integer
This is a version number assigned by the RA when the Registryentry is first created and it is maintained by the RA when an object is updated. Presumably, the version is derivable from the AuditTrail returned by the getAuditTrail() method, but there are no rules in ebRIM to specify when and/or how this happens. Since this value is derivable from other information that may be left unspecified until we add a robust version control facility, I'd prefer to see it as a method instead of an attribute. If it stays as an attribute, I think we need rules to reconcile its relationship with the AuditTrail!
Proposal: Remove minorVersion as an attribute of RegistryEntry; instead, define a new method for the RegistryEntry class, getMinorVersion(), that returns an integer. Its value is derivable from the AuditTrail using an implementation-defined version control algorithm.

status:LongName (128 characters)
The values of this attribute come from an enumeration that is defined in Section 6.4.6 of ebRIM. Just like the "objectType" attribute defined above, I think this enumeration will likely represent a "unique code" for a node in a classification taxonomy pre-defined by some responsible organization (RO). As such it is no problem to restrict the valid values to be more internationally interpretable.
Proposal: Let the data type of "status" be the CodeText type defined as above for "objectType".

expiration:Timestamp
This attribute specifies the expiration date of the registration status defined by the "status" attribute. Its value can be suggested by the SO, but final determination is at the prerogative of the RA. For example, an RA may have a policy that no registration status lasts longer than a fixed amount of time. I think it would be wise if ebXML Registry could agree on a standard default representation of timestamp as a character string literal, so that whenever a timestamp is passed to or from the registry, and a specific format is not specified, the default format will apply.
Proposal: Let the default string representation of timestamp be YYYYMMDD::HH:MM:SS.FFF, i.e. a representation as a CodeText string, with the default assumption, unless specified otherwise, that it represents Universal Coordinated Time (UT) as defined by ISO.

stability:LongName (128 characters)
The values of this attribute come from an enumeration that is defined in Section 6.4.5 of ebRIM. Just like the "status" attribute, its enumeration will likely represent a "unique code" for a node in a classification taxonomy pre-defined by some responsible organization (RO). As such it is no problem to restrict the valid values to be more internationally interpretable.
Proposal: Let the data type of "status" be the CodeText type defined as above for "objectType".

repositoryItemURL:URL
This is a new attribute for the RegistryEntry class proposed by the October 11 paper, “Support for External Repository Items”, cf http://lists.oasis-open.org/archives/regrep/200110/pdf00001.pdf. Whether the name is “contentURI”, “discoveryURL”, “locationURL”, or “repositoryItemURL” is not so important. What is important is that users of the Registry have a standard, web-resolvable way to “locate” the repository item described by a RegistryEntry instance.
Proposal: Add a new attribute, repositoryItemURL, to the RegistryEntry class. Define its semantics to be that specified for “discoveryURL” in the above referenced paper.

isExternal:Boolean
This is a new attribute, that should be considered if the “Support for External Repository Items” proosal discussed above is accepted by the F2F. It would explicitly state whether the “repository item” described by the RegistryEntry instance is “internal” to this Registry or “external” to it. If the repository item is external, then the RA has no responsibility to ensure that it validates to the declared “objectType”. For some “internal” repository items, the Registry may (or may not) assume some responsibility to validate the “repository item” to the declared “objectType”, especially if we’re dealing with “repository items” that have a “ValidatesTo” association with some registered XML DTD or schema.

isOpaque:Boolean
This attribute is currently defined for the ExtrinsicObject class, a sub-class of RegistryEntry. If the “Support for External Repository Items” proposal discussed above passes, then it may make sense for this attribute to move up to RegistryEntry so that it could be used to give more information about the representation of an “external repository item”.

mimeType:LongName (128 characters)
This attribute is currently defined for the ExtrinsicObject class, a sub-class of RegistryEntry. If the “support for External Repository Items” proposal discussed above passes, then it may make sense for this attribute to move up to RegistryEntry so that it could be used to give more information about the representation of an “external repository item”. For example, a RegistryEntry instance that describes an external classification taxonomy may have an objectType of “ClassificationScheme” because it will have a ClassificationScheme instance in this Registry that describes an external classification taxonomy. Then the mimeType will give additional information about how that taxonomy is represented, e.g. gif, jpeg, text, xml, etc.

Method Discussion

NOTE: In ebRIM v1.1, line 552+, a Collection type is defined to be a collection of multiple RegistryObject instances. There are different ways that one might interpret Collection: 1) a collection of REF’s or UUID’s each pointing to a RegistryObject instance, 2) a collection of strings each of which can be parsed to point to a RegistryObject instance, or 3) a collection of actual RegistryObject instances. None of these interpretations can be applied exclusively for all methods below. I think we need to take a closer look at how these methods are defined, their interpretation for each sub-class of RegistryObject, and what is the best interpretation of the data type of the returned result.

getAssociatedObjects():Collection
This is an inherited method from RegistryObject that can return many different subtypes of RegistryObject. Not all instances of these subtypes are represented by UUID’s (e.g. Slot, Association, Classification, cf ebRIM section 6.3.4). Wouldn’t it be better to allow (or require) an input variable to this method, e.g. objectType, that would allow the client to specify the specific type of associated object to be returned? Then getAssociatedObjects(RegistryEntry) would return only a homogeneous collection of UUID references to RegistryEntry instances, whereas getAssociatedObjects(Association) would return a homogeneous collection of Association instances, either as the compound triple-UUID strings defined in ebRIM Section 9.1.5, or as a collection of Association instances. A second issue is how this method differs from the Union of all of the methods defined below. In particular, do getExternalIdentifiers() and getPackages() both return subsets of getAssociations()?
Proposal: I think getAssociatedObjects() should offer “objectType” as an input variable, or it should be deleted in favor of more explicit methods, e.g. getRegistryEntries(), analogous to others defined below. I favor deletion and replacement by more specific methods. In particular, see the getTargetRegistryEntries() method proposed below.

getAssociations():Collection
This is an inherited method from RegistryObject. It's not clear if it returns a collection of UUID's, a collection of compound triple-UUID strings as specified in ebRIM Section 9.1.5, or a collection of Association instances. I think I favor the last alternative because it would avoid requiring the Client software having to parse a compound string to determine the source and target objects and would give access to all attributes of an Association instance.
Proposal_1: Clarify that getAssociations() returns a Collection of Association instances, where each instance carries at least the three attributes: sourceObject, targetObject, and associationType.
Proposal_2: Section 6.3.7 of ebRIM v1.1 defined getAssociations() to return all associations where "this object is the source of the Association". It ignores the case where this object is the target of the Association. Consider re-naming this method definition to be getSourceAssociations() to emphasize that this object is the "sourceObject" of the returned associations.

getTargetAssociations():Collection
The previous method gets all associations where this object is the sourceObject of the association. I think it will be valuable to have a method that goes the other way around too.
Proposal_1: Define a new method on RegistryEntry, getTargetAssociations(), that returns a Collection of Association instances where the given object is referenced by the targetObject attribute of the Association instance.
Proposal_2: Consider replacing the getSourceAssociations() and getTargetAssociations() by more explicit getTargetRegistryEntries() and getSourceRegistryEntries() methods discussed below. The advantage of the more explicit methods is that the Collection returned as a result could always be interpreted as a Collection of UUID's.

getAuditTrail():Collection of UUID
This is an inherited method from RegistryObject. Since AuditTrail can be interpreted as a set of AuditableEvent instances, each with a UUID identifier, we're OK interpreting Collection as a set of UUID's.
Proposal: Clarify that getAuditTrail() returns a Collection of UUID's, each pointing to an AuditableEvent instance.

getClassificationNodes():Collection of UUID
This is an inherited method from RegistryObject that returns the collection of ClassificationNode instances that classify the subject object. Since each ClassificationNode instance has a UUID identifier, we're OK interpreting Collection as a set of UUID's.
Proposal: Clarify that getClassificationNodes() returns a Collection of UUID's, each pointing to a ClassificationNode instance. Keep in mind that this method will only identify "internal classifications"; in order to include "external classifications" one will have to use the getClassifications() method.

getClassifications():Collection
This is an inherited method from RegistryObject. It's not clear if it returns a collection of UUID's, a collection of compound UUID strings as specified in ebRIM Section 10.3.4, or a collection of Classification instances. I think I favor the last alternative because it would avoid requiring the Client software having to parse a compound string to determine the parent and target objects.
Proposal: Clarify that getClassifications() returns a Collection of Classification instances, where each instance includes at least the attributes that uniquely identify the Classification instances, i.e. a ClassificationScheme UUID and a nodeRepresentation.

getExternalIdentifiers():Collection
This is an inherited method from RegistryObject. It's not clear if it returns a collection of UUID's, a collection of compound UUID strings as specified in ebRIM Section 6.8.4, or a collection of ExternalIdentifier instances. I think I favor the last alternative because it would avoid requiring the Client software having to parse a compound string to determine the parent and target objects. It should return at least the value attribute of each ExternalIdentifier instance.
Proposal_1: Clarify that getExternalIdentifiers() returns a Collection of ExternalIdentifier instances, where each instance carries at least the "value" attribute.
Proposal_2: Decide if it makes sense for classes other than RegistryEntry to have this method. If not, move this method from RegistryObject to RegistryEntry.

getExternalLinks():Collection
This is an inherited method from RegistryObject. It's not clear if it returns a collection of UUID's, a collection of externalURI attributes, or a collection of ExternalLink instances. I think I favor the last alternative because it would allow additional attributes on ExternalLink to be returned at the same time, e.g. an externalLinkType attribute.

Proposal_1: Clarify that getExternalLinks() returns a Collection of ExternalLink instances, where each instance carries all attributes that determine a unique instance of ExternalLink.
Proposal_2: Decide if it makes sense for classes other than RegistryEntry to have this method. If not, move this method from RegistryObject to RegistryEntry.

getOrganizations(String type) :Collection of UUID
This is an inherited method from RegistryObject. The "type" input variable is supposed to allow one to retrieve only Organizations that are the targetObject of Association instances having associationType=type. This is intended to support the "submitting organization" and "responsible organization" requirements on a RegistryEntry instance, but ebRIM v1.1 fails to make this clear. It does NOT define any AssociationType values for this purpose (cf ebRIM Section 9.1.2)
Proposal_1: Support explicit methods for getSubmittingOrganization() and getResponsibleOrganization() in all classes where they make sense. See below for proposals to do just that.
Proposal_2: With explicit methods for the above two important cases, decide if this general purpose method still has value in the model. If so, define the appropriate associationType values that make it useful.

getPackages():Collection of UUID
This is an inherited method from RegistryObject. This method returns the collection of Package instances that have this object as a member. Since Package is a sub-class of RegistryEntry, each instance will have a UUID, so it's OK to think of the result of this method as a Collection of UUID's, each of which identifies a Package instance.
Note: One could think of this method as a shorthand for first finding all of the Association instances having associationType="HasMember" and where this object is the targetObject, then extracting the sourceObject UUID from each of those instances.

getTargetRegistryEntries(String:associationType):Collection of UUID
This is a proposed new method on RegistryEntry that returns a collection of object identifiers that reference RegistryEntry instances that are linked to the given RegistryEntry instance via an Association instance having the given object as the sourceObject and having an associationType that matches the input parameter.
Proposal_1: Add this new method to the RegistryEntry class in ebRIM.
Proposal_2: Consider adding an analogous getSourceRegistryEntries(String:AssociationType) method. It would return a Collection of UUID's that are the sourceObjects of the identified Association instances.
Proposal_3: Consider replacement of the getAssociations(), getSourceAssociations(), and getTargetAssociations() methods defined above by more specific methods that identify the explicit class of all returned UUID's.

getSlots():Collection
This is an inherited method from RegistryObject. Since Slot has no "id" attribute, we have no choice but to say that this method returns a Collection of Slot instances rather than a Collection of UUID's or a collection of parseable strings. Each instance will contain all of the Slot attributes, i.e. name, slotType, and values.
Note: Consider whether or not it is important for each Slot instance to carry a "values" attribute that is itself a Collection of ShortName values. If it is sufficient that each Slot instance carry only a single value, then this class can be handled much more easily.
Proposal: Change the "values" attribute of Slot to "value" and make its data type ShortName.

getSubmittingOrganization():Organization
This is a new method proposed by a recent paper in the Query Team archive, http://lists.oasis-open.org/archives/regrep-query/200110/msg00021.html. I intend to forward this proposal to the Registry TC so that it will be on our F2F agenda. It adds a getSubmittingOrganization() method to each class where it makes sense, including RegistryEntry.
Proposal: Accept the getSubmittingOrganization() methods for RegistryEntry and other ebRIM classes proposed in http://lists.oasis-open.org/archives/regrep-query/200110/pdf00005.pdf.

getResponsibleOrganization():Organization
This is a new method proposed by a recent paper in the Query Team archive, http://lists.oasis-open.org/archives/regrep-query/200110/msg00021.html. I intend to forward this proposal to the Registry TC so that it will be on our F2F agenda. It adds a getResponsibleOrganization() method to the RegistryEntry class. The intent is that when an SO registers an object, it can optionally name a responsible organization (SO), whose responsibilities are defined in ISO/IEC 11179. This method makes sense only for the RegistryEntry class.
Proposal: Accept the getResponsibleOrganization() method for the RegistryEntry class proposed in http://lists.oasis-open.org/archives/regrep-query/200110/pdf00005.pdf.