Date: June 2013

CTS2 Statement Model and Services

Version 1.1

OMG Document Number: formal/2013-05-12
Standard document URL: http://www.omg.org/spec/cts2/1.1/PDF

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

The CTS2 specification is divided into a number of separably implementable profiles. This section on Statement Services specifies a model and interface that describes the relationship between the formal CTS2 model components and the 'native' structure from which the model was derived.

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 Statement Information Model

2.1 Statement Model

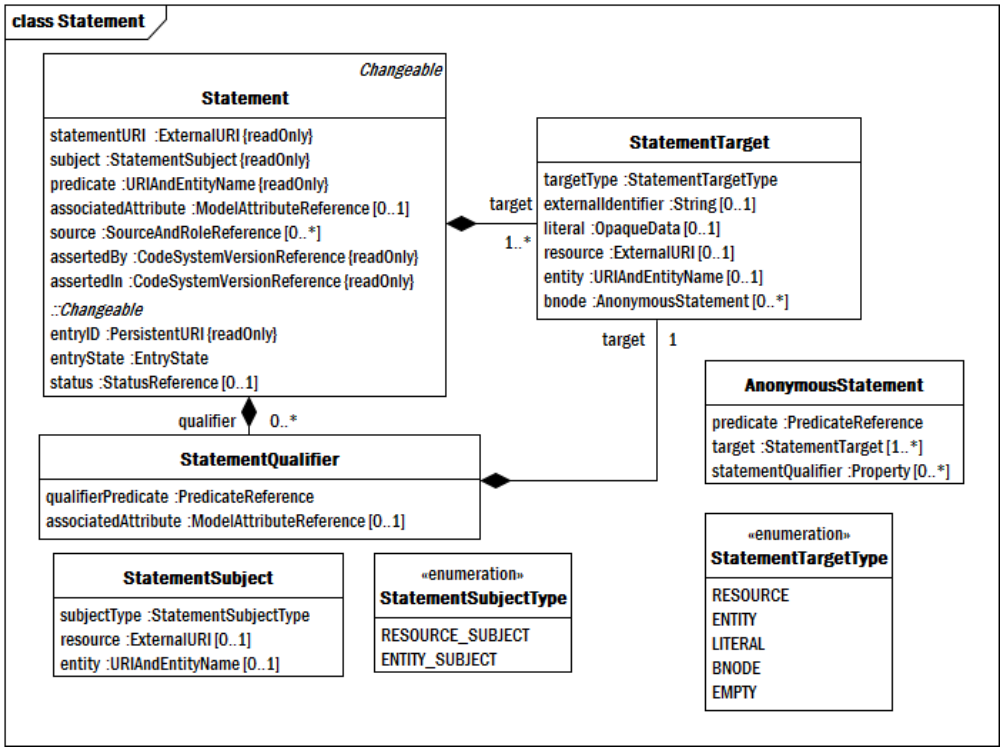

Figure 2.1 - Statement Model

The Statement model acts as a bridge between the structured model that is defined by the CTS2 for use in SOAP and REST style transactions and the minimal subject, predicate, object model used in RDF. Pure RDF provides a great deal of flexibility - flexibility that, from the CTS2 perspective has a number of undesired side effects.

CTS2 is, in no small part, about provenance - we need to know the source of a given statement, what ontology is asserting it, what ontology contained the statement, whether it is currently active, etc. While all of this is possible in RDF, it currently requires the use of reified statements.

One of the issues with the reified equivalent is that there *is* no model that asserts that all statements must have a subject, predicate, and object. The second issue involves the identity of the statement itself.

A second issue encountered by CTS2 is the set model used in RDF. (document) While the LISP model is elegant, it is not the sort of representational form that the terminology community is interested in seeing. The CTS2 statement has abstracted this detail away by declaring target as a SET rather than a single entity.

The third difference between CTS2 and traditional RDF is that the CTS2 model makes a distinction between external resources, which can be most anything, and entities within the terminological space, which we wish to represent as namespace/name tuples.

A final distinction between CTS2 and the traditional RDF model is that CTS2 does not reify bnodes. A bnode has no identity outside of the context of its surrounding statement and as a consequence cannot be revised or deleted unless accompanied by their encompassing context. For this reason, we do not reify bnodes but rather keep them in the context of their containing statement.

2.1.1 Class Statement

An assertion about a CTS2 model element.

Superclasses

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

Attributes

- **statementURI** - the unique statement identifier. Must be globally unique if information is to be exchanged and updated on the statement level
- **subject** - the subject of the statement - either a resource or a reference to a terminological entity. subject may reference a specific `Association`, `CodeSystem`, `CodeSystemVersion`, `Map`, `MapEntry`, `MapVersion`, `EntityDescription`, `ValueSet`, `ValueSetDefinition`, `ConceptDomain`, `ConceptDomainBinding`.
- **predicate** - the predicate of the statement. Note that this is type `URIAndEntityName` as, while it serves as a predicate in the statement itself it is not necessary that it be formally recognized as a predicate by the CTS2 service - it may have been translated or transformed by the load process.
- **associatedAttribute** - the identifier and name of the CTS2 attribute or element that renders this statement, if any.
- **source** - the source(s) of the statement and the role(s) they played.
- **assertedBy** - the code system version that is asserting this particular statement. In the case of ontologies that import other ontologies, this is the outermost importing ontology.
- **assertedIn** - the code system version (ontology) that actually makes this statement. In the case of ontologies, this is the innermost imported ontology.
- **target** - The object or target of the statement. If the cardinality is one, target may either represent a single element or a set containing one member. It is assumed that the distinction between these two cases will be obvious based on the usage context. Note, however, that a cardinality of greater than one *always* represents a set.

  At the moment target cannot represent sequences or bags - only sets.
- **qualifier** - Metadata about the statement itself, vs. the statement subject.

2.1.2 Class StatementSubject

The subject of a statement, which may either be a resource URI or an entity reference, which includes both the URI and the namespace/name.

Attributes

- **subjectType** - A discriminant that determines the statement subject type. PSM's that support string typing systems may implement this through the built-in-type system rather than providing a specific element.

- resource-Anon-entity statement subject such as the URI of a `CodeSystem`, `CodeSystemVersion`,`ValueSet`, etc.
- entity - The subject of a statement about a class, role, or individual.

Invariants

1. A statement subject must contain only one of resource or entity.

2.1.3 Enum StatementSubjectType

A discriminant that determines the type of `StatementSubject`.

Enumeration Types:

- **RESOURCE_SUBJECT** - the subject of the statement is a resource.
- **ENTITY_SUBJECT** - the subject of a statement is an entity.

2.2 Statement List and Directory

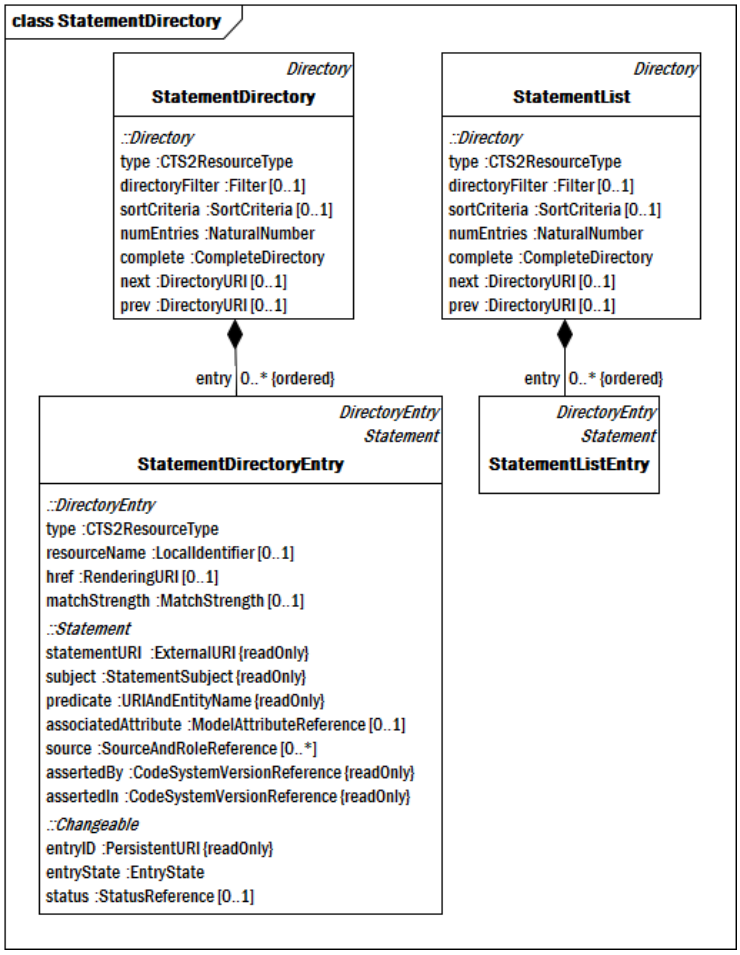

Figure 2.2 - Statement Directory

`StatementDirectories` and `StatementLists` have identical structures. The reason that the specification identifies both types is to maintain consistency with the rest of the specification.

2.2.1 Class StatementDirectory

A directory of Statement resources that meet a specified criteria.

Superclasses

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

Attributes

- **entry** - an entry in a statement directory.

2.2.2 Class StatementDirectoryEntry

A synopsis of a Statement along with information about how to access the complete resource.

Superclasses

- Every instance of `StatementDirectoryEntry` is also an instance of `Statement`.
- Every instance of `StatementDirectoryEntry` is also an instance of `DirectoryEntry`.

2.2.3 Class StatementList

A collection of complete Statement resources that meet a specified criteria.

Superclasses

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

Attributes

- **entry** - an entry in a `StatementList`

2.2.4 Class StatementListEntry

An instance of Statement that meets a specified filter criteria.

Superclasses

- Every instance of `StatementListEntry` is also an instance of `Statement`.
- Every instance of `StatementListEntry` is also an instance of `DirectoryEntry`.

3 Statement Services

3.1 Statement Read Service

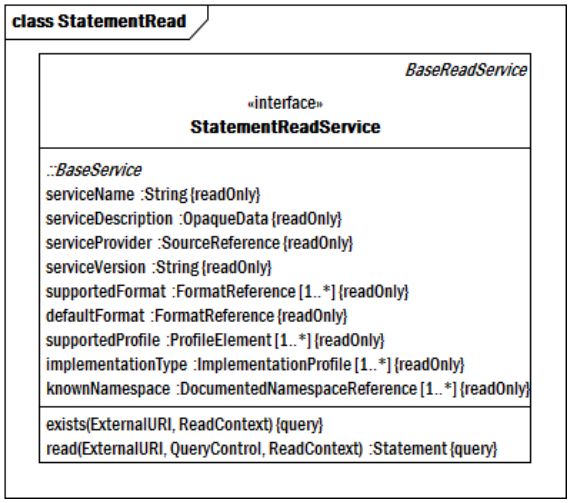

Figure 3.1 - Statement Read Service

3.1.1 Interface StatementReadService

`StatementReadService` provides direct access to statements - assertions about CTS2 resources, entities, associations, concept domain bindings, and map entries.

Superclasses

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

3.1.1.1 Operation: exists

Determine whether a statement identified by the supplied identifier is known to the service in the supplied context.

Input Parameters

- **StatementID** - The identifier of the statement in question (Type: `ExternalURI`).
- **context**$_{OPT}$ - parameters that control the language, date, 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.

3.1.1.2 Operation: read

Return the rendering of the supplied statement as it appears in the supplied context.

Input Parameters

- **statementID** - the identifier of a Statement (Type: `ExternalURI`).
- **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: `Statement`

Exceptions

- **UnknownStatement** - The referenced statement is not recognized 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.
- **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 Statement Query Service

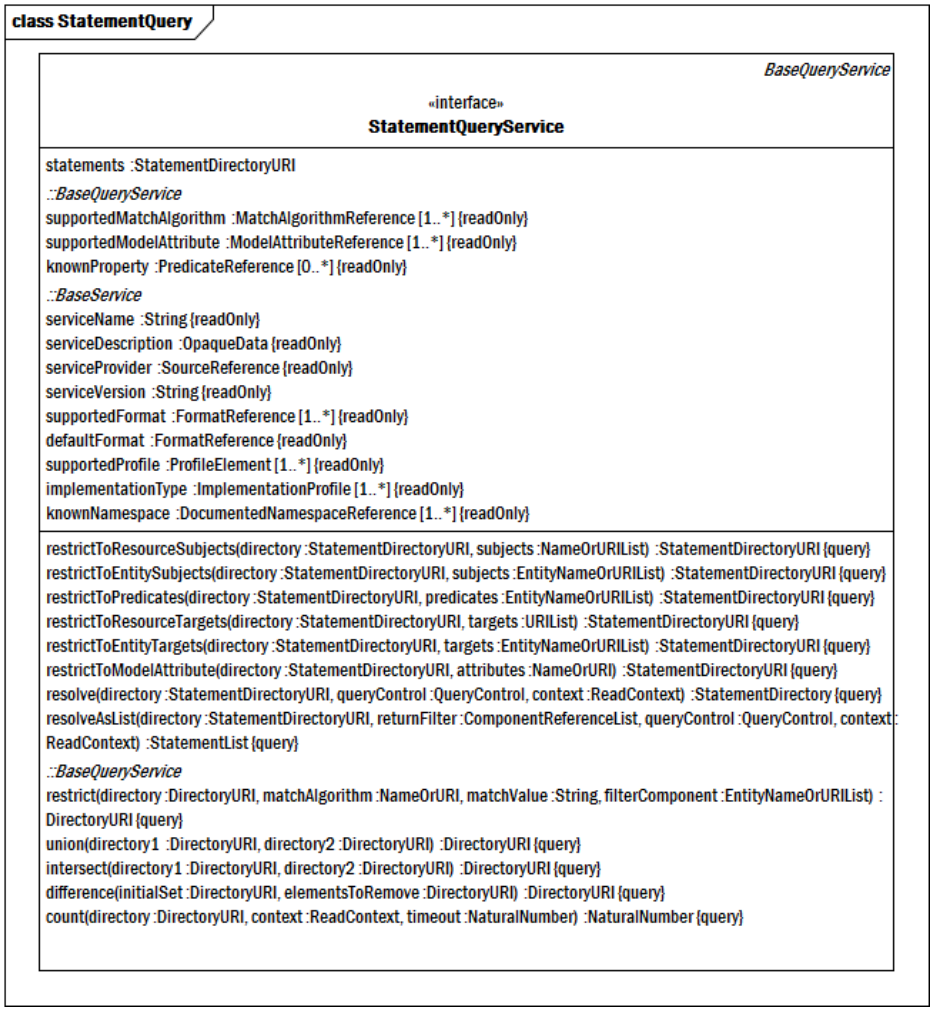

Figure 3.2 - Statement Query Service

3.2.1 Interface StatementQueryService

`StatementQueryService` provides an ability to filter directories of statements by subject, predicate, target and/or model element. It should be noted that the functionality provided here has significant overlap with specifications such as SPARQL and the API4KB. It is not the intent of the CTS2 specification to compete with or offer alternatives to these specifications. The key differences between this API and those mentioned above is the `Statement` model. First, it differentiates between simple resource URIs and the namespace/name structure in entity references. Secondly it provides bridge information that identifies how CTS2 structures were derived from more atomic information. Finally, and perhaps most importantly, it provides provenance and history for the sets of assertions in statements. The query service itself could (and probably should) be implemented using triple stores, SPARQL, or other alternatives.

Superclasses

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

Attributes

- **statements** - a `DirectoryURI` that resolves to the set of statements known to the service.

3.2.1.1 Operation: restrictToResourceSubjects

Return a `DirectoryURI` that resolves to the set of statements in directory that have subjects in subjects.

Input Parameters

- **directory** - a `DirectoryURI` that references a set of `Statements` (Type: `StatementDirectoryURI`).
- **subjects** - the resource subjects of the statement. Note that this specification constrains these to resources defined in the CTS2 specification (Type: `NameOrURIList`).

Return Type: `StatementDirectoryURI`

Exceptions

- **UnknownResourceReference** - A CTS2 resource was referenced by name, URI or external identifier that was not known to the CTS2 service instance.
- **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: restrictToEntitySubjects

Return a `DirectoryURI` that resolves to the set of statements in directory that have subjects in subjects.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of statements (Type: `StatementDirectoryURI`).
- **subjects** - a set of namespace/names or URIs that are used to filter the statement subject (Type: `EntityNameOrURIList`).

Return Type: `StatementDirectoryURI`

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.

3.2.1.3 Operation: restrictToPredicates

Return a `DirectoryURI` that resolves to the set of statements in directory that have subjects in subjects.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of statements (Type: `StatementDirectoryURI`).
- **predicates** - a set of namespace/names or URIs that are used to filter the statement predicate (Type: `EntityNameOrURIList`).

Return Type: `StatementDirectoryURI`

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: restrictToResourceTargets

Return a `DirectoryURI` that resolves to the set of statements in directory that have targets in targets. This includes both direct and any targets in BNODES.

Input Parameters

- **directory** - a `DirectoryURI` that references a set of `Statements` (Type: `StatementDirectoryURI`).
- **targets** - the resource targets of the statement. (Type: `URIList`).

Return Type: `StatementDirectoryURI`

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.

3.2.1.5 Operation: restrictToEntityTargets

Return a `DirectoryURI` that resolves to the set of statements in directory that have targets in targets. This includes both direct and any targets in BNODES.

Input Parameters

- **directory** - a `DirectoryURI` that references a set of `Statements`. (Type: `StatementDirectoryURI`)
- **targets** - the resource targets of the statement. This includes both direct entity targets and entity references that appear in bnodes (or bnodes in bnodes). (Type: `EntityNameOrURIList`)

Return Type: `StatementDirectoryURI`

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.

3.2.1.6 Operation: restrictToModelAttribute

Restrict the list of statement to those in the supplied list of model attributes.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `Statements` (Type: `StatementDirectoryURI`).
- **attributes** - a list of constraining model attributes (Type: `NameOrURI`).

Return Type: `StatementDirectoryURI`

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.

3.2.1.7 Operation: resolve

Resolve a `StatementDirectoryURI` and return a directory summarizing the statements it references.

Input Parameters

- **directory** - a DirectoryURI that resolves to a set of Statements (Type: `StatementDirectoryURI`).
- **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: `StatementDirectory`

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.

3.2.1.8 Operation: resolveAsList

Resolve a `StatementDirectoryURI` and return a list of the statements it references.

Input Parameters

- **directory** - a `DirectoryURI` that resolves to a set of `Statements` (Type: `StatementDirectoryURI`).
- **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 other contextual variables (Type: `ReadContext`).

Return Type: `StatementList`

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.
- **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.

3.3 Statement History Service

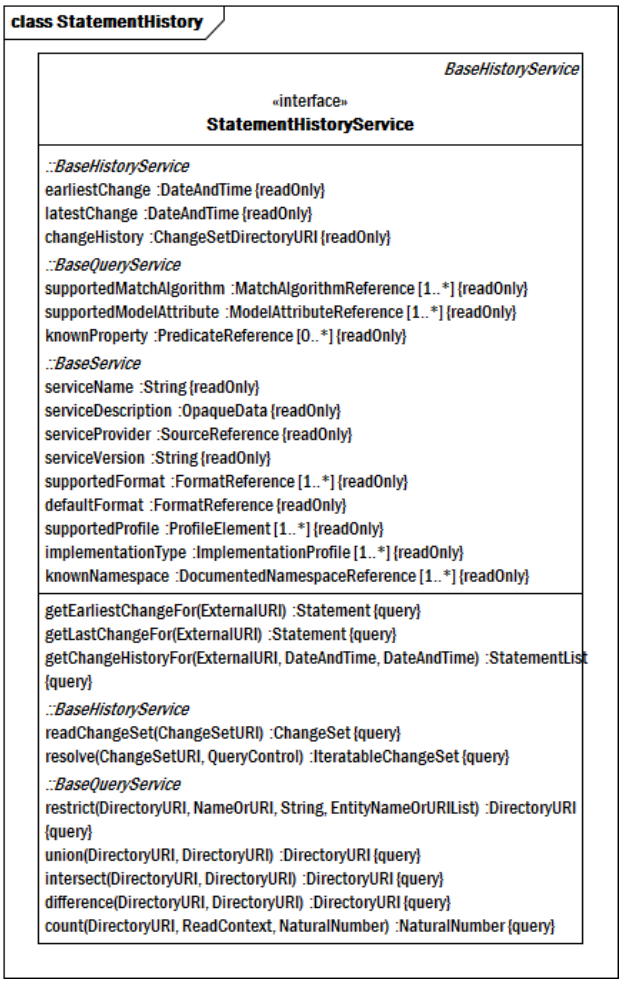

Figure 2.3 - Statement History

3.3.1 Interface StatementHistoryService

The `StatementHistoryService` provides the ability to view the history of changes for an individual statement. A key to implementing this service is the ability to uniquely identify a statement, which is the equivalent of constructing some sort of identifier in the process of reification. The CTS2 specification is silent when it comes to how these identifiers are formed, but does note that the ability to talk meaningfully about statements globally (vs. within the context of a single service) requires that a reproducible algorithm needs to be available.

Superclasses

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

3.3.1.1 Operation: getEarliestChangeFor

Return the earliest known version of the statement.

Input Parameters

- **statementID** - a URI that identifies a `Statement` (Type: `ExternalURI`).

Return Type: `Statement`

Exceptions

- **UnknownStatement** - The referenced statement is not recognized by the service.

3.3.1.2 Operation: getLastChangeFor

Input Parameters

- **statementID** - a URI that identifies a Statement (Type: `ExternalURI`).

Return Type: `Statement`

Exceptions

- **UnknownStatement** - The referenced statement is not recognized by the service.

3.3.1.3 Operation: getChangeHistoryFor

Return a list of records that reflects the change in the catalog entry over the specified time span. Records will be sorted from earliest to latest if (a) `fromDate` is omitted or (b) `toDate` is omitted, or (c) `fromDate` is earlier than `toDate`. In all other cases, records will be sorted from latest to earliest.

Input Parameters

- **statementID** - a URI that references a Statement (Type: `ExternalURI`).
- **fromDate**$_{OPT}$ - the earliest date to return changes for. If omitted, all changes up to the last one are returned. (Type: `DateAndTime`)
- **toDate**$_{OPT}$ - the last date to return changes for. If omitted, all changes up to the last one are returned. (Type: `DateAndTime`)

Return Type: `StatementList`

Exceptions

- **UnknownStatement** - The referenced statement is not recognized by the service.