Date: June 2013

CTS2 Entity Description Services

Version 1.1

OMG Document Number: formal/2013-05-10
Standard document URL: http://www.omg.org/spec/cts2/1.1/

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT

LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , IMM™ , MOF™ , OMG Interface Definition Language (OMG IDL)™ , and OMG Systems Modeling Language (OMG SysML)™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

Table of Contents

1 Introduction

This document describes the information model and corresponding services for the `EntityDescription` and `Association` resources. Entity descriptions describe entities - a collective term for a variety of terminological resources that are can be referred to as “classes,” “concepts,” “terms,” “categories,” “roles,” “predicates,” “properties,” and “individuals” in different contexts and models. `EntityDescription` is a structured model of the “non-semantic” or “lexical” aspects of these descriptions while Association models the formal or “semantic” relationships between them.

`EntityDescription` represents the classes *CodeSystemEntity*, *CodeSystemEntityVersion*, *CodeSystemNode*, *CodeSystemConcept*, *CodeSystemConceptCode*, *Designation*, *DesignationType*, *AssociationType*, and *DefinedEntityProperty* in the CTS2 SFM, while `Association` addresses *CodeSystemEntityVersionAssociation*.

From the OWL perspective, `EntityDescription` represents the collection of “annotation properties” - properties that are not inherited and do not participate in the formal reasoning process, while Association represents everything else. From the perspective of SNOMED-CT, `EntityDescription` represents the contents of the *Identifier* and *Description* and *RefsetLanguage* tables while `Association` models the *Relationship* tables.

1.1 Notation

Model elements are referenced using `Typewriter` font.

1.2 Identifiers

- Class names are capitalized camel case (`CodeSystemCatalog`, `String`).
- Attribute and role names are lower camel case (`version`, `sourceAndRole`).
- Enumeration values are all capital letters with undescores ('_') introduced as needed for clarity. (EMPTY, STOP_ON_ERROR)
- Method names are `lowerCamelCase`.

1.3 Stereotypes

The following stereotypes are used throughout the model:

- **dataType** - UML dataType semantics.
- **enumeration** - UML enumeration semantics.
- **interface** - UML interface semantics.
- **mixin** - a class that provides a certain functionality to be inherited by a subclass, while not meant for instantiation.
- **opt** - an optional parameter in a method call. This stereotype should always be accompanied by a 0.. 1 cardinality and is provided to make optionality visible in the modeling tool.
- **optparam** - an optional attribute in a class instance. This indicates that an implementation must be able to distinguish three alternatives: (1) add or change the attribute to whatever is in the class, (2) remove the attribute if its minimum cardinality is zero, and (3) leave the attribute unchanged.

- **delta** - a method call that changes the state of the service. Note that the semantics of this stereotype is the inverse of the is query model attribute.
- **exception** - an exception.
- **exceptionSet** - a set of exceptions, the members of which are represented by aggregation.

1.4 Read Only Attributes

The CTS2 PIM is based on a RESTful architectural style. Resources can be created, updated, read, and/or deleted, and every resource has one or more immutable characteristics - characteristics that, if changed, would change the identity of the resource itself. In the information model we distinguish these characteristics from those that can be modified by declaring the identifying characteristics as *ReadOnly*. The computational model utilizes a pattern based on this information where the *create* operation(s) supply the identifying characteristics plus any of the non-optional mutable characteristics. The *update* operations then supply a unique identifier and a set of one or more mutable characteristics to be changed.

2 Entity Description Information Model

2.1 Entity Description Model

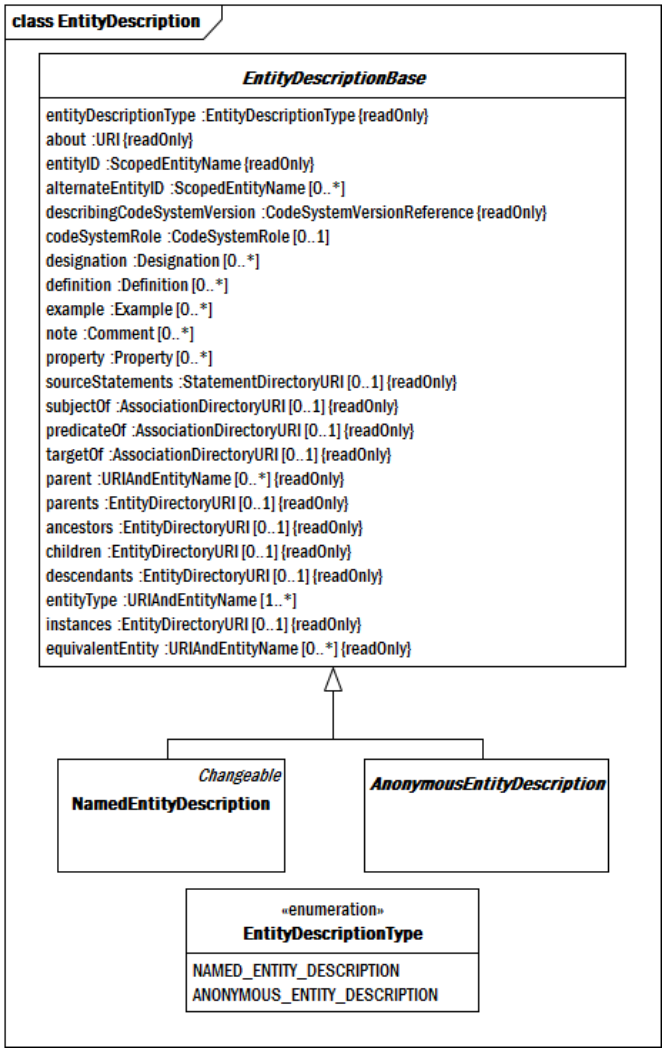

Figure 2.1 - Entity Description

`EntityDescription` represents the set of assertions about a particular entity (class, predicate, and/or individual) that are made by a specified version of a code system. It includes the set of designations that signify the entity in various languages or contexts, definitions where appropriate, examples, various forms of comments and any additional properties that are not classified elsewhere. It also contains information and links about where the entity fits in the code system hierarchy, including identifiers and, when available, references to its "parent" entities as well as references to the class(es) that it instantiates.

Note that there can be multiple `EntityDescription` entries for the same resource - one for every code system that makes assertions about it. Some of these descriptions may be fairly sparse - if a code system, for example, asserts: wine:Wine rdfs:subClassOf owl:Class, it is making assertions about both rdfs:subClassOf and owl:Class but, unless it explicitly imports these resources, the `EntityDescription` entries for these will be very minimal.

The requirement to support the OWL 2 notion of anonymous individuals has added considerable complexity to this model. Anonymous individuals appear to be a model of uniquely identifiable referential determiners, where the identity of the individual is only available in some context. As an example, the "primary residence of David Markwell" or "the fourth house on Main Street" refer to unique individuals without naming them. As in linguistics, the identity of these resources is contextual and local to the code system.

To support this class of resource, we have to create two distinct types of entity description - those with identity (e.g., sct:90708001(Kidney Disease), wine:Riesling) and those without.

2.1.1 Class AnonymousEntityDescription

An entity description that is local to a given code system version. Anonymous entity descriptions are not changeable and have no external URI.

Superclasses

- Every instance of `AnonymousEntityDescription` is also an instance of `EntityDescriptionBase`.

Invariants

1. The `entityDescriptionType` must be `ANONYMOUS_ENTITY_DESCRIPTION`.
2. The about URI of an anonymous entity description is a Local URI meaning that it cannot be used outside the context of the given service.
3. The scoping namespace of an anonymous entity description must be the same as the name of the code system version.

2.1.1.1 Class EntityDescriptionBase

The attributes common to both named and anonymous entity descriptions. `EntityDescriptionBase` is an abstract class and cannot be directly instantiated.

Attributes

- **entityDescriptionType** - discriminant that determines whether the particular description is about a named resource or it is an expression whose description constitutes its identity. See OWL[1] specification on Anonymous Individuals[2] for further detail.
- **about** - the URI that represents the entity (class, predicate, and/or individual) referenced by this `EntityDescription` about is a `PersistentURI` when entity `DescriptionType` is `NAMED_ENTITY_DESCRIPTION` and `LocalURI` when `entityDescriptionType` is `ANONYMOUS_ENTITY_DESCRIPTION`.

---

1. http://www.w3.org/TR/2009/REC-ow
2. syntax-20091027/#Anonymous Individuals

- **entityID** - the local namespace name and entity name that, together, uniquely name this referenced within the context of a CTS2 service instance. As an example, a service instance may choose to designate the entity referenced by the `about` URI of http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#CabernetSauvignon with the `namespaceName` "wine" and the `name` "CabernetSauvignon."

  The `name` portion of ScopedEntityName must uniquely name the particular entity within the context of the code system version.

- **alternateEntityID** - alternative unique identifiers that reference the `about` entity in the context of describing code system version. The `namespace` names the context from which the name (or code) is derived. Note that it is possible for the `entityId` `name` and/or one more `alternateEntityId` `names` to be represented in the `designation` attribute as well. This would occur when the `name` serves a dual role of unique identifier and human readable label.

  An example of `alternateEntityId` would be the SNOMED-CT fully specified name, which serves as a unique identifier in human readable form. This might or might not also be considered a designation depending on the context and use case. Similarly, the SNOMED-CT Clinical Terms V3 Identifier (CTV3) and SNOMED-3 identifiers would appear here as well, as would the HL7's case sensitive and case insensitive unit of measure identifiers.

- **describingCodeSystemVersion** - a reference to the version of an ontology or code system that makes the set of assertions contained in this description. Note that this references the "asserting," not the "containing" code system version. As an example, the Wine Ontology imports the Food Ontology, which, in turn contains descriptions of PotableLiquid. A service that represented a (version of) the Wine Ontology would contain an entity description `about` "http://www.w3.org/TR/2003/PR-owl-guide-2003 1209/food#PotableLiquid" (namespaceName: food, name: PotableLiquid). The describingCodeSystemVersion, however, would be "http://www.w3.org/TR/2003/PR-owl-guide2003 1209/wine#" even though this entity was described in the food ontology. Note also that if both the food and wine ontology were represented in a service, the service would contain two entity descriptions - one from the food ontology perspective and one from the wine ontology perspective.

- **codeSystemRole** - If present, identifies the role that the code system plays in the description of the entity. If absent, the role is unknown.

- **designation** - a "string of (UNICODE) characters in a given natural language, such as English or Japanese." [3] A designation provides the strongest clues as to the meaning of a class, predicate or individual. While designations are not mandatory in an entity description, as all that may be available to a service is a code, service implementers are strongly encouraged to provide at least one preferred or alternate designation to make the description visible to text search engines.

- **definition** - an explanation of the intended meaning of a concept. An `EntityDescription` may have multiple `definitions`, each derived from a different source, represented in a different language or having a different purpose. Definitions apply only to classes and predicates, not individuals.

- **example** - an example of an instance or instances of the referenced entity, typically written for human consumption. Examples apply only to classes and predicates, not individuals.

- **note** - a note or comment about the history, status, use or other descriptions about the `EntityDescription`. Observe that `note`, as used in the CTS2 specification, does include examples or definitions.

- **property** - additional "non-semantic" (annotation) assertions about the entity being described that do not fit into the `designation`, `definition`, `note`, or `example` or `entityId` categories.

---

3. http://www.w3.org/TR/2009/CR-skos-reference-20090317/#labels

- **sourceStatements** - a `DirectoryURI` that resolves to the list of `statements` that were used in the assembly of this description. `Statements` need only be present in service instances that support the STATEMENT profile, and serves as a bridge between atomic structure contained in the source resource and the CTS2 rendering.
- **subjectOf** - an `AssociationDirectoryURI` that resolves to a list of active associations ("semantic" assertions) in which the `about` entity appears as a subject and that are `assertedBy` the `describingCodeSystemVersion`. `subjectOf` is only included in an `EntityDescription` when resolution would yield at least one result. This attribute will only be present in CTS2 service implementations that support the ASSOCIATION QUERY profile.
- **predicateOf** - a `DirectoryURI` that resolves to the list of active associations ("semantic" assertions) in which the `about` entity appears as a predicate and that are `assertedBy` the `describingCodeSystemVersion`. `predicateOf` is only included in an `EntityDescription` when resolution would yield at least one result. This attribute will only be present in CTS2 service implementations that support the ASSOCIATION QUERY profile.
- **targetOf** - a `DirectoryURI` that resolves to the list of active associations ("semantic" assertions) in which the `about` entity appears as either a direct target or the target of a contained BNode and that are `assertedBy` the `describingCodeSystemVersion`. `targetOf` is only included in an `EntityDescription` when resolution would yield at least one result. This attribute will only be present in CTS2 service implementations that support the ASSOCIATION QUERY profile.
- **parent** - the set of direct "parents" asserted by `describingCodeSystemVersion`. It is the responsibility of the service to determine what predicate(s) represent "parent/child" relationships. Typically "parent" is associated with "rdfs:subClassOf" in the OWL/RDF world and skos:broader/skos:narrower in the SKOS environment. It is possible, however for some code systems to have other relationships as well. As an example, some of the Open Biomedical Ontologies (OBO) use a mixture of subClassOf and partOf relationships.
- **parents** - a `DirectoryURI` that resolves to the list of direct "parents" asserted by `describingCodeSystemVersion`. This returns the parent elements above.
- **ancestors** - a `DirectoryURI` that resolves to the transitive closure of the "parents" relationship(s). The primary purpose for this attribute is to provide a handle for subsumption queries. As an example, to determine whether Class X was a subclass of ClassY, one would query whether the `EntityReference` to Y was a member of X.ancestors.
- **children**- a `DirectoryURI` that resolves to the list of direct "children" asserted by `describingCodeSystemVersion`. As with parent, it is the responsibility of the service to determine what predicate(s) represent "parent/child" relationships. children may also include entity references that appear in the target of bnodes if they are determined to be such by the service.
- **descendants** - a `DirectoryURI` that resolves to the transitive closure of the "children" relationship(s). The primary purpose for this attribute is to provide a handle for subsumption queries. As an example, a second way to determine whether Class X was a subclass of ClassY, one would query whether the `EntityReference` to X was a member of Y.descendants.
- **entityType** - the set of type(s) which the `entityReference` is an instance of. Because this is a terminology service, `entityType` must include one of owl:Class, owl:Individual, rdf:Property or skos:Concept, although it may carry many other types as well.
- **instances** - a `DirectoryURI` that resolves to the list of entities that asserted to be instances of the type represented by about URI. This element is present only if resolution will return a non-empty set.
- **equivalentEntity** - an entity that has been determined to be equivalent to the `about` entity in the context of the assertions made by `describingCodeSystemVersion`.

Invariants

1. Every `entityID` and `alternateEntityID` for a given entity must be unique.

2.1.1.2 Class NamedEntityDescription

An entity description that is referenced by a globally unique external URI.

Superclasses

- Every instance of `NamedEntityDescription` is also an instance of `EntityDescriptionBase`.
- Every instance of `NamedEntityDescription` is also an instance of `Changeable`.

Invariants

1. entity `DescriptionType` must be `NAMED_ENTITY_DESCRIPTION`.
2. The about URI of a named entity description is an external, globally unique URI.
3. The `entryID` of a `NamedEntityDescription` is the about URI.

2.1.1.3 Enum EntityDescriptionType

A discriminant that identifies whether a subclass of `EntityDescriptionBase` is a `NamedEntityDescription` or an `AnonymousEntityDescription`.

Enumeration Types:

- **NAMED_ENTITY_DESCRIPTION** - the `EntityDescription` describes an entity that has a published URI.
- **ANONYMOUS_ENTITY_DESCRIPTION** - the `EntityDescription` describes an entity whose identity is implicit and known only in the context of the describing code system version.

2.2 Designations

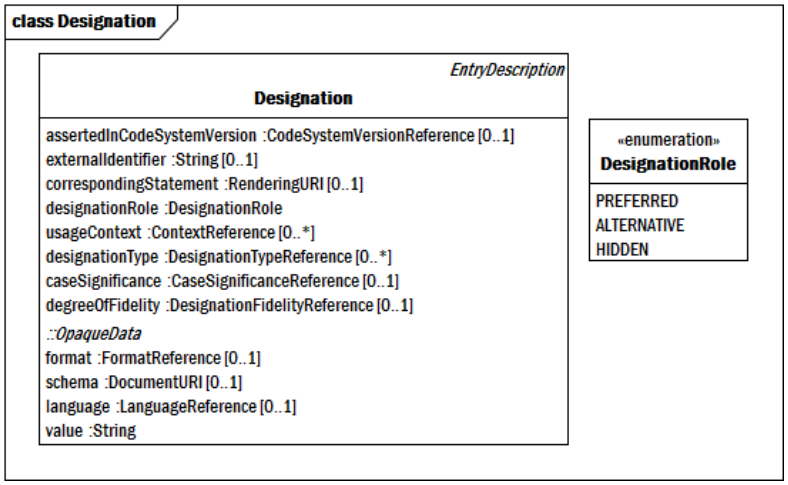

Figure 2.2 - Designation

2.2.1 Class Designation

A `Designation` is identical to a Lexical Label[4] as defined in SKOS[5]. It is a "string of (UNICODE) characters in a given natural language, such as English or Japanese." "These labels provide the strongest clues as to the meaning of a (SKOS concept)." The CTS2 specification generalizes the label use to include not only SKOS concepts but any class, predicate or individual - collectively referred to as `Entity`. Note that `Designation` is intended only for identifiers that are interpreted by human beings. Numeric codes and other machine readable identifiers should be recorded as `ScopedEntityNames`.

Superclasses

- Every instance of `Designation` is also an instance of `EntryDescription`.

Attributes

- **assertedInCodeSystemVersion** - the code system version in which this designation is asserted. If absent, it is the same as the `EntityDescriptioncontainingCodeSystemVersion`.
- **externalIdentifier** - an identifier that is assigned to this designation / entity pair by the authoring body. As an example this would carry the SCTID for SNOMED-CT concepts.
- **correspondingStatement** - a URI that, when de-referenced, provides the equivalent `Statement` that forms this designation. This is only present when the STATEMENT profile is supported by the implementing service.
- **designationRole** - an indicator that determines the particular role that this designation plays.

---

4. http://www.w3.org/TR/skos-reference/#labels
5. http://www.w3.org/2004/02/skos/intro

- **usageContext** - the context(s) in which this designation is applicable.
- **designationType** - the type of designation (e.g., abbreviation, eponym, short name, long name, etc.) of the given designation.
- **caseSignificance** - identifies the significance that case plays in the value of this particular designation.
- **degreeOfFidelity** - a measure of how closely this designation actually matches the intended meaning of the target entity. Typical values might represent “exact,” “approximate,” “broader,” “narrower,” etc.

2.2.1.1 Enum DesignationRole

The role that a designation plays in respect to the referenced `Entity`. `DesignationRole` is intended to convey the same semantics as the three types of SKOS Label[6] - skos:prefLabel (`PREFERRED`), skos:altLabel (`ALTERNATIVE`), and skos:hiddenLabel (`HIDDEN`). The SKOS consistency rules with respect to preferred labels also apply to designation with two exceptions:

1. There can be more than one preferred label for the same language if the usage Context is different.
2. If the service implementation supports structured data, it is possible to have more than one preferred label for the same language and context as long as each has a different `format`.

Enumeration Types:

- **PREFERRED** - The preferred human-readable representation of the entity in a given language, context (aka. domain in SKOS terms) and format. `PREFERRED` is equivalent to skos:prefLabel.
- **ALTERNATIVE** - The acceptable alternative human-readable representation of the entity in a given language, context (aka. domain in SKOS terms) and format. The term “synonym” is often used to indicate an alternate designation although strictly speaking, “synonym” is a term-term relationship. `ALTERNATIVE` is equivalent to skos:altLabel
- **HIDDEN** - The designation may be used to represent the entity under certain circumstances, but it is not normally displayed. Hidden designations are used primarily for search terms and typically represent common misspellings or deprecated usages. `HIDDEN` is equivalent to skos:hiddenLabel

6. http://www.w3.org/TR/skos-reference/#labels

2.3 Types of Entity Description

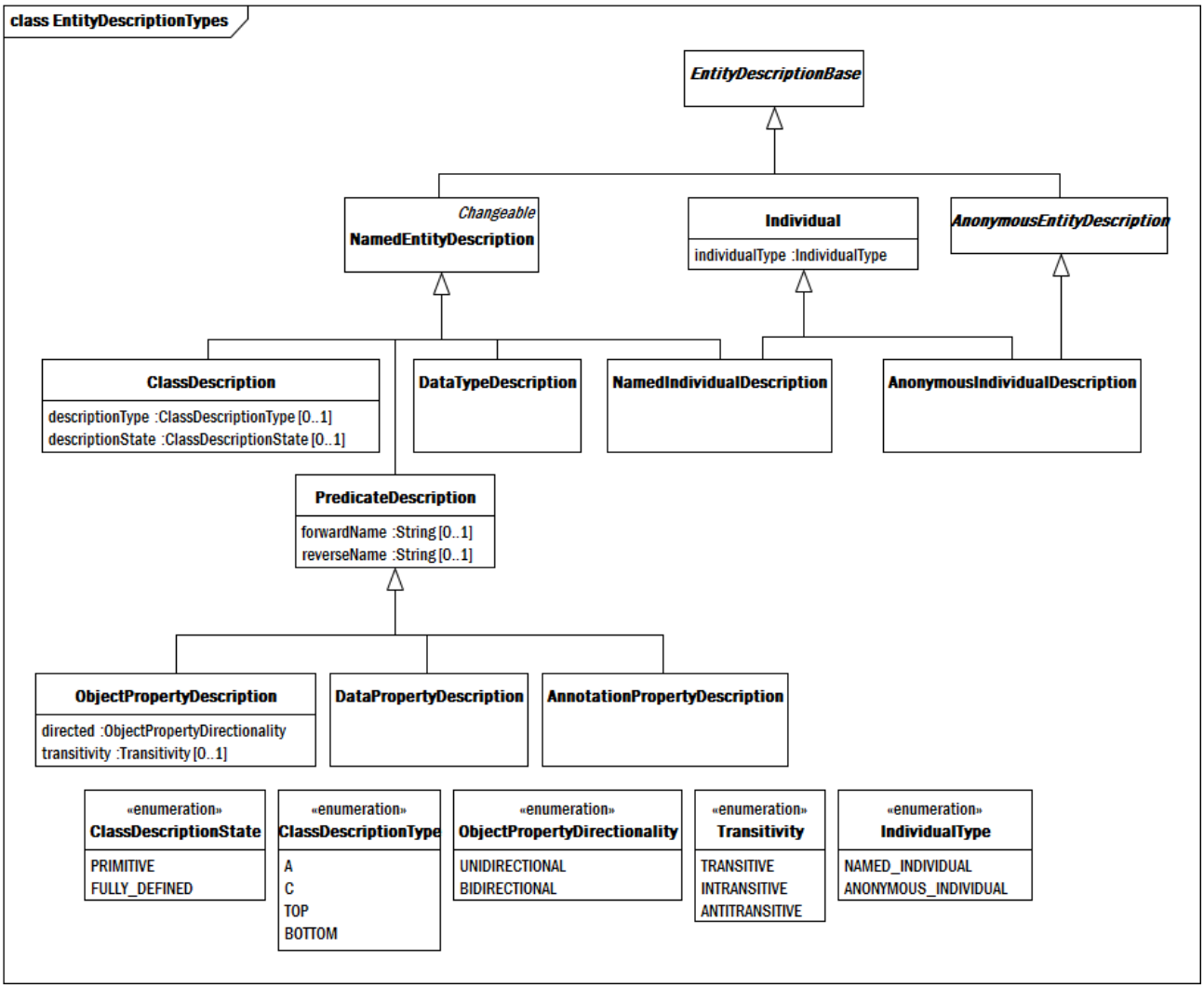

Figure 2.3 - Types of EntityDescription

The diagram above shows the various flavors of `EntityDescription` that are explicitly called out by the CTS2 specification. This diagram is a variation on a similar diagram in Entities, Literals, and Anonymous Individuals[7] from the OWL2 syntax[8] specification. The degree of formality exhibited by the various code systems that can be imported into the CTS2 specification can vary considerably. On one end of the spectrum, there are reasonably simple resources that are expressed as lists of code/value pairs. These would be loaded as simple `NamedEntityDescriptions`. More elaborate resources may make a distinction between "concepts" and "roles," which, depending on how specific the notion of "concept" was would either be loaded as their corresponding `ClassDescriptions` and `PredicateDescriptions`. RDF and OWL based resources would take up the whole spectrum of types.

7. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Entities.2C Literals.2C and Anonymous Individuals
8. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/

It should be noted, however, that while not explicitly stated in the diagram, the assumption is that the leaf classes are disjoint - something declared as an individual is assumed not to be a class, which presents an interesting issue to the model designers. One solution would be to specify all of the attributes of class, predicate, etc. at the `EntityDescriptionBase` level and then to remove the inapplicable ones in the leaf nodes. As an example, we could have added `forwardName` and `reverseName` to `EntityDescriptionBase` and then declared that they are not present in class descriptions. We felt, however, that this was unnecessarily complex and, instead, will count on providers of more sophisticated services to use a variation on punning or to extend the `EntityDescriptionBase` class to support this.

2.3.1 Class AnnotationPropertyDescription

A `PredicateDescription` that describes a predicate used for "lexical" annotation of an entity or other resource. Annotation properties are "semantically weak" (see below) in the sense that they are not inherited through subclass associations.

"For annotations properties note that annotations are not 'semantic-free' under the OWL 2 RDF-Based Semantics. Just like every other triple or set of triples occurring in an RDF graph, an annotation is assigned a truth value by any given OWL 2 RDF-Based interpretation. Hence, although annotations are meant to be 'semantically weak,' i.e., their formal meaning does not significantly exceed that originating from the RDF Semantics specification, adding an annotation may still change the meaning of an ontology." [9]

Superclasses

- Every instance of `AnnotationPropertyDescription` is also an instance of `PredicateDescription`.

Invariants

1. entityType must include owl:AnnotationProperty.

2.3.1.1 Class AnonymousIndividualDescription

An "intensional" description of an individual that describes the individual by its characteristics without supplying its identity. As an example, the description "The sole occupant of 773 Main Street in Sioux City, South Dakota" would reference an anonymous individual.

"If an individual is not expected to be used outside a particular ontology, one can use an anonymous individual, which is identified by a local node ID rather than a global IRI. Anonymous individuals are analogous to blank nodes in RDF." [10]

Superclasses

- Every instance of `AnonymousIndividualDescription` is also an instance of `Individual`.
- Every instance of `AnonymousIndividualDescription` is also an instance of `AnonymousEntityDescription`.

2.3.1.2 Class ClassDescription

The description of an entity that is a Class, Type, or "Concept." "Classes can be understood as sets of individuals." [11]

---

9. http://www.w3.org/TR/2009/REC-owl2-rdf-based-semantics-20091027/#Introduction .28Informative.29
10. http://www.w3.org/TR/owl2-syntax/#Anonymous Individuals

Superclasses

- Every instance of `ClassDescription` is also an instance of `NamedEntityDescription`.

Attributes

- **descriptionType** - the specific role that the class description plays, if known.
- **descriptionState** - an indicator that states whether the class description contains only necessary assertions (`PRIMITIVE`) or is considered to be both necessary and sufficient assertions (`FULLY_DEFINED`).

Invariants

1. entityType must include one owl:Class.

2.3.1.3 Class DataPropertyDescription

A `PredicateDescription` whose domain is a set of literals.

"Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes." [12]

Superclasses

- Every instance of `DataPropertyDescription` is also an instance of `PredicateDescription`.

Invariants

1. entityType must include owl:DataProperty.

2.3.1.4 Class DataTypeDescription

The description of a data type, which, in this context, represents a constraint on the set of possible values in a text or literal field. "Datatypes are entities that refer to sets of data values."

Superclasses

- Every instance of `DataTypeDescription` is also an instance of `NamedEntityDescription`.

Invariants

1. entityType must include owl:DataType.

2.3.1.5 Class Individual

An instance of a category or class. "Individuals in the OWL 2 syntax represent actual objects from the domain."

Attributes

- **individualType** - a discriminant that determines whether the `Individual` is anonymous or named.

---

11. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Classes
12. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Object Properties

Invariants

1. entityType must include owl:Individual.

2.3.1.6 Class NamedIndividualDescription

A description of an individual that has identity.

Superclasses

- Every instance of `NamedIndividualDescription` is also an instance of `Individual`.
- Every instance of `NamedIndividualDescription` is also an instance of `NamedEntityDescription`.

Invariants

1. entityType must include owl:NamedIndividual.

2.3.1.7 Class ObjectPropertyDescription

The description of a “semantic” predicate. “Object properties connect pairs of individuals.”

Superclasses

- Every instance of `ObjectPropertyDescription` is also an instance of `PredicateDescription`.

Attributes

- **directed** - an indicator that states whether the object property is unidirectional (i.e., requires an inverse to be explicitly declared) or is bidirectional.
- **transitivity** - transitivity characteristics of the property. If not supplied, transitivity characteristics are either unknown or not applicable (i.e., domain ? range = 0).

Invariants

1. entityType must include owl:ObjectProperty.

2.3.1.8 Class PredicateDescription

`PredicateDescription` covers the spectrum of entities deemed “Role” in description logic, “Relation” in predicate logic, “Property” in RDF and OWL, and “Association” in Ontylog DL. It describes the nature and the purpose of the role, not the individual elements.

Note that section 2.4.2.11 of the HL7 SFM calls for an attribute called “associationKind.” While we are not absolutely certain what this was intended to represent, we believe that its intent is probably subsumed under the various individual types as well as the transitivity attribute.

`forwardName` and `reverseName` are called for in the SFM. The way that they are represented in this model lacks the ability to represent them in a multi-lingual environment that may cause difficulties down the line. Service implementers who are concerned about this aspect are encouraged to extend the service to allow forward and reverse Name to be returned in the language appropriate to the service viewer.

The `directed` attribute is actually more of a characteristic of the type of logic being used than the individual properties, but returning it as an attribute of a property seems sufficient.

The HL7 SFM ruleSetId construct is not supported in this specification.

Superclasses

- Every instance of `PredicateDescription` is also an instance of `NamedEntityDescription`.

Attributes

- **forwardName** - the name assigned to the predicate when applied in the source to target direction. The primary purpose of this attribute is to provide some notion of "directional" semantics, as predicates named "part" or "broader term" often lack sufficient information to understand where the whole or broader component goes. Implementers will note that, as opposed to official designations, `forwardName` and `reverseName` are not keyed by language. Implementers are encouraged to develop designations and use the `forwardName` and `reverseName` tags if a richer model is desired.
- **reverseName** - the name assigned to the predicate when applied in the target to source direction. The primary purpose of this attribute is to provide some notion of "directional" semantics, as predicates named "part" or "broader term" often lack sufficient information to understand where the whole or broader component goes.

Invariants

1. entityType must include rdf:Property.

2.3.1.9 Enum ClassDescriptionState

Determines whether a class description is considered to consist of necessary or necessary and sufficient definitions of the member individuals.

Enumeration Types:

- **PRIMITIVE** - The defining assertions of the class description are necessary for all individuals that are members of the class. If an individual is asserted to be a member of the class, it can be asserted that it possesses all of the defining characteristics.
- **FULLY_DEFINED** - The defining assertions of the class description are both necessary and sufficient. If an individual is asserted to be a member of the class, it can be asserted that it possesses all of the defining characteristics and, if an individual is determined to possess all of the defining characteristics in the class definition, it can be asserted to be a member of the class.

2.3.1.10 Enum ClassDescriptionType

The type, from a description logic perspective, of a class description.

Enumeration Types:

- **A** - an atomic class description
- **C** - a complex class description
- **TOP** - a class that represents the set of all individuals in the domain of discourse represented by the containing code system or ontology.
- **BOTTOM** - a class that represents the set that contains no individuals in the context of the containing domain or ontology.

2.3.1.11 Enum IndividualType

A discriminant that identifies the type of individual description.

Enumeration Types:

- **NAMED_INDIVIDUAL** - the `Individual` type is a `NamedIndividualDescription`.
- **ANONYMOUS_INDIVIDUAL** - the `Individual` type is an `AnonymousIndividualDescription`.

2.3.1.12 Enum ObjectPropertyDirectionality

Indicates whether the semantics of an object property are unidirectional (i.e., *r(a, b) ? r(b, a)* or bidirectional *(r(a, b) ? r(b, a))*.

Enumeration Types:

- **UNIDIRECTIONAL** - *r(a, b) ? r(b, a)*
- **BIDIRECTIONAL** - *r(a, b) ? r(b, a)*

2.3.1.13 Enum Transitivity

An indicator that determines whether an object property is considered to be transitive.

Enumeration Types:

- **TRANSITIVE** - asserts *Va,b,c: C op(a,b) Ap(b,c) ? p(a,c)*
- **INTRANSITIVE** - asserts *Va,b,c: C o (p(a,b) Ap(b,c) ? p(a,c))*
- **ANTITRANSITIVE** - asserts *V a, b, c: C o p(a, b) A p(b, c) ? p(a, c)*

2.4 Entity Description Directories

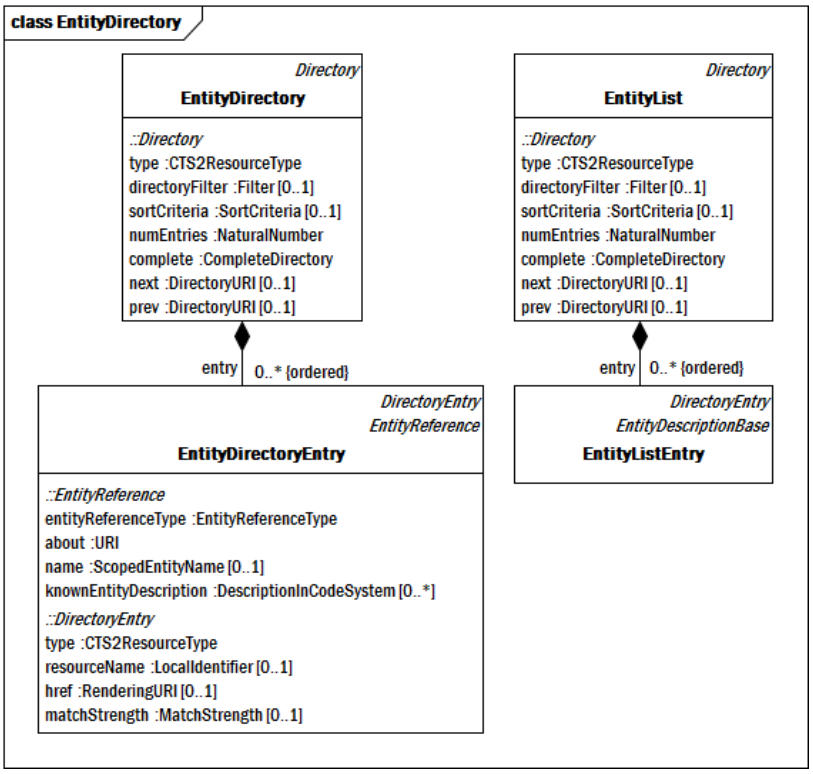

Figure 2.4 - Entity Description Directory

2.4.1 Class EntityDirectory

A directory of `Entity` resources that meet a specified criteria.

Superclasses

- Every instance of `EntityDirectory` is also an instance of `Directory`.

Attributes

- entry - an entry in the directory

2.4.1.1 Class EntityDirectoryEntry

A synopsis of an `Entity` along with information about how to access the complete resource.

Superclasses

- Every instance of `EntityDirectoryEntry` is also an instance of `DirectoryEntry`.

- Every instance of `EntityDirectoryEntry` is also an instance of `EntityReference`.

2.4.1.2 Class EntityList

A collection of complete `Entity` resources that meet a specified criteria.

Superclasses

- Every instance of `EntityList` is also an instance of `Directory`.

Attributes

- **entry** - an entry in the list.

2.4.1.3 Class EntityListEntry

An instance of `Entity` that meets a specified filter criteria.

Superclasses

- Every instance of `EntityListEntry` is also an instance of `DirectoryEntry`.
- Every instance of `EntityListEntry` is also an instance of `EntityDescriptionBase`.

3 Entity Description Services

3.1 Entity Description Read Service

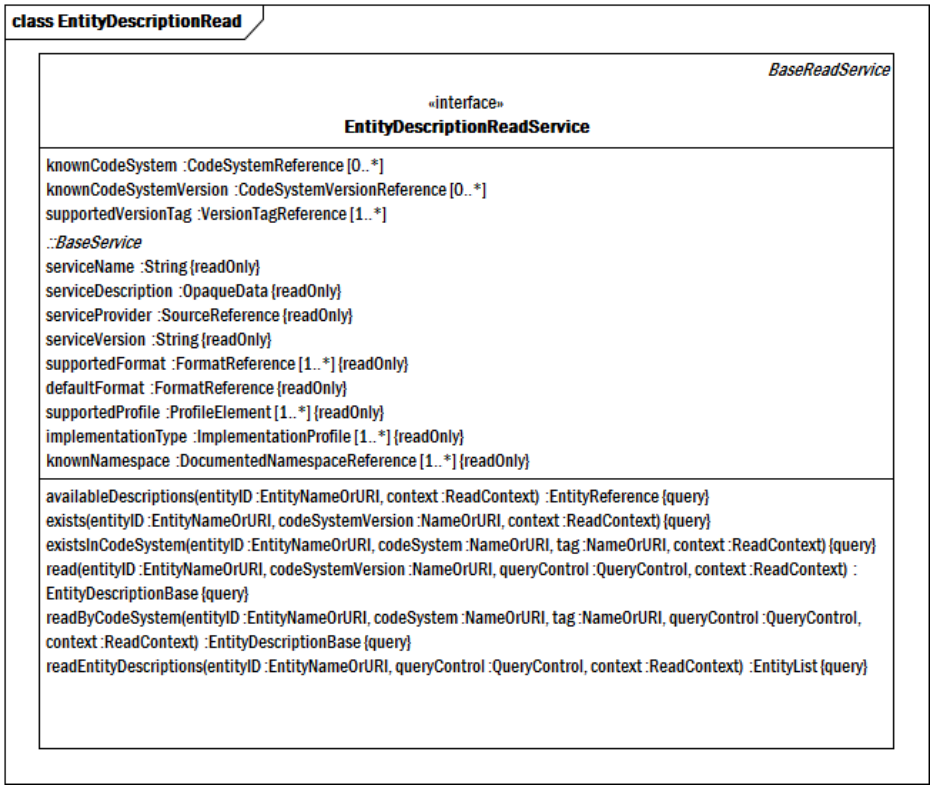

Figure 3.1 - Entity Description Read Service

The `EntityDescriptionReadService` provides direct access to `EntityDescriptions` contained in specific code system versions. `EntityDescriptions` are identified by either the `about` URI or a namespace/name pair. They can be accessed via a named code system version or by identifying a code system and supplying a version tag that is assigned to a particular version at a particular point in time.

3.1.1 Interface EntityDescriptionReadService

A service that provides direct read access to `EntityDescriptions` by namespace/name or URI in the context of a particular code system or code system version.

Superclasses

- Every instance of `EntityDescriptionReadService` is also an instance of `BaseReadService`.

Attributes

- **knownCodeSystem** - the name and URI of a code system that is known to this service.

- **knownCodeSystemVersion** - the name and URI of the code system version that are recognized by this service.
- **supportedVersionTag** - the set of version tags supported by this service. All service implementations must support the `CURRENT` version tag.

Invariants

1. Service must be presented in the `supportedProfile` list.
2. All known code systems include URIs.
3. Every code system reference is also included in known Namespace.

3.1.1.1 Operation: availableDescriptions

Return the available `EntityDescriptions` for the referenced entity. This returns a list of the latest versions of all of the code systems that make assertions about the referenced entity.

Input Parameters

- **entityID** - the namespace/name or URI of the target entity (Type: `EntityNameOrURI`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityReference`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

3.1.1.2 Operation: exists

Determine if the specified entity description exists in the specified code system version.

Input Parameters

- **entityID** - the namespace/name or URI of the target entity (Type: `EntityNameOrURI`).
- **codeSystemVersion** - the local identifier or URI of the describing code system version (Type: `NameOrURI`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `Boolean`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.

- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

3.1.1.3 Operation: existsInCodeSystem

Determine if the specified entity description exists in the specified code system.

Input Parameters

- **entityID** - the namespace/name or URI of the target entity (Type: `EntityNameOrURI`).
- **codeSystem** - the local identifier or URI of the describing code system version (Type: `NameOrURI`).
- **tag**$_{OPT}$ - the name or URI of a version tag that resolves to the code system version describing the entity. If omitted, tag defaults to CURRENT (Type: `NameOrURI`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `Boolean`

Exceptions

- **UnknownCodeSystem** - The referenced `CodeSystem` is not recognized by the service.
- **UnsupportedVersionTag** - The `versionTag` is not recognized by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

3.1.1.4 Operation: read

Retrieve an entity description from a specified code system version.

Input Parameters

- **entityID** - the namespace/name or the ExternalURI of the entity whose description is to be retrieved (Type: `EntityNameOrURI`).
- **codeSystemVersion** - the local identifier or the URI of the code system version describing the referenced entity (Type: `NameOrURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDescriptionBase`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

Preconditions

1. The code system version name or URI must be known to the service.

3.1.1.5 Operation: readByCodeSystem

Retrieve the specified entity description from the specified code system.

Input Parameters

- **entityID** - the namespace/name or URI of the referenced entity (Type: `EntityNameOrURI`).
- **codeSystem**$_{OPT}$ - the local identifier or URI of the referencing code system. If not supplied, the code system identified as the PRIMARY code system will be used instead. If no code system is identified as primary and there is only one code system that has descriptions for the entity, it will be used. If more than one is present, the service may either treat this call as an availableDescriptions call or return an error. (Type: `NameOrURI`)
- **tag**$_{OPT}$ - the name or URI of a version tag that resolves to the code system version describing the entity. If omitted, tag defaults to CURRENT. (Type: `NameOrURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDescriptionBase`

Exceptions

- **UnsupportedVersionTag** - The `versionTag` is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

3.1.1.6 Operation: readEntityDescriptions

Retrieve all of the `EntityDescriptions` for the supplied entity Id. This function will return the current `EntityDescription` for every describing code system version that makes non-semantic assertions about the referenced entity.

Input Parameters

- **entityID** - the namespace/name or URI of the target entity. (Type: `EntityNameOrURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityList`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.

- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not OPEN.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

3.2 Entity Description Query Service

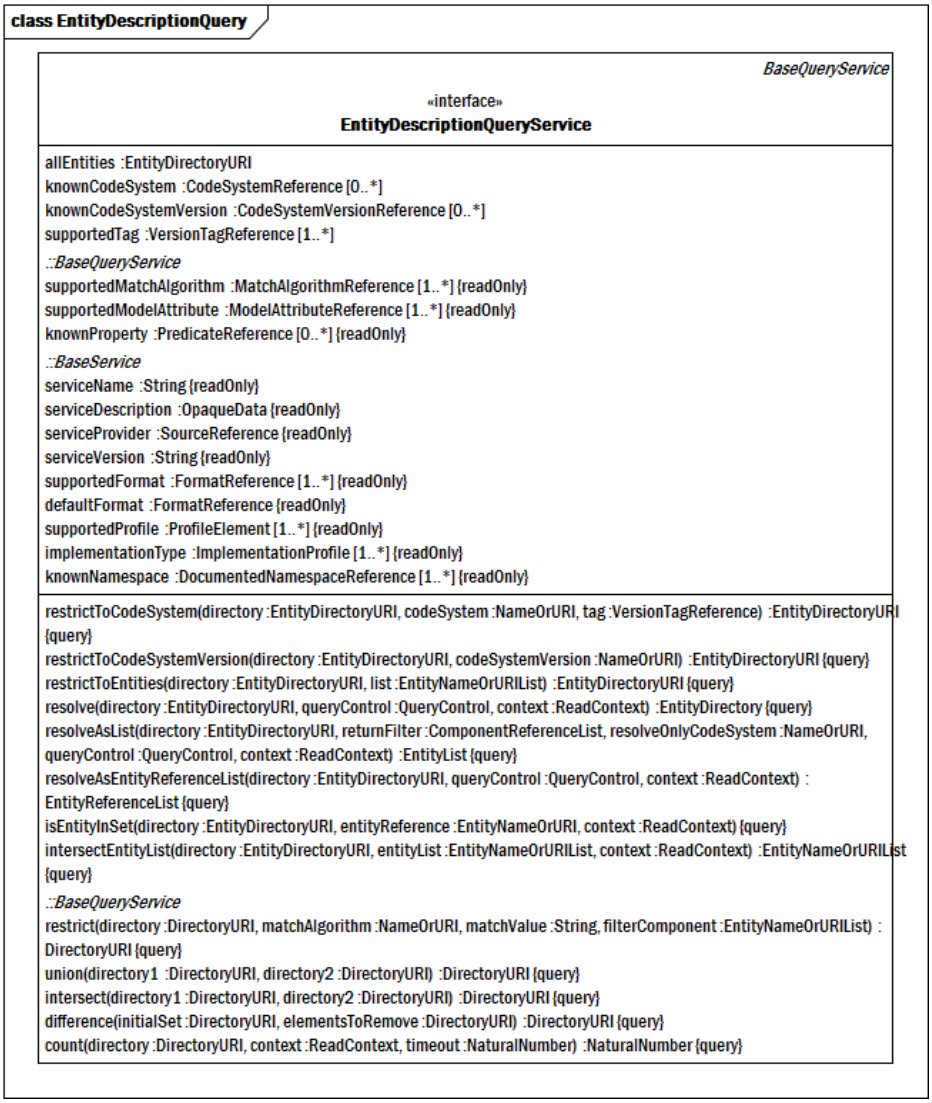

Figure 3.2 - Entity Description Query Service

`EntityDescriptionQueryService` provides the functionality necessary to select subsets of the set of `EntityDescriptions` known to the service via various restriction criteria, the intersection, union, and difference of these sets and the ability to render the result of these queries in different formats. It also provides tools to determine the presence or absence of `EntityReferences` in these sets.

3.2.1 Interface EntityDescriptionQueryService

A service that allows the selection, manipulation, and rendering of sets of `EntityDescriptions`.

Superclasses

- Every instance of `EntityDescriptionQueryService` is also an instance of `BaseQueryService`.

Attributes

- **allEntities** - a `DirectoryURI` that resolves to all entities known to the service.
- **knownCodeSystem** - the code systems that are known to the service. This is a list rather than a directory URI because of the requirement that the ENTITY_QUERY service be independent from the CODE_SYSTEM_CATALOG_QUERY, and CODE_SYSTEM_VERSION_CATALOG_QUERY profiles.
- **knownCodeSystemVersion** - the code system versions known to the service. As with `knownCodeSystem`, this is a list rather than a directory URI because of the requirement that the ENTITY_QUERY service be independent from the CODE_SYSTEM_CATALOG_QUERY and CODE_SYSTEM_VERSION_CATALOG_QUERY profiles.
- supportedTag - the set of version tags that are recognized by this service.

3.2.1.1 Operation: restrictToCodeSystem

Return an `EntityDirectoryURI` that references only the `EntityReferences` in directory that have assertions about the (EntityDescriptions) in the tagged version of the supplied code system. Note that this does not constrain the `knownEntityDescription` attribute within the `EntityReference` itself - an `EntityReference` that has assertions in code system A and B would still have two `knownEntityDescription` entries even after this restriction is applied. Further constraint can be accomplished via the `resolve` functions.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **codeSystem** - the name or URI of the code system which filters the set of entities to be returned (Type: `NameOrURI`).
- **tag**$_{OPT}$ - the name or URI of version tag that identifies which code system version is to be used for the purposes of restriction. If omitted, the CURRENT tag will be used (Type: `VersionTagReference`).

Return Type: `EntityDirectoryURI`

Exceptions

- **UnknownCodeSystem** - The referenced `CodeSystem` is not recognized by the service.
- **UnsupportedVersionTag** - The `versionTag` is not recognized by the service.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.
- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.

3.2.1.2 Operation: restrictToCodeSystemVersion

Return an `EntityDirectoryURI` that references only the `EntityReferences` in directory that have assertions about the (`EntityDescriptions`) in the named code system version.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **codeSystemVersion**$_{OPT}$ - the name or URI of a code system version used to restrict the set of entities in directory (Type: `NameOrURI`).

Return Type: `EntityDirectoryURI`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.

3.2.1.3 Operation: restrictToEntities

Return an `EntityDirectoryURI` that represents the set of `EntityReferences` common to directory and list.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **list** - the list of namespace/names or URIs to intersect with the directory (Type: `EntityNameOrURIList`)

Return Type: `EntityDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.2.1.4 Operation: resolve

Return an `EntityDirectory` that contains the set of `EntityReferences` identified by directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).

- **context**$_{OPT}$ - parameters that control the language, date, and time and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDirectory`

Exceptions

- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.

3.2.1.5 Operation: resolveAsList

Return a directory containing the set of `EntityDescriptions` identified by directory. One `EntityDescription` entry will be returned for each known `EntityDescription` in the corresponding `EntityReference`. This means that it is possible for there to be zero, one, or more `EntityDescriptions` returned per `EntityReference`. The `restrictToCodeSystem` or `restrictToCodeSystemVersion` operations can be used to constrain the set of references to a single code system.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **returnFilter**$_{OPT}$ - a list of zero or more component references. If present, the returned list entries will contain only the required elements and attributes plus any elements or attributes named in the filter. If absent all elements and attributes will be returned. (Type: `ComponentReferenceList`).
- **resolveOnlyCodeSystem**$_{OPT}$ - the name or URI of a code system. If present, the service will return only the EntityDescriptions whose describingCodeSystemVersion.codeSystem matches the supplied parameter. If omitted, multiple `EntityDescriptions` may be returned per `EntityReference`. (Type: `NameOrURI`)

- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date, and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityList`

Exceptions

- **UnknownCodeSystem** - The referenced `CodeSystem` is not recognized by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.

3.2.1.6 Operation: resolveAsEntityReferenceList

Return a complete `EntityReferenceList` that contains all of the `EntityReferences` identified by directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior. In this case, the ordering parameter is ignored and `maxToReturn` regulates the maximum number of entries in the list - if more than `maxToReturn` entries are available, an error will be returned instead. (Type: `QueryControl`)
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityReferenceList`

Exceptions

- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

3.2.1.7 Operation: isEntityInSet

Determine whether the supplied entity name or URI is in the set referenced by directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **entityReference** - the name or URI of the target entity (Type: `EntityNameOrURI`).
- **context** - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `Boolean`

Exceptions

- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.

- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.2.1.8 Operation: intersectEntityList

Return an `EntityReferenceList` that represents the intersection of the supplied list and the directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `EntityReferences` (Type: `EntityDirectoryURI`).
- **entityList** - a list of namespace/names or URIs of the entities to be intersected (Type: `EntityNameOrURIList`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityNameOrURIList`

Exceptions

- **InvalidDirectoryURI** - The supplied `directory` URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied `directory` URI is not the one required by the service invocation.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3 Entity Description History Service

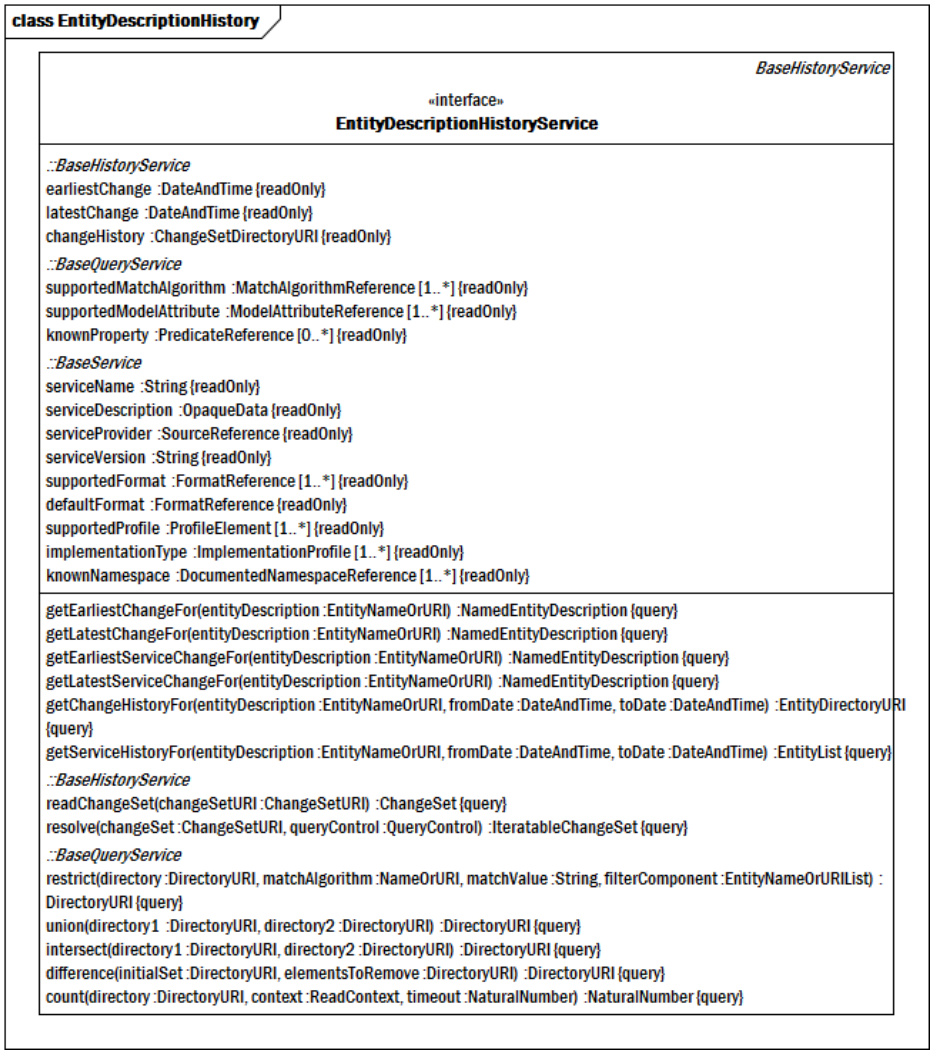

Figure 3.3 - Entity Description History

The `EntityDescriptionHistoryService` allows the client to find the first change, last change, and the set of changes that occurred to an `EntityDescription` in a particular range of time. Changes can be viewed either from the perspective of the service - determining when a change became visible to the service clients or from the perspective of the publisher - determining when the author of the change stated that it was to become active.

3.3.1 Interface EntityDescriptionHistoryService

A service that provides a historical perspective on what changes occurred to a given `EntityDescription`.

Superclasses

- Every instance of `EntityDescriptionHistoryService` is also an instance of `BaseHistoryService`.

3.3.1.1 Operation: getEarliestChangeFor

Select from the set of `ChangeSets` that contain one or more changes for the referenced entity the `ChangeSet` that has the earliest `officialEffectiveDate` and return the first change for the named entity in the set.

Input Parameters

- **entityDescription** - the namespace/name or URI of the `entityDescription` being queried (Type: `EntityNameOrURI`).

Return Type: `NamedEntityDescription`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3.1.2 Operation: getLatestChangeFor

Select from the set of `ChangeSets` that contain one or more changes for the referenced entity the `ChangeSet` that has the latest `officialEffectiveDate` and return the last change for the named entity in the set.

Input Parameters

- **entityDescription** - the namespace/name or URI of the entityDescription being queried (Type: `EntityNameOrURI`).

Return Type: `NamedEntityDescription`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3.1.3 Operation: getEarliestServiceChangeFor

Get the first change that was applied to the named entity from the perspective of the service.

Input Parameters

- **entityDescription** - the namespace/name or URI of the entityDescription being queried (Type: `EntityNameOrURI`).

Return Type: `NamedEntityDescription`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3.1.4 Operation: getLatestServiceChangeFor

Get the last change that was applied to the named entity from the perspective of the service.

Input Parameters

- **entityDescription** - the namespace/name or URI of the entityDescription being queried (Type: `EntityNameOrURI`).

Return Type: `NamedEntityDescription`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3.1.5 Operation: getChangeHistoryFor

Return a directory of `EntityDescriptions` matching the supplied namespace/name or URI whose `ChangeSet officialEffectiveDate` falls between `fromDate` and `toDate`. The order of the dates controls the return order - if `fromDate` is later than `toDate` the return order is latest to earliest, otherwise it is earliest to latest.

Input Parameters

- **entityDescription** - The namespace/name or URI of the entityDescription being queried (Type: `EntityNameOrURI`).
- **fromDate**$_{OPT}$ - The starting date. If later than toDate, the return order is latest to earliest, otherwise it is earliest to latest. (Type: `DateAndTime`)
- **toDate**$_{OPT}$ - The end date. If later than fromDate, the order is earliest to latest, otherwise it is latest to earliest. (Type: `DateAndTime`).

Return Type: `EntityDirectoryURI`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.3.1.6 Operation: getServiceHistoryFor

Return a directory of `EntityDescriptions` matching the supplied namespace/name or URI whose changes became visible in the service between `fromDate` and `toDate`. The order of the dates controls the return order - if `fromDate` is later than `toDate` the return order is latest to earliest, otherwise it is earliest to latest.

Input Parameters

- **entityDescription** - the namespace/name or URI of the entityDescription being queried (Type: `EntityNameOrURI`).
- **fromDate**$_{OPT}$ - the earliest possible service `changeDate`. `fromDate` must be less than `changeDate`. If not supplied, earliest possible time is not checked. (Type: `DateAndTime`)
- **toDate**$_{OPT}$ - the latest possible service change date. `changeDate` must be less than or equal to `toDate`. If not supplied, `toDate` is not checked (Type: `DateAndTime`).

Return Type: `EntityList`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.4 Entity Description Maintenance Service

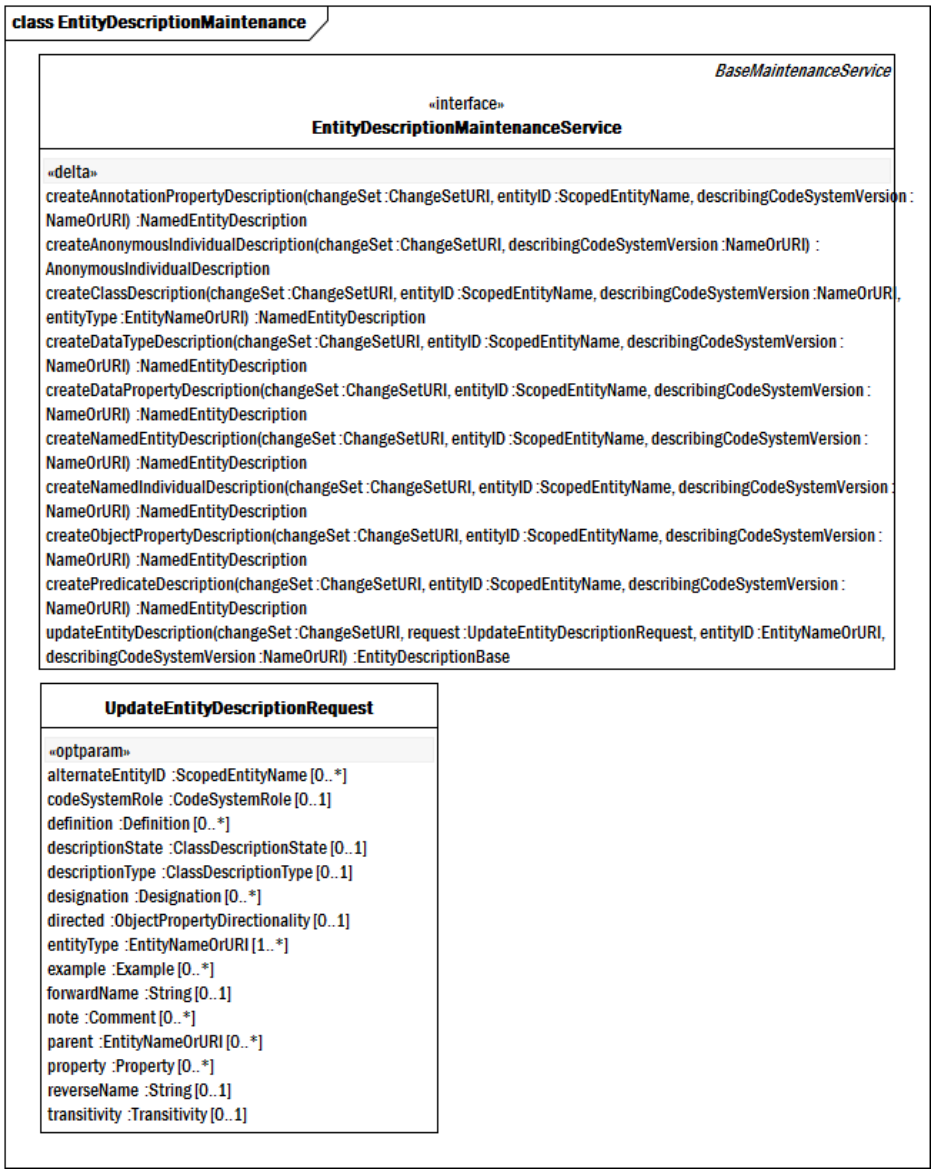

Figure 3.4 - Entity Description Maintenance Service

3.4.1 Interface EntityDescriptionMaintenanceService

Superclasses:

- Every instance of `EntityDescriptionMaintenanceService` is also an instance of `BaseMaintenanceService`.

3.4.1.1 Operation: createAnnotationPropertyDescription

Create a new `AnnotationPropertyDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type:`ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state (Type: `NameOrURI`).

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.2 Operation: createAnonymousIndividualDescription

Create a new `AnonymousIndividualDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).

- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state (Type: `NameOrURI`).

Return Type: `Anonymous IndividualDescription`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ResourceIsNotOpen** - The target resource version description has been finalized and cannot be updated.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.

3.4.1.3 Operation: createClassDescription

Create a new `ClassDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type:`ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)
- **entityType**$_{OPT}$ - the type (e.g., skos:concept, owl:class, rdf:property, owl:individual) of this entity. If not supplied the type will default to skos:concept. (Type: `EntityNameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.4 Operation: createDataTypeDescription

Create a new `DataTypeDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type: `ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.5 Operation: createDataPropertyDescription

Create a new `DataPropertyDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - The entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type:`ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.6 Operation: createNamedEntityDescription

Create a new `NamedEntityDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is server specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type: `ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.7 Operation: createNamedIndividualDescription

Create a new `NamedIndividualDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type: `ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.8 Operation: createObjectPropertyDescription

Create a new `ObjectPropertyDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change. (Type: `ChangeSetURI`)
- **entityID**$_{OPT}$ - the entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type:`ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.9 Operation: createPredicateDescription

Create a new `PredicateDescription`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change. (Type: `ChangeSetURI`)
- **entityID**$_{OPT}$ - The entity code and/or namespace identifier of the entity to be created. The decision about whether this parameter has to (or can be) supplied is service specific. Some services may choose to manage both namespaces and codes themselves, others may require a namespace without a code or a code without a namespace. (Type:`ScopedEntityName`)
- **describingCodeSystemVersion** - the URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `NamedEntityDescription`

Exceptions

- **EntityIDMustBeGeneratedByService** - The service expects to automatically generate the entityID for the supplied code system. It cannot be supplied in the API call.
- **EntityIDMustBeSupplied** - The service requires that the scoped entity name be provided.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not OPEN.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **EntityAlreadyExists** - A description of the named entity already exists in the code system version.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

3.4.1.10 Operation: updateEntityDescription

Update an existing entity description of any type.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - The identifier of an OPEN change set that will record the change. (Type: `ChangeSetURI`)
- **request** - The set of fields to be changed. (Type: `UpdateEntityDescriptionRequest`)
- **entityID** - The namespace and name or URI of the referenced entity. (Type: `EntityNameOrURI`)
- **describingCodeSystemVersion** - The URI or local identifier of the code system version that this entity description is both described in and described by. The referenced code system version must be in an `OPEN` (vs. `FINAL`) state. (Type:`NameOrURI`)

Return Type: `EntityDescriptionBase`

Exceptions

- **UnsupportedCaseSignificance** - The case Significance indicator is not recognized by the service.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedDesignationFidelity** - The `designationFidelity` is not recognized by the service.
- **UnsupportedDesignationType** - The `designationType` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ResourceIsNotOpen** - The target resource version description has been finalized and cannot be updated.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

3.4.2 Class UpdateEntityDescriptionRequest

The set of attributes that can be changed in an `EntityDescription`. The optparam stereotype means that, if not supplied in the `UpdateEntityRequest`, the named attribute will not be changed. If a parameter is supplied, it can add an attribute where none existed before (original cardinality = 0, update cardinality > 0), can remove an attribute that was previously present (original cardinality > 0, update cardinality = 0), or can update an attribute if both the original and update cardinality > 0. Note that lists must be replaced as units - to add an element to a list (e.g., to add a new note), the entire new list must be supplied. Finer grained update capability can be recognized via. the STATEMENT service implementation.

Attributes

- **alternateEntityID** - an identifier that uniquely references the `about` entity in the context of the describing code system version.
- **codeSystemRole** - If present, identifies the role that the code system plays in the description of the entity. If absent, the role is unknown.
- **definition** - an explanation of the intended meaning of a class or predicate.
- **descriptionState** - the specific role that a class description plays, if known.
- **descriptionType** - an indicator that states whether the class description contains only necessary assertions (`PRIMITIVE`) or both necessary and sufficient assertions (`FULLY_DEFINED`).
- **designation** - a label that represents the intended meaning in a natural language.
- **directed** - an indicator whether the predicate semantics are mono or bi-directional. This attribute only applies to entities of type `PREDICATE`.
- **entityType** - a type that the referenced entity is an instance of. All entity references must have at least one type. A service may add constraints about the allowed type transitions (e.g., from Class to Individual or Individual to Property).
- **example** - an example representing an instance or uses of the referenced entity.
- **forwardName** - a name assigned to a predicate when used in the source to target direction. Only applies to entities of type `PREDICATE`.
- **note** - a note or comment about the history, status, use, etc. of the referenced entity.
- **parent** - an asserted or direct “parent” of the entity.
- **property** - an additional “non-semantic” (annotation) assertion about the referenced entity.
- **reverseName** - a name assigned to a predicate when used in the target to source direction. Only applies to entities of type `PREDICATE`.
- **transitivity** - an indicator that states whether an object property is transitive. Only applies to entities of type `OBJECT_PROPERTY`.

3.5 Entity Description Transform Service

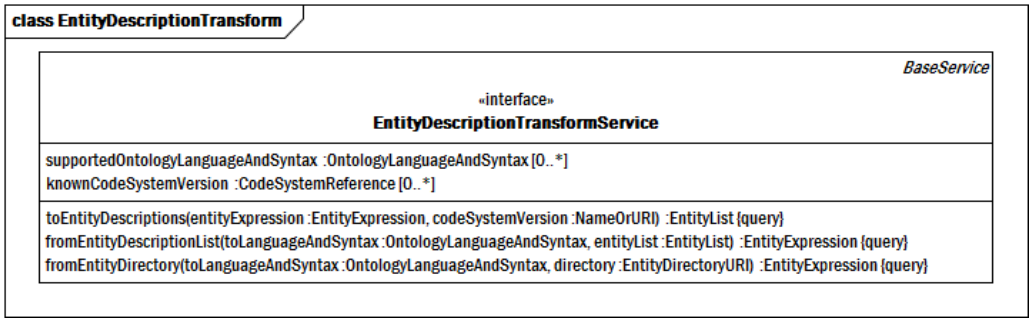

Figure 3.5 - Entity Description Transform Service

The `EntityDescriptionTransformService` transforms entity descriptions from external formats such as OWL/RDF, Manchester Syntax, SNOMED-CT Compositional Grammar, etc. into the corresponding `EntityDescriptions` without entering the descriptions into the service database. It also allows existing `EntityDescriptions` to be represented in external formats for viewing and portability.

3.5.1 Interface EntityDescriptionTransformService

A service that transforms various external formats into the CTS2 `EntityDescription` model and vice versa.

Superclasses

- Every instance of `EntityDescriptionTransformService` is also an instance of `BaseService`.

Attributes

- **supportedOntologyLanguageAndSyntax** - an ontology language and syntax that is supported by the transformation service.
- **knownCodeSystemVersion** - a URI and name of a code system version that can serve as the basis for a transformation by this service.

3.5.1.1 Operation: toEntityDescriptions

Transform `entityExpression` into a list of one or more `EntityDescriptions` in the context of the supplied `codeSystemVersion`.

Input Parameters

- **entityExpression** - the expression in the external language (Type: `EntityExpression`).
- **codeSystemVersion**$_{OPT}$ - the name or URI of the code system version that supplies the surrounding context for the transformation, if any (Type: `NameOrURI`).

Return Type: `EntityList`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **EntityExpressionSyntaxError** - The entity expression is syntactically incorrect and cannot be parsed.
- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.
- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.

3.5.1.2 Operation: fromEntityDescriptionList

Transform the list of entity descriptions in `entityList` into the semantics and syntax specified in `toLanguageAndSyntax`.

Input Parameters

- **toLanguageAndSyntax** - the target language and syntax for the transformation (Type:

`OntologyLanguageAndSyntax`).

- **entityList** - the list of entity descriptions to be transformed (Type: `EntityList`).

Return Type: `EntityExpression`

Exceptions

- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.
- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.

3.5.1.3 Operation: fromEntityDirectory

Transform the list of entity descriptions referenced in `entityDirectory` into the semantics and syntax specified in to `LanguageAndSyntax`.

Input Parameters

- **toLanguageAndSyntax** - the target language and syntax for the transformation (Type: `OntologyLanguageAndSyntax`).
- **directory** - a URI that resolves to the set of `EntityReferences` to be transformed (Type: `EntityDirectoryURI`).

Return Type: `EntityExpression`

Exceptions

- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.
- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

4 Association Information Model

4.1 Association Model

.

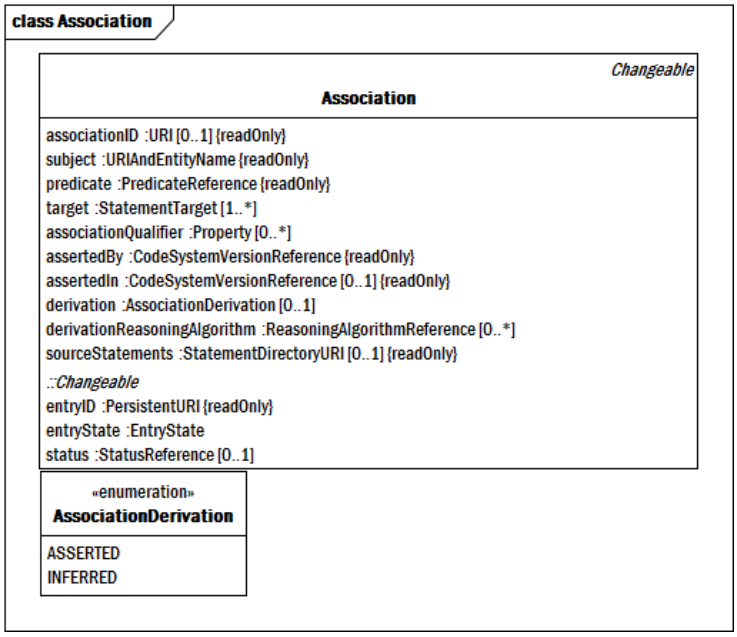

Figure 4.1 - Association Model

4.1.1 Class Association

A “semantic” assertion about a relationship between a `subject` entity and a target entity, literal or compound expression as represented by an unnamed blank node (“bnode”).

Superclasses

- Every instance of `Association` is also an instance of `Changeable`.

Attributes

- **associationID** - the unique identifier of the association instance. While the update mechanism of the CTS2 specifications requires that all `Changeable` elements have unique identifiers, the rules and mechanism for assigning unique identifiers to statements and associations are non-trivial. At the moment, the CTS2 specification remains silent when it comes to arriving at how this identifier is generated except to note that, if two service implementations do not generate the same unique identifier for the same assertion, then the `prevImage` element in `Changeable` becomes the only way to identify which Association instance is being updated or deleted.
- **subject** - the subject of the association statement. `subject` must reference an `Entity` - i.e., a class, predicate, and/or individual.
- **predicate** - the predicate of the association statement.

- **target** - the target (object) of the association statement. A target may be a reference to an entity, an external resource URI, a literal or a compound expression (bnode) or a set of targets.
- **associationQualifier** - additional tag/value attributes that qualify the association statement itself as opposed to the statement subject.
- **assertedBy** - the code system version that is making this assertion. A code system version makes an assertion when (a) it contains the assertion, (b) it imports a code system version that contains the assertion, or (c) the assertion is inferred from other assertions.
- **assertedIn** - the code system version that actually contains this assertion. If absent, it is the same resource as `assertedBy`.
- **derivation** - how this association was derived. `ASSERTED` takes precedence over `DERIVED`, meaning that an assertion that is both asserted and derived is marked as asserted. If absent, the derivation is unknown.
- **derivationReasoningAlgorithm** - a reasoning algorithm that was used to derive an association of `derivation INFERRED`. Note that more than one reasoning algorithm may be present for the same association.
- **sourceStatements** - detail about the provenance and history of the statement(s) from which this particular association is derived. This attribute will only be present in CTS2 service implementations that support the statements model.

Invariants

1. The unique identifier of an `Association` element is the `associationId`.

4.1.1.1 Enum AssociationDerivation

The derivation of a given statement - whether it is directly asserted by the source, is derived by the application of a reasoner or is unknown because the reasoner or source does not supply such information.

Enumeration Types:

- **ASSERTED** - The statement was explicitly asserted by the asserting code system version.
- **INFERRED** - The statement was not asserted by the code system version, but was logically inferred by the application of one or more reasoning algorithms to other statements within the code system version.

4.2 Association Graph

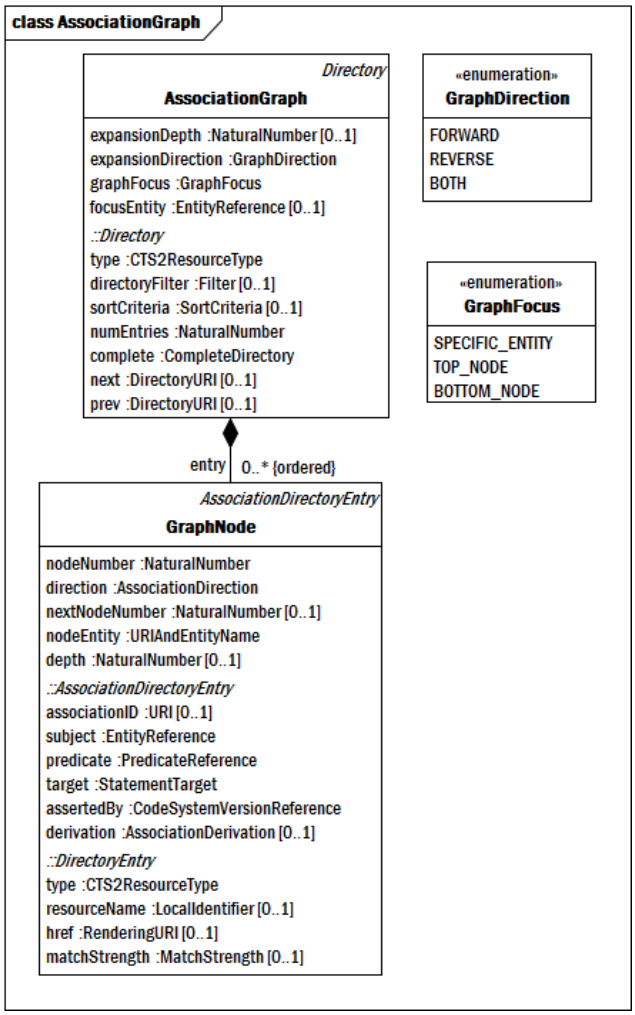

Figure 4.2 - Association Graph Model

`AssociationGraph` is an augmented rendering of a set of `Associations` where the linkage between subject and target nodes has been normalized for easier traversal. The challenge that is faced in the representation of graphs is that they can potentially be quite large and, as such, need to be able to be returned through the use of iterators, which are represented by the class `Directory` in the CTS2 model. This requirement, however, makes it very difficult to represent graphs hierarchically, as one graph may represent a single node with many thousands of direct descendents, another may represent a set of trees each 20 or more nodes deep and a third some other combination of these.

For this reason, the CTS2 model defines a flattened representation of the corresponding graph that is designed for manipulation and reassembly. To accomplish this, every `URIAndEntityName` that occurs in the set of associations is assigned a unique number. These numbers are assigned in depth first order and, when both forward and reverse association traversal is supported, in the order of forward first followed by reverse.

4.2.1 Class AssociationGraph

An augmented rendering of a set of `Associations` where the linkage between subject and target nodes has been normalized for easier traversal.

Superclasses

- Every instance of `AssociationGraph` is also an instance of `Directory`.

Attributes

- **expansionDepth** - the depth (one based) of the graph. This is a copy of the `resolveDepth` parameter of the `resolveAsGraph` method in the `AdvancedAssociationQueryService`.
- **expansionDirection** - the direction that the graph was expanded. This is a copy of the `resolveDirection` parameter from the `resolveAsGraph` method in the `AdvancedAssociationQueryService`.
- **graphFocus** - the focus of the graph.
- **focusEntity** - the focus entity of the graph, if any. This is a copy of the focus parameter from the `resolveAsGraph` method in the `AdvancedAssociationQueryService`.
- **entry** -

Invariants

1. The `graphFocus` can be on a `SPECIFIC_ENTITY` if and only if there is a `focusEntity` present.

4.2.1.1 Class GraphNode

The summary of an association as represented in a flattened graph structure.

Superclasses

- Every instance of `GraphNode` is also an instance of `AssociationDirectoryEntry`.

Attributes

- **nodeNumber** - an integer that corresponds to the particular `EntityReference` in `nodeEntity`. Every instance of the same `EntityReference` will be represented by the same `nodeNumber`. There will be graph node for every `source predicate target` combination in the association, with multiple `GraphNodes` being generated for `targets` that are of type set or BNode.
- **direction** - an indicator whether `nodeEntity` plays a source or a target role in the accompanying `AssociationDirectory` instance.
- **nextNodeNumber** - an integer that represents the next graph node that is reached by traversing the specified predicate in the specified direction. This number will only be present if the result of traversing predicate results in an `EntityReference`. Target nodes of type `ResourceReference` or Literal will not be assigned node numbers. Note that each `EntityReference` target of a nested BNODE will be represented by a different `GraphNode`.
- **nodeEntity** - the URI, name and a designation for the `EntityReference` represented by this node. `nodeEntity` represents the association subject when direction is `SOURCE_TO_TARGET` and (one of the) targets when direction is `TARGET_TO_SOURCE`.
- **depth** - The one based distance in hops between the nodeEntity and the graph focus.

Invariants

1. A graph node does not have a local identifier.

4.2.1.2 Enum GraphDirection

An indicator that states whether the graph represents a graph at which the focus is in the subject role, the target role, or both.

Enumeration Types:

- **FORWARD** - Indicates that graph is based the graph focus in the source role - the graph is drawn from source to target with the source at the root.
- **REVERSE** - Indicates that graph is based the graph focus in the target role - the graph is drawn from target source with the source at the bottom.
- **BOTH** - Indicates that graph is based the graph focus in both the source and the target role - the graph represents all associations in which the focus is either in the source or target role.

4.2.1.3 Enum GraphFocus

Identifies the root of an association graph.

Enumeration Types:

- **SPECIFIC_ENTITY** - The focus on the graph is contained in `focusEntity`.
- **TOP_NODE** - The focus of the graph is on the `TOP` or "universe" node.
- **BOTTOM_NODE** - The focus of the graph is on the `BOTTOM` or empty node.

4.3 Association Directories

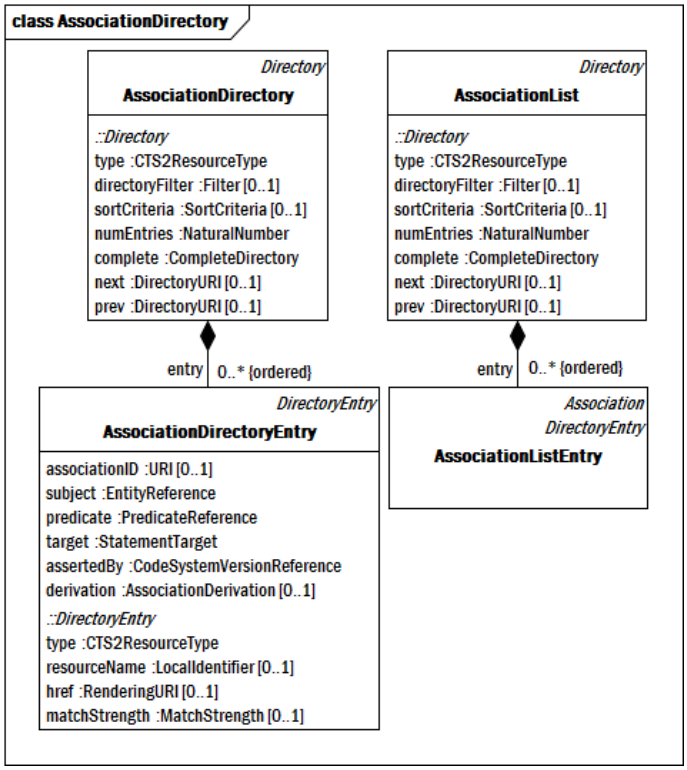

Figure 4.3 - Association Directory

4.3.1 Class AssociationDirectory

A directory of `Association` resources that meet a specified criteria.

Superclasses

- Every instance of `AssociationDirectory` is also an instance of `Directory`.

Attributes

- **entry** - an entry in an association directory

Invariants

1. The type of an `AssociationDirectory` is `ASSOCIATION`.

4.3.2 Class AssociationDirectoryEntry

A synopsis of an `Association` along with information about how to access the complete resource.

Superclasses

- Every instance of `AssociationDirectoryEntry` is also an instance of `DirectoryEntry`.

Attributes

- **associationID** - the unique identifier of the association instance. While the update mechanism of the CTS2 specifications requires that all Changeable elements have unique identifiers, the rules and mechanism for assigning unique identifiers to statements and associations are non-trivial. At the moment, the CTS2 specification remains silent when it comes to arriving at how this identifier is generated except to note that, if two service implementations do not generate the same unique identifier for the same assertion, then the prevImage element in Changeable becomes the only way to identify which Association instance is being updated or deleted.
- **subject** - the subject of the association statement.
- **predicate** - the predicate of the association statement.
- **target** - the target (object) of the association statement.
- **assertedBy** - the code system version that is making this assertion.
- **derivation** - how this association was derived. ASSERTED takes precedence over DERIVED, meaning that an assertion that is both asserted and derived is marked as asserted.

Invariants

1. The type of an `AssociationDirectoryEntry` is `ASSOCIATION`.
2. `AssociationDirectoryEntry` does not have a `resourceName`.

4.3.3 Class AssociationList

A collection of complete `Association` resources that meet a specified criteria.

Superclasses

- Every instance of `AssociationList` is also an instance of `Directory`.

Attributes

- entry -

Invariants

1. The type of an `AssociationList` is `ASSOCIATION`.

4.3.4 Class AssociationListEntry

An instance of `Association` that meets a specified filter criteria.

Superclasses

- Every instance of `AssociationListEntry` is also an instance of `Association`.
- Every instance of `AssociationListEntry` is also an instance of `DirectoryEntry`.

Invariants

1. The type of an `AssociationListEntry` is `ASSOCIATION`.

5 Association and Reasoning Services

5.1 Base Association Services

5.1.1 Association Read Service

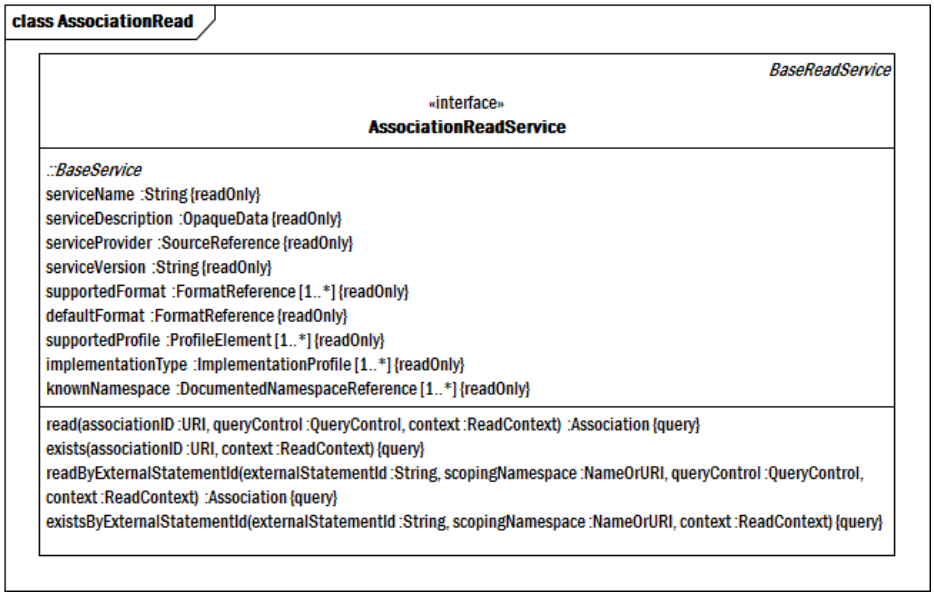

Figure 5.1 - Association Read Service

`AssociationReadService` provides direct access to association instances in a CTS2 service implementation. Instances may be retrieved by URI or by an external identifier associated with them in the context of a scoping namespace. Note that currently there is no widely accepted scheme for assigning "universal" URIs to association instances, meaning that the mechanism is left to the specific service implementation. This does mean, however, that associations pretty much have to be exchanged and addressed by value instead of URI.

In the case that Association instances have been given external identifiers, it is strongly recommended that the association URI be constructed from a combination of the namespace and external identifier. As an example, the International Health Standards Terminology Development Organization (IHTSDO)[13] assigns a unique "SctId" to every row in the relationship table. The combination of this external identifier and an appropriate namespace could be combined giving something in the form http://id.ihtsdo.org/2537148029. Note that the service also supports direct access to Associations by external identifier, meaning that the above example could be accessed by `scopingNamespace` "http://id.ihtsdo.org/"and an external identifier of 2537148029.

13. http://www.ihtsdo.org/

5.1.1.1 Interface AssociationReadService

A service that provides direct read access to associations.

Superclasses

- Every instance of `AssociationReadService` is also an instance of `BaseReadService`.

5.1.1.2 Operation: read

Retrieve the association instance identified by the `entryID` URI.

Input Parameters

- **associationID** - the URI that uniquely identifies the association instance (Type: `URI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, format, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced `Association` is not recognized by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

5.1.1.3 Operation: exists

Determine whether an association identified by the `entryID` URI is accessible from the service in the specified (optional) context.

Input Parameters

- **associationID** - the URI that uniquely identifies the association instance (Type: `URI`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `Boolean`

Exceptions

- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

5.1.1.4 Operation: readByExternalStatementId

Retrieve the association identified by the `identifierNamespace` and `externalStatementId`.

Input Parameters

- **externalStatementId** - the identifier assigned to the association instance in the context of the scoping namespace (Type: `String`).
- **scopingNamespace** - the name or URI of the namespace that contextualizes `externalStatementId` (Type: `NameOrURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, format, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

5.1.1.5 Operation: existsByExternalStatementId

Determine whether the association identified by the `identifierNamespace` and `externalStatementId` is accessible from the service in the specified (optional) context.

Input Parameters

- **externalStatementId** - the identifier assigned to the association instance in the context of the scoping namespace (Type: `String`).
- **scopingNamespace** - the name or URI of the namespace that contextualizes externalStatementId (Type: `NameOrURI`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: Boolean Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

5.1.2 Association Query Service

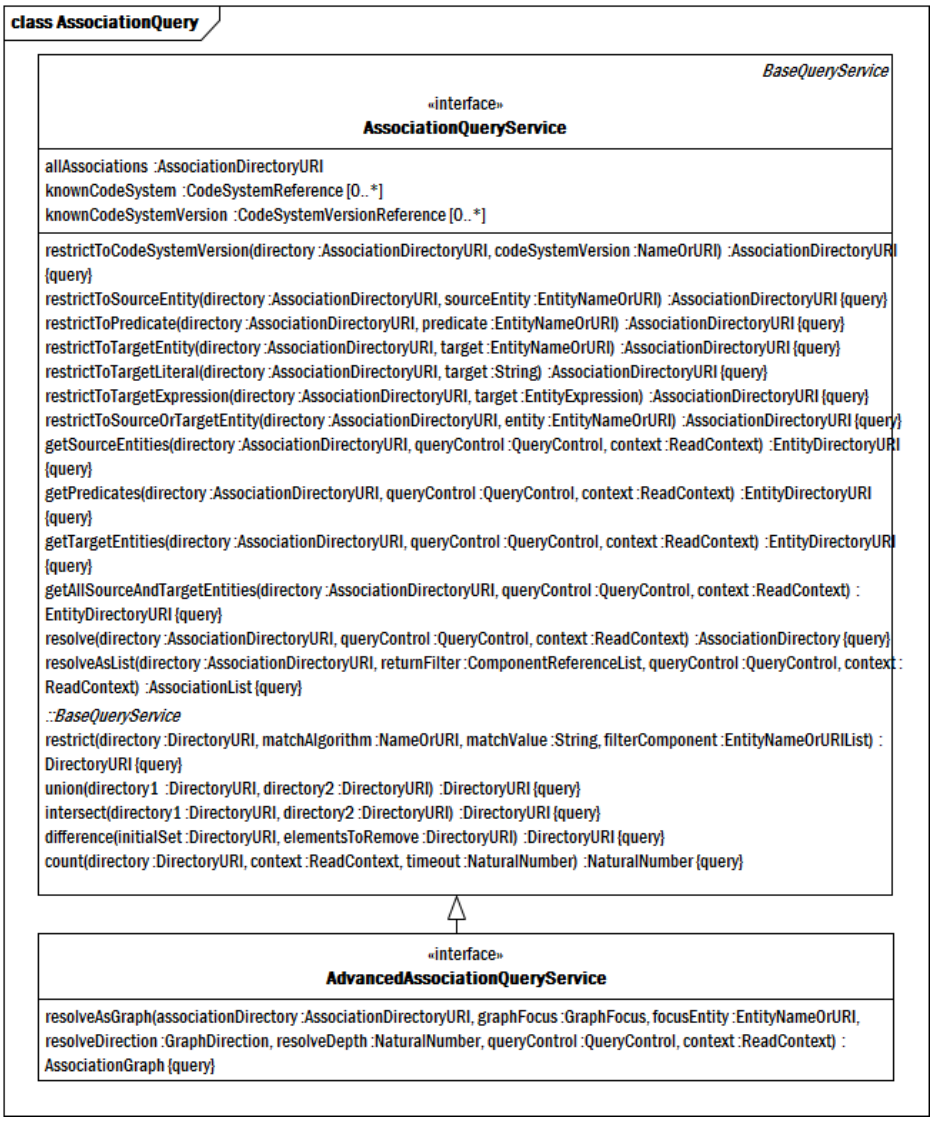

Figure 5.2 - Association Query Service

`AssociationQueryService` provides an ability that is in some ways similar to that provided by the SPARQL query language - it enables the creation of association subsets via various restriction criteria, the intersection, union, and difference of these sets and the ability to render the result of these queries in useful fashions. The primary difference between `AssociationQueryService` and SPARQL is that the `AssociationQueryService` works in the context of the Association structure defined by the CTS2 specification. This structure includes built-in provenance, direct representation of dependent BNODES and mechanisms for representing the output of the queries in ways that allow easy and direct RESTful manipulation.

5.1.2.1 Interface AdvancedAssociationQueryService

An optional extension to the `AssociationQueryService` that allows `AssociationDirectoryURIs` to be resolved as graphs.

Superclasses

- Every instance of Advanced `AssociationQueryService` is also an instance of `AssociationQueryService`.

5.1.2.2 Operation: resolveAsGraph

Resolve the set of associations referenced by `associationDirectory` as a graph.

Input Parameters

- **associationDirectory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **graphFocus**$_{OPT}$ - the focus (level 0) node of the graph. The focus node may either be the (potentially virtual) topmost node in the graph, the bottommost node in the graph or any valid node that is a member of the resolution of `associationDirectory`. (Type: `GraphFocus`)
- **focusEntity**$_{OPT}$ - the specific entity to focus on if `graphFocus` is SPECIFIC_ENTITY (Type: `EntityNameOrURI`).
- **resolveDirection** - the direction that the resolution occurs - "downwards" (in a forward direction) or "upwards" (in a reverse direction) or both (Type: `GraphDirection`).
- **resolveDepth**$_{OPT}$ - the number of levels to resolve. If omitted, the closure of the graph is resolved. (Type: `NaturalNumber`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, format, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `AssociationGraph`

Exceptions

- **UnknownFocusEntity** - either (a) the `focusEntity` is not reachable in association Directory or (b) `graphFocus` was not set to SPECIFIC_ENTITY but `focusEntity` was supplied.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

5.1.2.3 Interface AssociationQueryService

A service that allows the specification and rendering of useful subsets of associations. `AssociationQueryService` allows associations to be restricted by subject, predicate, target, asserting ontology, etc. It also provides set operations that allow the results of restrictions to be combined.

Superclasses

- Every instance of `AssociationQueryService` is also an instance of `BaseQueryService`.

Attributes

- **allAssociations** - A `DirectoryURI` that resolves to all Associations known to the service.
- **knownCodeSystem** - a code system that is known to this service.
- **knownCodeSystemVersion** - a code system version that is known to this service.

5.1.2.4 Operation: restrictToCodeSystemVersion

Return a `DirectoryURI` that represents the subset of the associations in directory that were `assertedBy` the named code system version.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **codeSystemVersion** - the name or URI of a code system version that constrains the output associations to those `assertedBy` this version (Type: `NameOrURI`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

5.1.2.5 Operation: restrictToSourceEntity

Return a `DirectoryURI` that represents the subset of the associations in directory whose subject is `sourceEntity`.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **sourceEntity** - the namespace/name or URI of the entity that must occur as the source of every association in the output set (Type: `EntityNameOrURI`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

5.1.2.6 Operation: restrictToPredicate

Return a `DirectoryURI` that represents the subset of the associations in directory whose predicate is `predicate`.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **predicate** - the namespace/name or URI of the entity that must occur as the predicate of every association in the output set (Type: `EntityNameOrURI`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

5.1.2.7 Operation: restrictToTargetEntity

Return a `DirectoryURI` that represents the subset of the associations in directory having a target directly matching target or containing one or more bnodes who reference target.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **target** - the namespace/name or URI of an entity that occurs either (a) in the target position or (b) as the target of a BNODE in the target position (Type: `EntityNameOrURI`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

5.1.2.8 Operation: restrictToTargetLiteral

Return a `DirectoryURI` that represents the subset of the associations in directory having a string literal that matches target.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **target** - the target literal to search for (Type: `String`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

5.1.2.9 Operation: restrictToTargetExpression

Return a `DirectoryURI` that represents the subset of the associations in directory whose target matches the supplied expression.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **target** - an expression in an external grammar that represents a `StatementTarget` (Type: `EntityExpression`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **EntityExpressionSyntaxError** - The entity expression is syntactically incorrect and cannot be parsed.
- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.
- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.

5.1.2.10 Operation: restrictToSourceOrTargetEntity

Return a `DirectoryURI` that represents the subset of the associations having either a source or a target matching the supplied entity. This is equivalent to the union of `restrictToSourceEntity` and `restrictToTargetEntity` with source and target being the same.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **entity** - the namespace/name or URI of the filter entity (Type: `EntityNameOrURI`).

Return Type: `AssociationDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnknownEntity** - The `EntityNameOrURI` is not known to the service.

5.1.2.11 Operation: getSourceEntities

Return a `DirectoryURI` that resolves to the set of entities that occur in the source position of the resolution of the supplied association directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

5.1.2.12 Operation: getPredicates

Return a `DirectoryURI` that resolves to the set of entities that occur in the predicate position of the resolution of the supplied association directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDirectoryURI`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.

- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.

5.1.2.13 Operation: getTargetEntities

Return a `DirectoryURI` that resolves to the set of entities that occur in the target position or as the target of a BNODE in the resolution of the supplied association directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDirectoryURI`

Exceptions

- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.

- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

5.1.2.14 Operation: getAllSourceAndTargetEntities

Return a `DirectoryURI` that resolves to the set of entities that occur in either the source or the target position of the resolution of the supplied association directory.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `EntityDirectoryURI`

Exceptions

- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

5.1.2.15 Operation: resolve

Return a directory that summarizes the resolution of the supplied association directory URI.

Input Parameters

- **directory** - a DirectoryURI that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `AssociationDirectory`

Exceptions

- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The `format` is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.

5.1.2.16 Operation: resolveAsList

Return the set of associations that are referenced by the supplied directory URI.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of associations (Type: `AssociationDirectoryURI`).
- **returnFilter**$_{OPT}$ - a list of zero or more component references. If present, the returned list entries will contain only the required elements and attributes plus any elements or attributes named in the filter. If absent all elements and attributes will be returned. (Type: `ComponentReferenceList`).

- **queryControl**$_{OPT}$ - parameters that control the ordering, timeout, and query behavior (Type: `QueryControl`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).

Return Type: `AssociationList`

Exceptions

- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.
- **InvalidDirectoryURI** - The supplied directory URI is not valid.
- **InvalidDirectoryType** - The type represented by the supplied directory URI is not the one required by the service invocation.

5.1.3 Association History Service

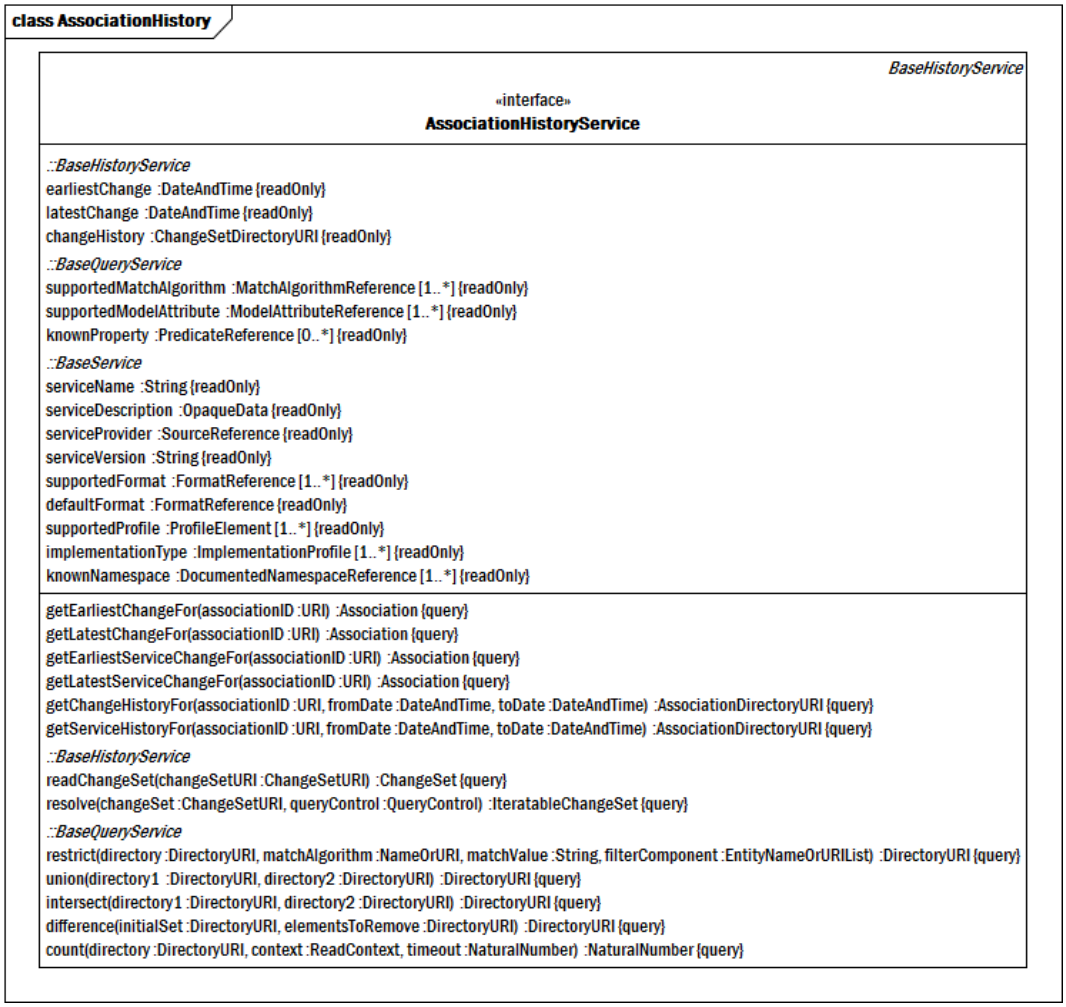

Figure 5.3 - Association History

The `AssociationHistoryService` allows the client to find the first change, last change, and the set of changes that occurred to an Association in a particular range of time. Changes can be viewed either from the perspective of the service - determining when a change became visible to the service clients or from the perspective of the publisher - determining when the author of the change stated that it was to become active.

5.1.3.1 Interface AssociationHistoryService

A service that provides a historical perspective on what changes occurred to a given Association.

- Every instance of `AssociationHistoryService` is also an instance of `BaseHistoryService`.

5.1.3.2 Operation: getEarliestChangeFor

Select the Association instance in the `ChangeSet` that has the earliest `officialEffectiveDate` and return the first change for the referenced Association in the set. The return value represents the earliest known published state of the Association.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.

5.1.3.3 Operation: getLatestChangeFor

Select the Association instance in the `ChangeSet` that has the latest `officialEffectiveDate` and return the last change for the referenced Association in the set. The return value represents the most recent known published state of the Association.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.

5.1.3.4 Operation: getEarliestServiceChangeFor

Get the first change that was applied to the named entity from the perspective of the service instance.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.

5.1.3.5 Operation: getLatestServiceChangeFor

Get the last change that was applied to the named entity from the perspective of the service instance.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).

Return Type: `Association`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.

5.1.3.6 Operation: getChangeHistoryFor

Return a directory of `Associations` with the supplied URI whose `ChangeSet officialEffectiveDate` falls between `fromDate` and `toDate`. The order of the dates controls the return order - if `fromDate` is later than `toDate` the return order is latest to earliest, otherwise it is earliest to latest.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).
- **fromDate**$_{OPT}$ - the starting date. If later than `toDate`, the return order is latest to earliest, otherwise it is earliest to latest. If not supplied, it is treated as the earliest possible date. (Type: `DateAndTime`)
- **toDate**$_{OPT}$ - the end date. If later than `fromDate`, the return order is earliest to latest otherwise it is latest to earliest. (Type: `DateAndTime`)

Return Type: `AssociationDirectoryURI`

Exceptions

- **UnknownAssociation** - The referenced Association is not recognized by the service.

5.1.3.7 Operation: getServiceHistoryFor

Return a directory of `Associations` with the supplied URI whose changes became visible in the service between `fromDate` and `toDate`. The order of the dates controls the return order - if `fromDate` is later than `toDate` the return order is latest to earliest, otherwise it is earliest to latest.

Input Parameters

- **associationID** - an association identifier (Type: `URI`).
- **fromDate**$_{OPT}$ - the starting date. If later than `toDate`, the return order is latest to earliest, otherwise it is earliest to latest. If not supplied, it is treated as the earliest possible date. (Type: `DateAndTime`)
- **toDate**$_{OPT}$ - the end date. If later than `fromDate`, the return order is earliest to latest otherwise it is latest to earliest. (Type: `DateAndTime`)

Return Type: `AssociationDirectoryURI`

Exceptions

- **UnknownAssociation** - The referenced `Association` is not recognized by the service.

5.1.4 Association Maintenance Service

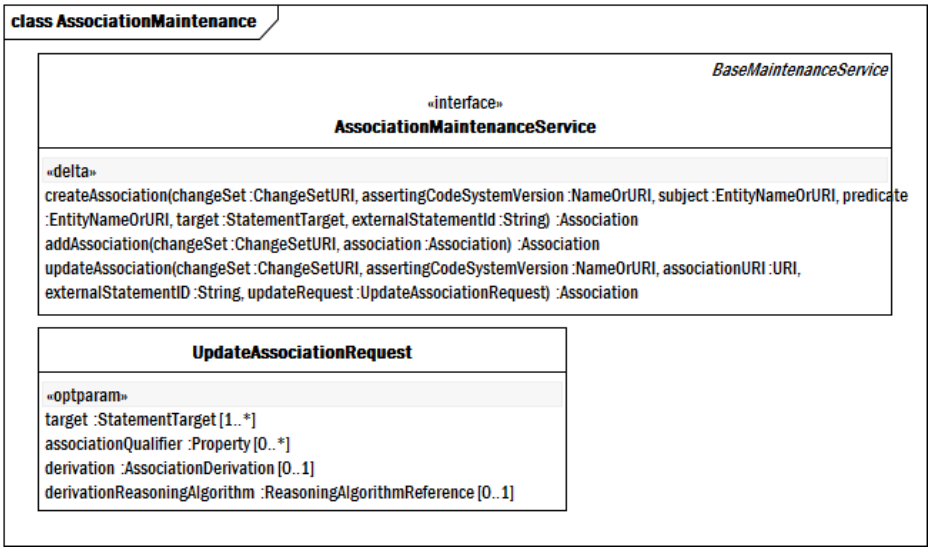

Figure 5.4 - Association Maintenance

The `AssociationMaintenanceService` provides the ability to create, update, or remove associations from the set of assertions.

5.1.4.1 Interface AssociationMaintenanceService

A service that allows the creation and modification of semantic assertions made by an ontology or code system.

Superclasses

- Every instance of `AssociationMaintenanceService` is also an instance of `BaseMaintenanceService`.

5.1.4.2 Operation: createAssociation

Create a new `Association`.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change (Type: `ChangeSetURI`).
- **assertingCodeSystemVersion** - the local identifier or URI of the code system version that is making this assertion. If the service supports the `CodeSystemVersion` authoring profile, the version must exist and be `OPEN` (not `FINAL`) (Type:`NameOrURI`).
- **subject** - the namespace/name or URI of the subject of the association (Type: `EntityNameOrURI`).
- **predicate** - the namespace/name or URI of the association predicate (Type: `EntityNameOrURI`).

- **target** - the statement target (Type: `StatementTarget`).
- **externalStatementId**$_{OPT}$ - An optional statement identifier that is assigned to this statement from an outside organization (Type: `String`).

Return Type: `Association`

Exceptions

- **UnknownAssociationQualifierNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **UnknownTargetEntity** - The target `EntityNameOrURI` is not known to the service.
- **UnsupportedAssociationQualifier** - The predicate of an association qualifier is not recognized by the service or is not considered to be a valid association qualifier.
- **UnknownPredicateNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnsupportedSubjectNamespaceName** - The namespace in the subject identifier is not known to or supported by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownSubjectEntity** - The `EntityNameOrURI` is not known to the service.
- **UnknownTargetNamespaceName** - The namespace in a target entity identifier is not known to or supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.

5.1.4.3 Operation: addAssociation

Add the input association to the service.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - the identifier of an OPEN change set that will record the change. (Type: `ChangeSetURI`)
- **association** - the new association to be added to the service. (Type: `Association`)

Return Type: `Association`

Exceptions

- **UnknownAssociationQualifierNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **UnknownTargetEntity** - The target `EntityNameOrURI` is not known to the service.

- **UnsupportedAssociationQualifier** - The predicate of an association qualifier is not recognized by the service or is not considered to be a valid association qualifier.
- **UnknownPredicateNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnsupportedSubjectNamespaceName** - The namespace in the subject identifier is not known to or supported by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnknownSubjectEntity** - The `EntityNameOrURI` is not known to the service.
- **UnknownTargetNamespaceName** - The namespace in a target entity identifier is not known to or supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.

5.1.4.4 Operation: updateAssociation

Update an existing Association resource.

This operation may alter the state of the service.

Input Parameters

- **changeSet** - (Type: `ChangeSetURI`)
- **assertingCodeSystemVersion** - (Type: `NameOrURI`)
- **associationURI**$_{OPT}$ - (Type: `URI`)
- **externalStatementID**$_{OPT}$ - (Type: `String`)
- **updateRequest** - (Type: `UpdateAssociationRequest`)

Return Type: `Association`

Exceptions

- **UnknownAssociationQualifierNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **UnknownTargetEntity** - The target `EntityNameOrURI` is not known to the service.
- **UnsupportedAssociationQualifier** - The predicate of an association qualifier is not recognized by the service or is not considered to be a valid association qualifier.
- **UnknownPredicateNamespaceName** - The namespace in the predicate identifier is not known to or supported by the service.
- **UnsupportedSubjectNamespaceName** - The namespace in the subject identifier is not known to or supported by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.

- **UnknownSubjectEntity** - The `EntityNameOrURI` is not known to the service.
- **UnknownTargetNamespaceName** - The namespace in a target entity identifier is not known to or supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.

5.1.4.5 Class UpdateAssociationRequest

The set of elements that can be changed in an existing association. Association identity has always been an interesting issue, and the CTS2 specification has chosen to say that, while identity is undoubtedly more than just the subject and the predicated, we can't really draw a clean line in the target / qualifier range. The basic CTS2 specification allows updates on any part of the target or any qualifier, although we anticipate that service implementations may choose more strict interpretations.

Attributes

- **target** - the new or modified association target.
- **associationQualifier** - a qualifier whose subject is that association itself.
- **derivation** - the new or modified derivation.
- **derivationReasoningAlgorithm** - the new or modified derivation reasoning algorithm.

5.1.5 Association Transform Service

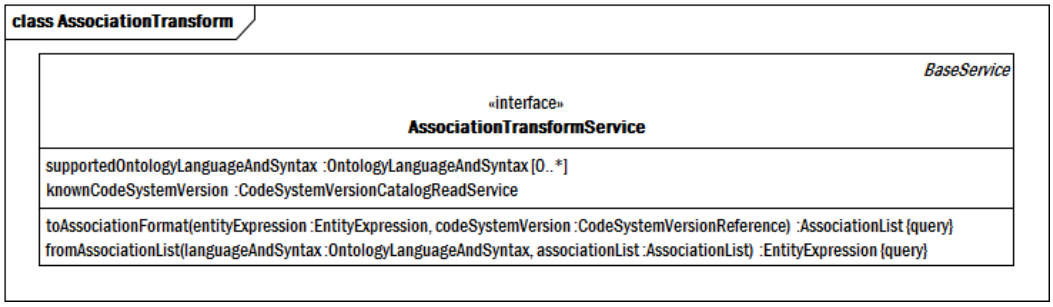

Figure 5.5 - Association Transform

The `AssociationTransformService` transforms association statements from external formats such as OWL/RDF, Manchester Syntax, SNOMED-CT Compositional Grammar, etc. into the corresponding `Associations` without entering the descriptions into the server database. It also allows existing `Associations` to be represented in external formats for viewing and portability.

5.1.5.1 Interface AssociationTransformService

A service that transforms various external formats into the CTS2 `Association` model and visa versa.

Superclasses

- Every instance of `AssociationTransformService` is also an instance of `BaseService`.

Attributes

- **supportedOntologyLanguageAndSyntax** - an ontology language / syntax combination that is supported by this transformation service.
- **knownCodeSystemVersion** - the URI and name of a code system version that can serve as the basis for a tranformation by this service.

5.1.5.2 Operation: toAssociationFormat

Transform `entityExpression` into a lot of one or more `Associations` in the context of the supplied code `SystemVersion`.

Input Parameters

- **entityExpression** - the expression of a set of associations in a named ontology language and syntax. (Type: `EntityExpression`)
- **codeSystemVersion** - the name and URI of the code system version that supplies for the surrounding context for the transformation. (Type: `CodeSystemVersionReference`)

Return Type: `AssociationList`

Exceptions

- **EntityExpressionSyntaxError** - The entity expression is syntactically incorrect and cannot be parsed.
- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.
- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.

5.1.5.3 Operation: fromAssociationList

Return an expression that represents `associationList` in the named `languageAndSyntax`.

Input Parameters

- **languageAndSyntax** - the target ontology language and syntax (Type: `OntologyLanguageAndSyntax`).
- **associationList** - a list of one or more CTS2 format associations (Type: `AssociationList`).

Return Type: `EntityExpression`

Exceptions

- **UnsupportedOntologySyntax** - The supplied ontology syntax is not supported by the service.
- **UnsupportedOntologyLanguage** - The supplied ontology language is not supported by the service.

5.2 Reasoning Service

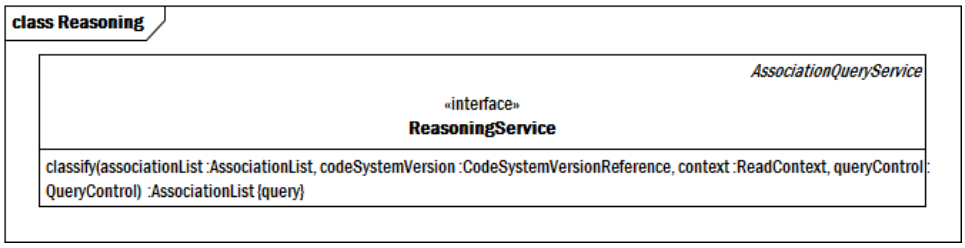

Figure 5.6 - Reasoning Service

The `ReasoningService` allows the posing of "what if" questions. Whenever a new `Association` (i.e., semantic assertion) is written into the service database, a service implementation that supports inference and reasoning would be expected to automatically derive the logical consequences of the change. As an example, if `ChangeSet` is imported into a service that asserts that Class X is a subclass of Class Y, and Class Y has an assertion that Y hasPart some H, Class X, when retrieved from the service will include an assertion that Class X hasPart some H, with an indicator that this was derived through the application of a reasoning algorithm.

One approach to doing "what if" reasoning would be to take advantage of the UPDATE and AUTHORING services. One could create a new change set, add whatever "what if" inferences they wanted tested, analyze the result in the context of the change set and then roll it back. This is, effectively, what many of the current ontology authoring tool kits do - you import an ontology source file, create new assertions, classify those assertions and evaluate the results. Once complete, the new assertions are then discarded.

This is less than an ideal approach, however, for a CTS2 compliant service as requiring services that provide a reasoning interface should not be required to support the authoring interface. Furthermore, the authoring architecture was intended for submissions that had a high likelihood of becoming accepted, with the roll back function intended to be the exception rather than the rule. The `ReasoningService`, as described below, is intended to be a "lighter weight" alternative.

The classify interface described above takes a set of assertions, combines it with the set of active assertions that are `assertedBy` the named `codeSystemVersion` and invokes the reasoner(s) that are considered appropriate by the service implementation. It then returns the resulting set of assertions that result from the combination of the input and the application of the reasoner that are not already asserted by the referenced code system version.

5.2.1 Interface ReasoningService

An extension to the `AssociationQueryService` that allows the posing of "what if" questions.

Superclasses

- Every instance of `ReasoningService` is also an instance of `AssociationQueryService`.

5.2.1.1 Operation: classify

The set of new assertions that can be inferred from the combination of the assertions in `associationList` and the set that are `assertedBy codeSystemVersion`.

Input Parameters

- **associationList** - the set of assertions about one or more subjects to be classified (Type: `AssociationList`).
- **codeSystemVersion** - the name or URI of a code system version that is used to establish the base set of axioms against which the reasoning will be performed. All active associations that are `assertedBy` the referenced `codeSystemVersion` will be considered to be in the base set (Type: `CodeSystemVersionReference`).
- **context**$_{OPT}$ - parameters that control the language, date and time, and other contextual variables (Type: `ReadContext`).
- **queryControl**$_{OPT}$ - parameters that control the ordering, format, timeout, and query behavior (Type: `QueryControl`).

Return Type: `AssociationList`

Exceptions

- **UnknownCodeSystemVersion** - The referenced `CodeSystemVersion` is not recognized by the service.
- **InferenceNotSupported** - Inference is not supported for the supplied `codeSystemVersion`.
- **UnsupportedMatchAlgorithm** - The `matchAlgorithm` is not supported by the service.
- **UnsupportedPredicate** - The predicate name or URI is not recognized by the service.
- **UnsupportedNamespaceName** - The supplied namespace name is not one that is known to the service.
- **UnsupportedModelAttribute** - The name or URI of a CTS2 model attribute is not recognized and/or supported by the service implementation.
- **MatchValueFormatException** - The match value does not meet the format or syntax required or the supplied match algorithm.
- **UnsupportedFormat** - The format is not supported by the service implementation.
- **QueryTimeout** - The `timeLimit` was exceeded by the service.
- **UnsupportedLanguage** - The `referenceLanguage` is not supported by the service.
- **UnknownChangeSet** - The change set specified could either not be read or located by the service.
- **ChangeSetIsNotOpen** - The `changeSetContext` is recognized by the service, but its state is not `OPEN`.
- **UnsupportedContext** - One or more `changeSetContext` is not supported by the service.