HL7 Common Terminology Services 2 Service Functional Model (SFM)

From ApelCore

Service Functional Model Specification

Common Terminology Services Release 2 (CTS 2)

Version 1.0

19-February-2009

Project Leads	Russell Hamm rhamm@apelon.com (mailto:rhamm@apelon.com) (Apelon, Inc.) Craig Stancl stancl.craig@mayo.edu (mailto:stancl.craig@mayo.edu) (Mayo Clinic/Foundation)
Principal Contributors	Kristi Eckerson keckers@sph.emory.edu (mailto:keckers@sph.emory.edu) (Emory University) Davera Gabriel davera@ucdavis.edu (mailto:davera@ucdavis.edu) (University of California, Davis) Russell Hamm rhamm@apelon.com (mailto:rhamm@apelon.com) (Apelon, Inc.) Alan Honey aphoneysys@gmail.com (mailto:aphoneysys@gmail.com) (Independent consultant to Booz&Co / II4SM) Senthil Nachimuthu snachimuthu@mmm.com (mailto:snachimuthu@mmm.com) (3M Health Information Systems) Craig Stancl stancl.craig@mayo.edu (mailto:stancl.craig@mayo.edu) (Mayo Clinic/Foundation)
Contributors	John Carter jcarter@apelon.com (mailto:jcarter@apelon.com) (Apelon, Inc.) Sundak Ganesan dsq3@CDC.GOV (mailto:dsq3@CDC.GOV) (SAIC Consultant to CDC) Stanley Huff, M.D. stan.huff@ihc.com (mailto:stan.huff@ihc.com) (Intermountain Health Care)
HL7 Vocabulary Co-Chairs	Heather Grain heather@lginformatics.com (mailto:heather@lginformatics.com) (Latrobe University) Russell Hamm rhamm@apelon.com (mailto:rhamm@apelon.com) (Apelon, Inc.) Ted Klein ted@tklein.com (mailto:ted@tklein.com) (Klein Consulting) Beverly Knight beverly.knight@procurement.ca (mailto:beverly.knight@procurement.ca) (Canada Health Infoway)

Preface

Notes to Readers

This document is the Service Functional Model for the Common Terminology Services 2 specification, which is specified under the Service Development Framework process under the auspices of the Healthcare Services Specification Project (HSSP). Further context is given in the overview section below, but one key point to note is that the SFM provides a Service **Interface** specification, NOT the specification of a Service implementation. This is a critical distinction in terms of Service Oriented Architecture. There could be many different ways of implementing all or part of the functionality to support the behavior described in this specification.

NOTE: For the purposes of this specification, the terms *vocabulary* and *terminology* are used interchangeably.

Changes from Previous Release

This is the first public release of this document.

Acknowledgments

This document is the result of the collaboration of many individuals and organizations. The terminology and standards community - all involved in the numerous meetings and teleconferences are to be thanked for their contributions and support. In addition to the listed authors, the following individuals are acknowledged for their contributions and support during the development of this specification.

- The HL7 Vocabulary Technical Committee
- Christopher Chute, M.D., PhD. (Mayo Clinic/Foundation)
- John Koisch (OCTL Consulting)
- Ken Rubin (EDS)
- Tony Weida, PhD. (Apelon, Inc.)

Note: Sections of this document in blue indicate text that is consistent across HSSP specifications.

Contents

Overview

Introduction and Scope

The Service Specification Development Framework Methodology is the methodology followed to define HSSP specifications. The methodology sets out an overall process, and also defines the responsibilities of the Service Functional Model (SFM). Section 2 sets out the business context for this particular specification, but firstly it is important to understand the overall context within which this specification is written, i.e. its purpose from a methodology standpoint.

HL7-OMG Healthcare Services Specification Project (HSSP)

The Healthcare Services Specification Project (HSSP) [http://hssp.wikispaces.com (http://hssp.wikispaces.com/)] is a joint endeavor between Health Level Seven (HL7) [http://www.hl7.org (http://www.hl7.org/)] and the Object Management Group (OMG) [http://www.omg.org (http://www.omg.org/)]. The HSSP was chartered at the January 2005 HL7 meeting under the Electronic Health Records Technical Committee, and the project was subsequently validated by the Board of Directors of both organizations.

The HSSP has several objectives. These objectives include the following:

- To stimulate the adoption and use of standardized “plug-and-play” services by healthcare software product vendors
- To facilitate the development of a set of implementable interface standards supporting agreed-upon services specifications to form the basis for provider purchasing and procurement decisions.
- To complement and not conflict with existing HL7 work products and activities, leveraging content and lessons learned from elsewhere within the organization.

Within the process, HL7 has primary responsibility for (1) identifying and prioritizing services as candidates for standardization; (2) specifying the functional requirements and conformance criteria for these services in the form of Service Functional Model (SFM) specifications such as this document; and (3) adopting these SFMs as balloted HL7 standards. These activities are coordinated by the HL7 Services Oriented Architecture SIG in collaboration with other HL7 committees, which currently include the Vocabulary TC and the Clinical Decision Support TC.

Based on the HL7 SFMs, OMG will develop “Requests for Proposals” (RFPs) that are the basis of the OMG standardization process. This process allows vendors and other submitters to propose solutions that satisfy the mandatory and optional requirements expressed in the RFP while leaving design flexibility to the submitters and implementation flexibility to the users of the standard. The result of this collaboration is an RFP Submission, which will be referred to in the HSSP process as a Service Technical Model (STM). HL7 members, content, and concerns are integral to this process, and will explicitly included in the RFP creation and evaluation process.

It is important to note that the HL7 SFMs specify the *functional* requirements of a service, the OMG RFPs specify the *technical* requirements of a service, and the STM

represents the resulting technical model, except as specified below. In many cases, SFMs describe an overall coherent set of functional capabilities and / or define a minimum set of behaviors necessary to guarantee a minimal level of service in a deployment scenario. These capabilities may be specialized or subdivided from both functional and informational (semantic) perspectives to provide conformance "profiles" that may be used as the basis for the OMG RFP process and/or implemented.

Service Definition Principles

The high level principles regarding service definition that have been adopted by the Services Specification Project are as follows:

- Service Specifications shall be well defined and clearly scoped and with well understood requirements and responsibilities.
- Services should have a unity of purpose (e.g., fulfilling one domain or area) but services themselves may be composable.
- Services will be specified sufficiently to address functional, semantic, and structural interoperability.
- It must be possible to replace one conformant service implementation with another meeting the same service specification while maintaining functionality of the system.

A Service at the SFM level is regarded as a system component; the meaning of the term "(system) component" in this context is consistent with UML usage[1].A component is a modular unit with well-defined interfaces that is replaceable within its environment. A component can always be considered an autonomous unit within a system or subsystem. It has one or more provided and/or required interfaces, and its internals are hidden and inaccessible other than as provided by its interfaces.

Each Service's Functional Model defines the interfaces that the service exposes to its environment, and the service's dependencies on services provided by other components in its environment. Dependencies in the Functional Model relate to services that have or may in future have a Functional Model at a similar level; detail dependencies on low-level utility services should not be included, as that level of design is not in scope for the Functional Model.

The manner in which services and interfaces are deployed, discovered, and so forth is outside the scope of the Functional Model. However, HSSP Functional Models may reference content from other areas of HSSP work that deals with architecture, deployment, naming and so forth. Except where explicitly specified, these references are to be considered informative only. All other interactions within the scope of the scenarios identified above are in the scope of the Functional Model.

Reference may be made to other specifications for interface descriptions, for example where an interface is governed by an existing standard.

Overall disclaimers

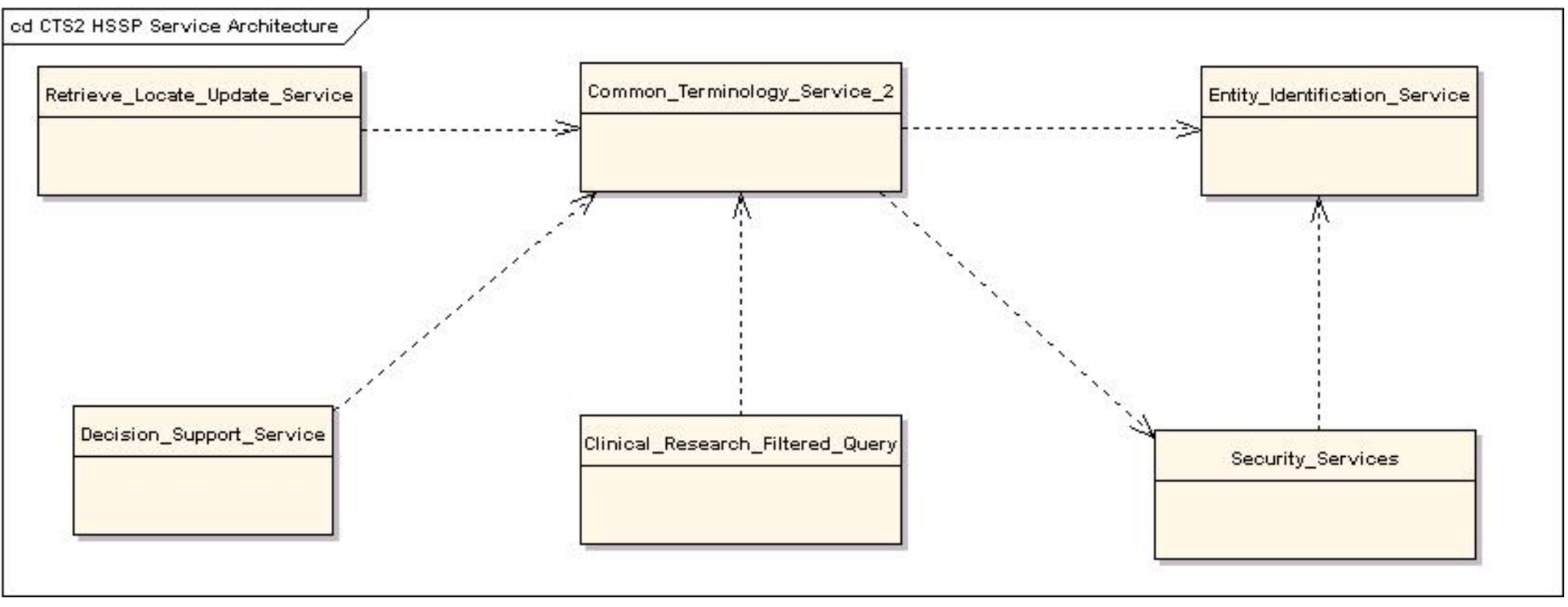
- Examples are illustrative and not normative unless otherwise specified
- The scope of information content of HSSP service specifications is not limited to HL7 content models. At a minimum, however, specifications should provide a semantic profile as part of its conformance profile to provide support for HL7 content models where applicable.

Context of this SFM within HSSP Roadmap

As described above, the purpose of an HL7 SFM is to identify and document the functional requirements of services deemed important to healthcare. Accordingly, the CTS 2 service provides a critical component within the larger context of service specifications in that it defines both the expected behaviors of a terminology service and a standardized method of accessing terminology content. This consistent approach to terminology interaction will benefit other business context services by providing a level of terminology interoperability that currently only exists in a limited form.

Once adopted as an HL7 standard, it is anticipated that the CTS 2 service will serve as the basis for one or more OMG technical specifications. It is expected that CTS 2 will effectively leverage other HSSP specifications to enhance overall functionality in integration environments. In particular, the CTS 2 service is expected to interact with one or more infrastructure services as outlined below.

At a minimum, it is expected that CTS 2 will make use of an Entity Identification Service, which in turn references a set of Security Services. CTS 2 itself will make use of the Security Services to implement its own functional profile restrictions. Additionally, services such as a Decision Support Service, Clinical Research Functional Query, and Resource Locate and Update Service may find the use of the CTS 2 service a key resource in improving content disambiguation.



This specification will provide an important foundation component for many healthcare interoperability scenarios, both within and across organizations. Although in many business scenarios CTS 2 may be used in conjunction with other services, it has been specified to provide stand alone capabilities when referenced solely for terminology access and management purposes.

Service Overview and Business case

Service Overview

CTS 2 Service Description and Purpose

The goal of the Common Terminology Services 2 (CTS 2) Specification is to expand on the original functionality outlined in HL7's Common Terminology Service (CTS) Specification. CTS 2 defines the functional requirements of a set of service interfaces to allow the representation, access, and maintenance of terminology content either locally, or across a federation of terminology service nodes.

The CTS 2 specification strives to expand on the original functionality outlined in HL7's Common Terminology Service specification, specifically looking to:

1. Establish the minimal common structural model for terminology behavior independent from any specific terminology implementation or interchange model, and how it is related to meta-data (information about data) and data (the information itself)
2. Integrate into CTS 2 the functional coverage outlined in the existing CTS specification.
3. Specify both an information and functional model that addresses the relationships and use of terminology, e.g. how value sets are built and queried, and how terminological information is validated.
4. Specify the interactions between terminology providers and consumers – how terminology users can submit unambiguous requests for corrections and extensions and how revisions to content are identified, distributed and integrated into running systems.
5. Specify how mapping between compatible terminologies and data models is defined, exchanged and revised.
6. Specify how logic-based terminologies can be queried about subsumption and inferred relationships.
7. Engage broad community participation to describe the dimensions of use and purpose for vocabularies and value sets. This aim will attempt to harmonize these efforts in terms of models, use cases, and requirements for creating a functional model for CTS 2.

Scope

To address the above stated purpose of CTS2, the scope of functionality addresses several broad categories.

Terminology services represent functions necessary to manage, search, and access terminology content. Terminology services provide a consistent specification for accessing and managing terminology content, independent of the terminology content and underlying technology stack. Terminology content represents various resources including lists, value sets, taxonomies, and formal description logic based ontologies. The following thematic areas are considered in scope for CTS 2.

- **Administration**: This is a set of functionality that provides the ability to manage content as part of a terminology service. Administration functions include the ability to load terminologies, export terminologies, activate terminologies, and retire terminologies. These functions are generally protected and accessible by service administrators with appropriate authorization.
- **Search / Query**: This is a set of functionality that provides the ability to find concepts based on some search criteria. This includes restrictions to specific associations or other attributes of the terminology, including navigation of associations for result sets. This represents the primary utility for using terminology content in a number of application contexts.
- **Authoring / Maintenance:** This is a set of functionality that provides the ability to create and maintain content. From a terminology service perspective, this would include the appropriate APIs to add, change, or delete concepts and associations. This would also include the processing of change events from various terminology providers.
- **Associations**: This is a set of functionality that provides the ability to map concepts and the concept's associated attributes from a source terminology to a concept in a target terminology, or create relationships between concepts within a single code system.

CTS 2 is intended to allow the look up and management of a wide variety of terminology components, including, but not limited to, Concepts, Associations, and Value Sets. This includes the ability to resolve content bound to a specific *Context of Use* (Concept Domain) or *Jurisdictional Domain* (Realm). At the functional level, the service interface will explicitly allow the query, definition, publication, and modification of the different terminology components that are required of terminologies and terminology services.

Conformance profiles are defined in this specification which are intended to focus specific implementations of CTS 2 to address a specific class of functionality and pre-define minimum trait sets for each specified functional class. This will also allow for performance optimizations to be defined for terminology searches and queries (which are implementation considerations which will be considered in the technical specification arising from the OMG RFP process) The scope of this functional specification covers support for multiple terminology sources and a federated terminology environment.

The reason why the service is necessary

The original HL7 CTS specification deliberately steered clear of developing a generic model of terminology, and avoided issues related to terminology distribution and versioning. The value set, or sub-setting section of CTS focused on static value sets and didn't fully address the definition or resolution of value sets that define post-coordinated expressions – issues that are now in scope.

Adopting organizations have recognized the existing HL7 CTS standard serves an important role in defining the common functional characteristics that a terminology service (either internal or external) must be able to provide. However, these organizations are also realizing that CTS fails to address many of the maintenance, versioning and represnetation requirements necessary for a truly interoperable terminology service.

CTS 2 as a commonly accepted standard for terminology services, will enhance the capabilities of the initial CTS specification for sub-setting and mapping, and extend the specification into domains such as terminology distribution, authoring, versioning, and classification. Standardizing the functionality at this level will allow applications using terminology services to build on a common infrastructure, and improve interoperability at the terminology layer across applications.

CTS 2 will provide the terminology community with a defined set of standards interfaces that can be used to evaluate terminology source structure, terminology source content, and terminology tools.

Structure of the CTS 2 Service

In order to provide for the maximum implementation flexibility, this functional model defines several enumerated functional profiles for CTS 2. These profiles serve to subset and focus the functionality of a CTS 2 implementation to accomplish a targeted set of tasks. These profiles include:

- **Minimal CTS 2 Profile** - The minimal functional coverage necessary for a service to declare itself as being a conformant CTS 2 service. The minimal CTS 2 includes capabilities for searching and query terminology content, representing terminology content in the appropriate HL7 Datatypes, and structuring terminology content appropriately when HL7 Datatypes are not available for representing the necessary terminology content being queried (i.e. value sets.)
- **Terminology Administration Profile** - The functional operations necessary for terminology administrators to be able to access and make available terminology content obtained from a Terminology Provider. Terminology Administrators are required to interface with Terminology Provider systems in order to obtain the terminology content, then load that terminology content on local Terminology Servers.

- **Terminology Authoring Profile** - The functional operations necessary for terminology authors to analyze the existing terminology content, as well as directly edit terminology content.

The degree to which an organization's interoperability deployment supports a conformance profile is directly related to agreements implemented with a business partner. A single CTS 2 service may respond to different real-world business partners depending on the underlying agreements and needs. For example, an organization may implement a CTS 2 (Authoring) compliant service with a trusted partner (i.e., a Terminology Provider). A separate partner may only be allowed CTS 2 (Minimal) access to the content from that Terminology Provider as dictated by other factors.

Additionally, CTS 2 explicitly makes no distinctions at the functional level regarding semantics of the underlying systems. Instead, it provides for a semantic profile as part of CTS 2 conformance profiles. This allows definition, publication, and discovery of vital semantic artifacts between sharing partners through CTS 2 interfaces without requiring strict, tightly coupled integration. Thus, CTS 2 does not preclude a strategy for semantic interoperability to be realized, though it would likely depend on other factors (for example, a security service and / or an entity identification service). This improves CTS 2 as an interoperability mechanism by relegating the issue of semantic interoperability to the trading partners, allowing semantic transformations to be performed at the least cost for the most derived value.

Implementation Considerations

Interface Interoperability Considerations

CTS 2 is an interface specification, not an implementation specification. As such, it is intended to be an interoperability mechanism for terminology resources between applications. There is nothing inherent in the CTS 2 specification that restricts its use to be within a single organization. To the contrary, CTS 2 is intended to expose a single or multiple terminology sources for use by various applications that may or may not be within the same organization, providing a standardized method for terminology access.

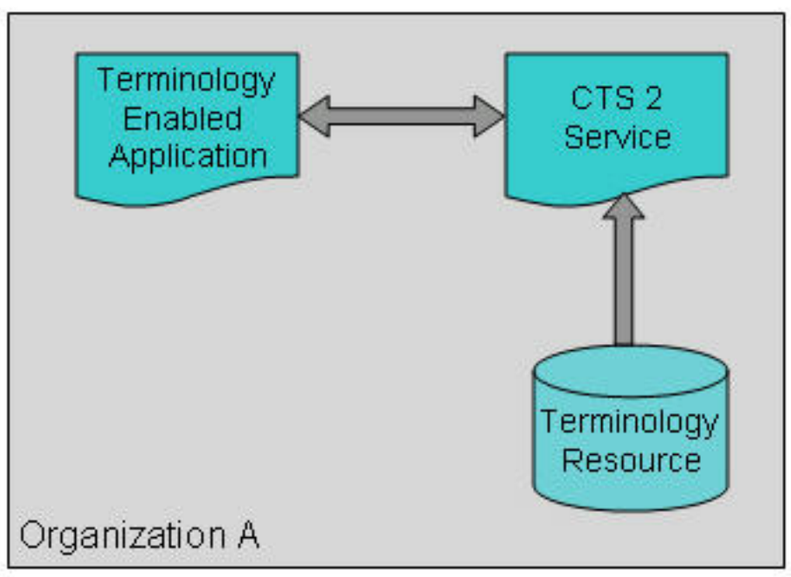


Figure 2.4-1 CTS 2 Service Accessed by a Single Organization

CTS 2 will provide for terminology interoperability between organizations. While coded concepts from structured terminology can unambiguously identify the concept(s) being communicated, a standard way of structuring and communicating those coded entries is required.

CTS 2 can be used in an inter-organizational setting where each organization maintains its own security and application specific provisions. CTS 2 will enable consistent access to a high availability or international standard terminology resource, made available to subscribers via a CTS 2 interface.

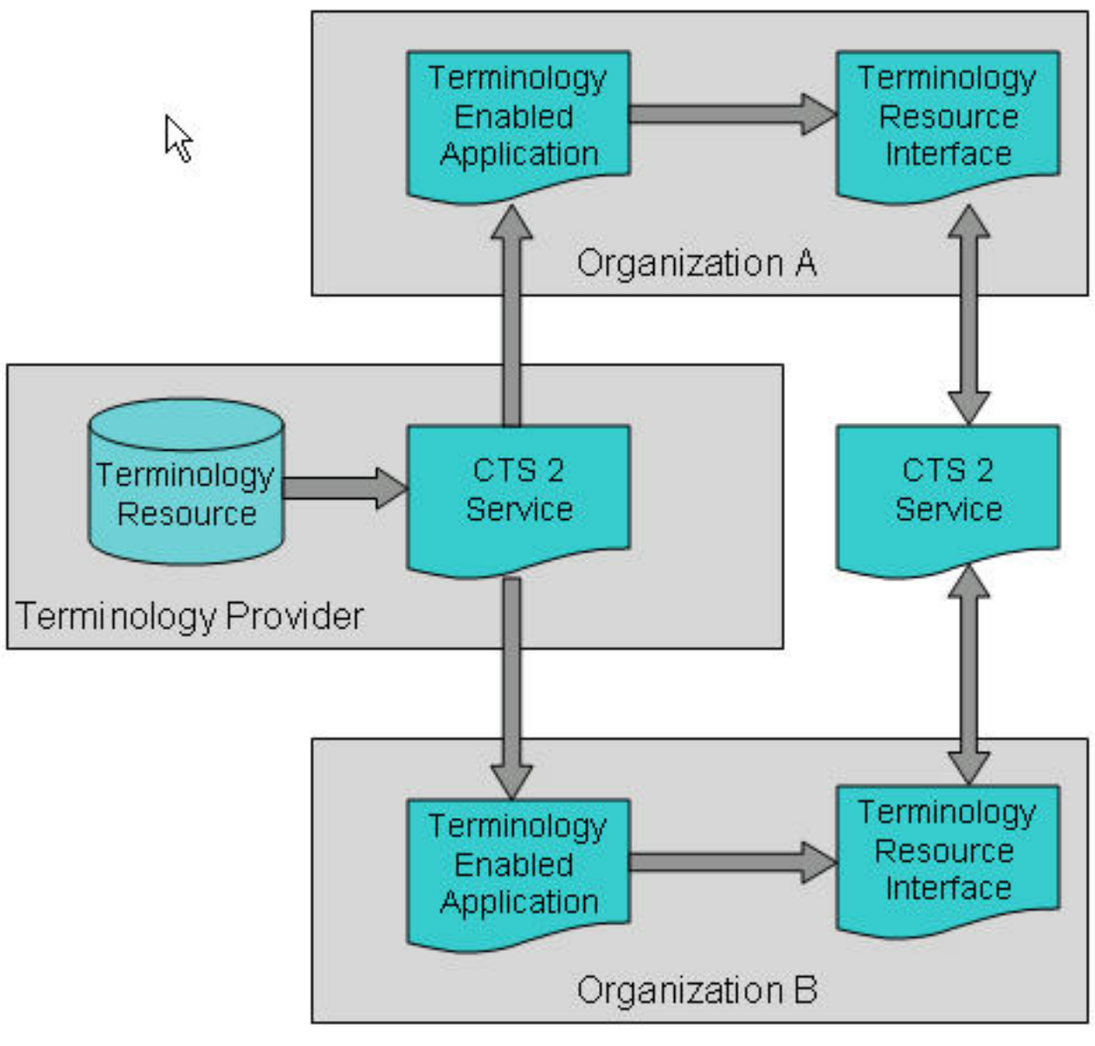


Figure 2.4-2 Multi Organizational access to a CTS 2 Service

Since terminology content is not static, CTS 2 will also provide functionality to maintain and update terminology content. Updates and update requests to terminology sources need to be reviewable and traceable over time. Often, a terminology source provider will want to maintain the "gold standard" or master release of a code system, as to maintain a consistent standard terminology that can be used across multiple organizations and realms. Notwithstanding, users of any given source terminology may wish to

extend that terminology for their own use, and may even wish to recommend the addition of those “local” extensions to the terminology provider to be included as part of the release.

CTS 2 will provide a mechanism to allow for terminology users to extend a given terminology, share those extensions with others, or feed those extensions back to the source provider in a structured format to be reviewed, modified as necessary, and fed into a CTS 2 server as input to update the source terminology with the content contained in the change request. As depicted in Figure 2.4-3, Organization A is applying its own local extensions to a terminology resource being served by a CTS 2 service. In addition to applying its own local extensions, Organization B is feeding some of those local extensions back to the terminology provider as suggestions to be included in the next release of the code system.

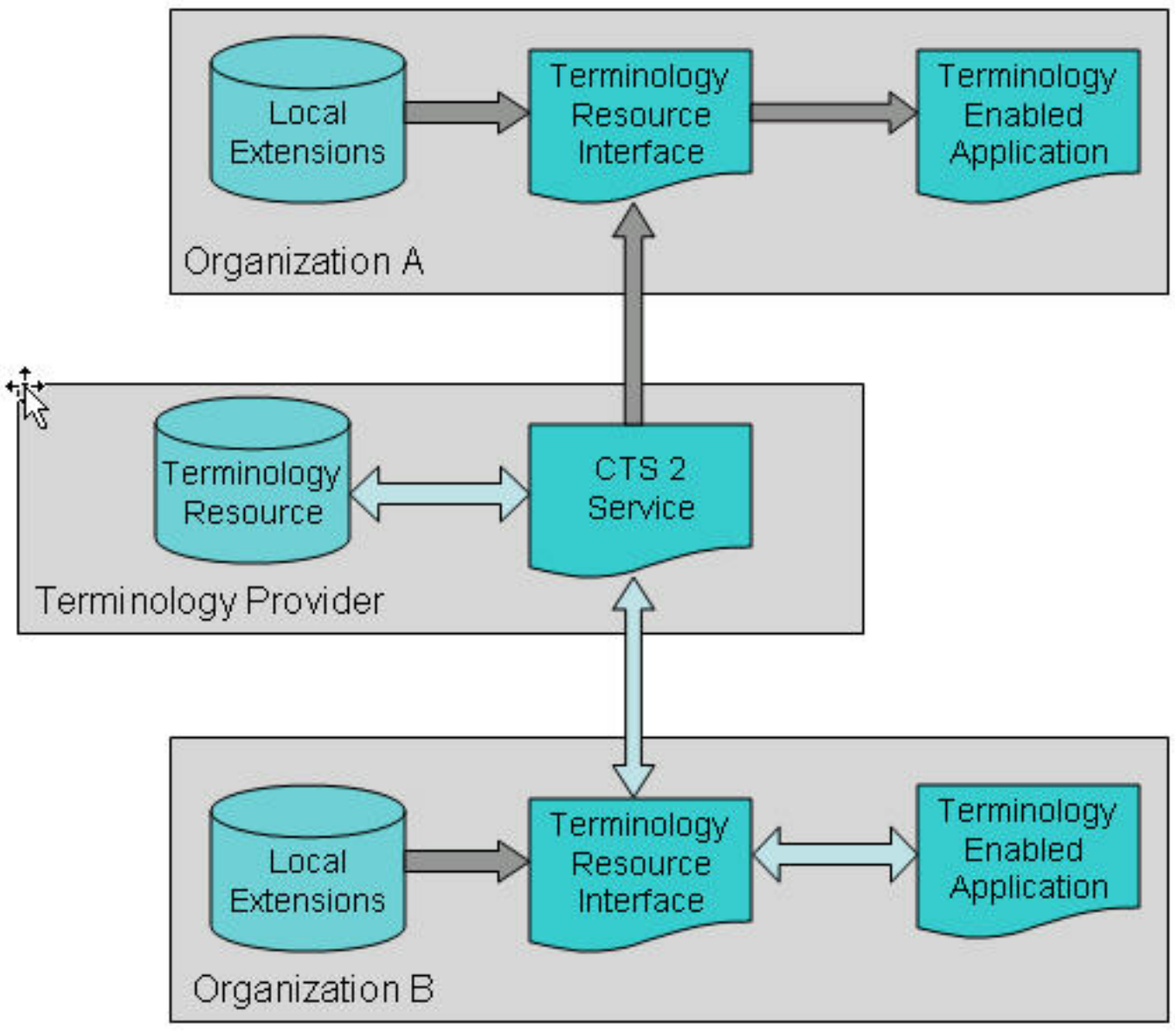


Figure 2.4-3 Multi Organization Access with Write Permissions by One Organization

Terminology Structure Considerations

Controlled terminologies are often developed with specific purposes and use cases in mind, and as such are often structured very differently. The format of any given terminology may range from from a simple flat list of concepts, to complex poly-hierarchies. The variety of attributes of the entities of code systems vary as well. Even the formats of the identifiers are different, with some concept identifiers being meaningless identifiers, to others which have explicit or implied meaning.

The functional components of CTS 2 must be able to operate on this broad spectrum of terminology sources. At a minimum, CTS 2 must specify a concept based terminology model that is capable of representing most varieties of structured terminologies. The basic structure of the code system is illustrated in the **CTS 2 Conceptual Model** below. This model outlines the various terminology components and the cardinality between them but does not dictate particular levels of data normalization or other technical details of implementation.

The purpose of this model is to define the semantics of the requirements of CTS 2. It is intended to assist with outlining the expected behaviors and interactions of terminology components that are outlined served by a CTS 2 terminology service.

NOTE: An platform independent implementation model will be necessary as part of the technical specification developed as a response to the RFP for CTS 2. It is expected that this PIM would be employed as a shared terminology model, which can represent the behaviors and semantics of a large set of terminology sources.

This is a logical (analysis level) model, although it is the intent that most if not all key attributes that affect behavior have been defined, unless specifically excluded below. It is the intent of this model to support both *well behaved* and more arbitrarily defined code systems, although some level of form and structure must be imposed by the model. Code Systems, Concepts and Value Sets have specific*version* classes. Additionally *curation* has been supported explicitly by inclusion of a number of attributes, e.g. versionOrder, status and status date to appropriate classes. Status date in particular can be seen as a denormalization. Beyond that, specific concerns of audit trail / history have not been explicitly represented. In a typical implementation, it is likely that more than one status value may be needed on certain classes, covering different purposes (e.g. curation vs actual *business* state of the class) This level of detail can be fully resolved in the detailed design specification (PIM), as will the actual values of the various state related attributes.

Also a note on the use of "identifiers". In general, identifiers have been included to ensure uniqueness and would be more for system generation, especially on some of the lower level classes.

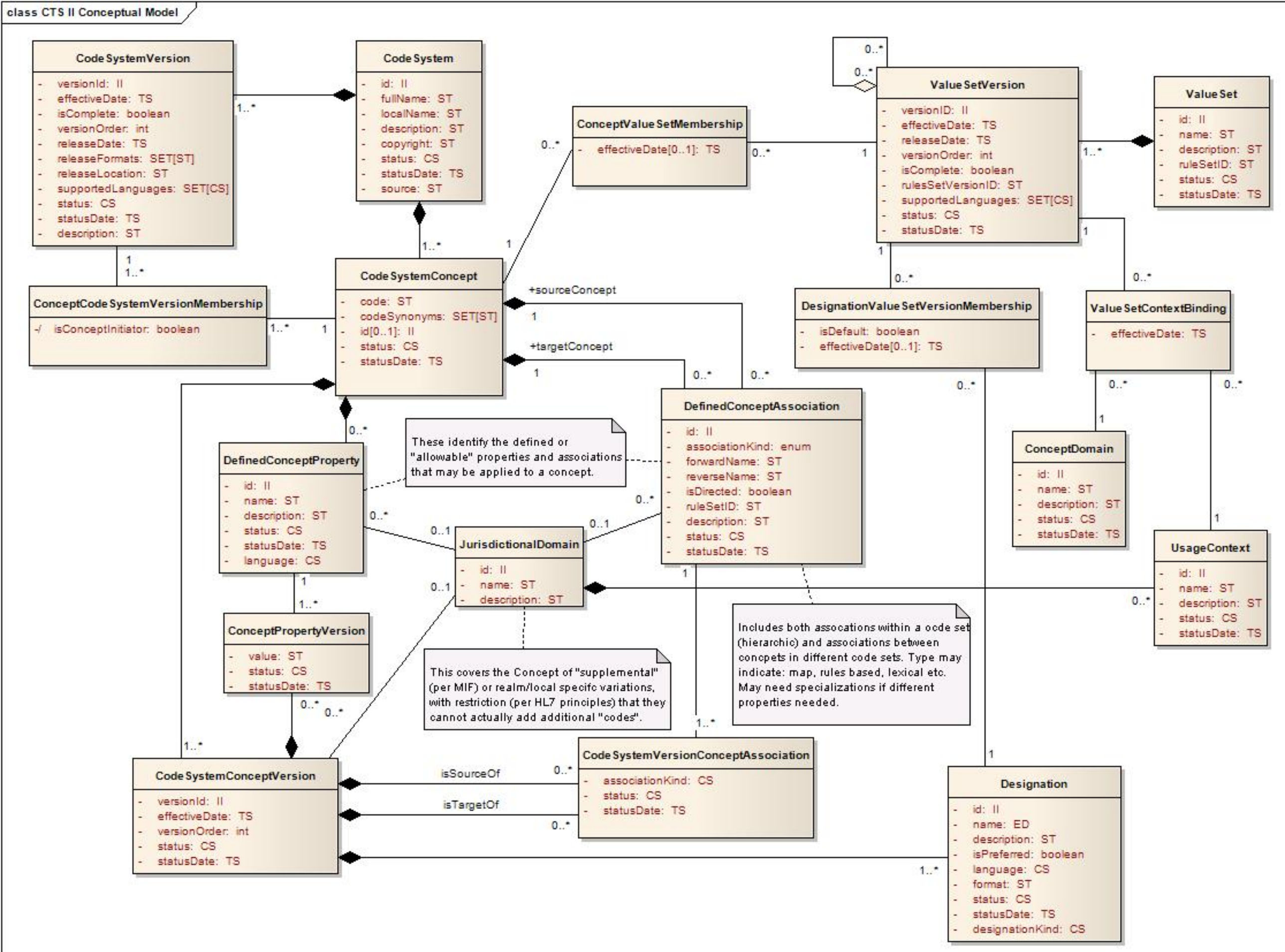


Figure 1 CTS 2 Conceptual Model

CodeSystem

A code system is defined as a collection of uniquely identifiable concepts with associated designations, associations and meanings. Examples of code systems include ICD-9 CM, SNOMED CT, LOINC, and CPT. To meet the requirements of a code system as defined by HL7, a given code must resolve to one and only one meaning within the code system. In the terminology model, a code system is represented by the Code System class. Code systems themselves can be concepts and each can be represented by a Concept class with a unique identifier.

At a minimum, Code Systems have the following attributes:

- An identifier (id) that uniquely identified the Code System
- A name (fullName) that is the official name of the code system as assigned by the terminology provider.
- A name (localName) that the Code system is normally referred to
- A description (description) that describes the Code System. This may include the code system uses and intent.
- Copyright information (copyright) pertaining to the Code System
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)
- An attribute (source) that identifies the authority or source of the terminology. i.e. IHTSDO

CodeSystemVersion

Code systems are not generally static entities and change over time. A CodeSystemVersion is a static snapshot of a CodeSystem at a given point of time, and specifies the version of the code system in which any given concept can be found. A CodeSystemVersion is represented by attributes including:

- A version identifier (releaseVersion) that uniquely identifies each version of a Code System
- The start date (effectiveDate) when the version is deemed to be valid for use.
- A flag (isComplete) indicating that the version in question is complete (i.e. standalone) or requires other previous or later versions to be complete.
- An optional ordering parameter (versionOrder) that identifies the order in which the version should be applied (used for version deltas).
- A date (releaseDate) that represents the date when the version of the Code System became available
- The format (releaseFormats) that indicates the format(s) that the version of the Code System is available in.
- The official location (releaseLocation) where the version of the code system is available from
- The different languages (supportedLanguages) supported by the Code System
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

CodeSystemConcept

A Code System Concept defines a unitary mental representation of a real or abstract thing within the context of a specific Code System; an atomic unit of thought. Generic representations of concepts outside of the bounds of a code system are not included in the model, although their effect can be approximated by setting up Concept Associations (maps) across code systems. Concepts should be unique within a given code system, but may have synonyms in terms of both the codes used (e.g. “l” and “L” in UCUM for Liter) and in textual representation (as Designations). Concepts may be simple or compositional in nature. A compositional concept is one that contains more than one concept concatenated within it – for example “severe hypertension” – a combination of “hypertension” and a qualifier for severity. Each CodeSystem will have a set of Concepts associated with it. Terminology best practices dictate that concepts are not deleted from code systems, but are instead deprecated or retired from use, although nothing in the model prevents this. CodeSystemConcepts are represented by attributes including:

- A unique concept code (conceptCode)
- A set of zero or more code synonyms (codeSynonyms)
- An optional, but unique where used, identifier (id) for use in disambiguation for less well behaved code systems that reuse concept code values.
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

In the model, this class is not explcitly linked to the various version of information for a Concept. At the conceptual level, this can be derived by looking at the effective dates. However, it is envisaged that explicit derived relationships woudl be used at the implementation level.

JurisdictionalDomain

A JurisdictionalDomain identifies an organization or other domain that may define and manage its own code systems or concepts, including localization of a broader code system. Its presence in this model is specifically to allow for localization of certain concept elements. HL7 rules prohibit new codes being added to a code system locally, but do allow for additional concept relationships, concept properties and designations. (However, any organization could use the same model, interfaces etc. to define its own code systems for internal use.) This class provides the link to those classes to enable the localization to be recorded. The Jurisdictional Domain has the following attributes:

- An identifier (id) that uniquely identified the Jurisdictional Domain
- A name (name) that the Jurisdictional Domain is normally referred to
- A description (description) that describes the Jurisdictional Domain.

DefinedConceptProperty

A defined concept property is a named characteristic of a concept that can be assigned a value. In the terminology model a defined concept property is represented by the DefinedConceptProperty class. This explicitly defines the allowable concept properties for any given Concept within a Code System and not individual changes or versions or what is supported by a specific version of a code set. Each Concept may have zero to many concept properties. Each Code System defines the properties supported in a specific version using the ConceptPropertyVersion class, where the actual values of the concept properties are defined. Defined Concept Properties are represented with attributes including:

- A code (id) that uniquely identifies the property
- A name (name) for the property
- A description (description) of the property
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)
- The language (language) in which the property is expressed

DefinedConceptAssociation

Associations define the relationships or linkages between concepts. For example, in SNOMED CT, the concept of “pneumonia” has an “is-a” relationship to the concept of “lung consolidation,” and “lung consolidation” has an “is-a” relationship to the concept of “disorder of lung.” This represents the logical conclusion that “pneumonia” is a “disorder of lung.” In the terminology model, allowable relationships are represented by the DefinedConceptAssociation class and are defined as directed semantic relationship triples between two concepts. It is not mandatory for concepts to have associations to other concepts. However, when associations exist, the cardinality and the explicit declaration of source and target would indicate the directionality that restricts the designation of the association. For example, from the concept relationship (an association between concepts within a single code system) in the above example, we can infer that “pneumonia” is a “disorder of lung,” but the inverse concept relationship of “disorder of lung” is-a “pneumonia” cannot be inferred. If we want the inverse concept relationship, it must be explicitly stated, that is, there has to be a specific relation of “disorder of lung” “is-a” “pneumonia. In the case of Concept Maps (where the source and target concepts are from different code systems) the direction and designation of the relationship have similar restrictions, except in the case where the Concept Map indicates semantic equivalence. The equal association in this case obviates the requirement for interpreting the association direction. A ConceptAssociation links a source Concept to a target Concept. As with DefinedConceptProperty, there is a separate Concept Version level representation (CodeSystemVersionConceptAssociation) to identify the Concept Associations supported within a specific version of a Code System. Concept associations are minimally defined by attributes including:

- A code (id) that uniquely identifies either the association instance (for concept maps) or type (for concept relationships).
- A type (associationKind) that describes the nature of the association (e.g. ConceptMap vs Hierarchic relationship within a code system) - see also note below.
- A name (forwardName) that represents how the association should be represented when reading from source concept to target concept.
- A name (reverseName) that represents how the association should be represented when reading from target concept to source concept.
- A flag (isDirected) indicating whether the association is one-way or can be interpreted as semantic equivalence.
- A description (description) that provides a textual definition of the relationship
- A rule set Identifier (ruleSetId) which provides further constraints on the nature of the relationship (see CreateConceptRelationship operation definition for details)
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

A further note on associationKind. This is intended to distinguish between lexical, rules based and explicitly enumerated relationships. Further reltionship types can be explicitly defined using the operations that manipulate relationship types explicitly. Further investigation will determine whether we actually need two levels of this attribute to distinguish between these purposes. This will be investigated further during production of the detailed specification (PIM).

Also, there is no "history" record identified for this class, since it is viewed as "immutable", i.e. any change would give rise to a new relationship and the previous one is deprecated.

CodeSystemConceptVersion

Although the actually value of a code will not change, associated information may, e.g. associations, designations and concept properties. The Code System Concept Version provides a means to organize and manage variations to these elements over time. Typically, these variations will coincide with new versions of the owning code system. The model can represent this, but does not impose this as a restriction. Note also that "good" terminologies will normally not change concept relationships within the code system but retire the concepts and re-author them with the new relationships. Code System Concept Versions are represented by attributes including:

- A version identifier (versionId) that uniquely identifies each version of a Code System Concept Version.
- The start date (effectiveDate) when the version is deemed to be valid for use.
- An optional ordering parameter (versionOrder) that identifies the order in which the version should be applied (used for version deltas).
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)
- A description (description) of the version

Note that this representation is intended to indicate that a new code system concept version would be created for any change to designations, properties or associations, so individual history is not depicted for each of these items. This can be considered further when the detailed design is carried out.

ConceptCodeSystemVersionMembership

This class represents the use (or membership) of a specific Concept within a specific Version of a Code System. This has the following attributes:

- A (derivable but useful) indicator (isConceptInitiator) that identifies which Code System Version initiated the definition of the Concept.

(The absence of date attributes on this class deliberately imposes the restriction that a new concept being added would require a new code system version. In the case of code systems defined and maintained within an organization, they could lift that restriction by adding date information here)

ConceptPropertyVersion

This explicitly defines the supported concept properties for any given Version of a Code System. For example, the concept of Hematocrit with a LOINC code of 11271-4 has a specimen property with the value of "blood" and a method property with the value of "automated count." The specimen and method properties are part of how LOINC assigns the code and, when these properties change, a different LOINC code will be assigned. This implies that concept properties do not change over time. However, exceptions may be possible when the addition of a new property or change in property value does not change the concept and code (however it would result in a new "CodeSystemConceptVersion" as represented in this model, where the effective date is held). For example, if LOINC decides to add "analyte chemical structure" as a new property, there may not be a need to change the existing LOINC codes since the new information can apply to all of the LOINC concepts. As a result, each Code System may have its own unique set of concept properties associated with its Concepts for any specific Code System Version (note that the relationship to the CodeSystemVersion is not in the model explicitly but can be derived, and may be explicit in implementation models). Concept Property Versions are represented with attributes including:

- A value (value) of the property.
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

ConceptAssociationVersion

This explicitly defines the supported concept associations for any given Version of a Code System. (The same considerations of variability apply as for the ConcpetPropertyVersion described above). Concept Association Versions are minimally defined by attributes including:

- A type (associationKind) that describes the nature of the association (e.g. ConcpetMap vs Hierarchic relationship within a code system).
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

Designation

Concept designations are representations of concepts. The designation identifier must uniquely map to a given text string, bitmap, etc. within the context of the containing concept. In some terminologies, every unique text string will have exactly one identifier, which means that the same identifier may occur under more than one concept. In other terminologies, there may be more than one identifier for a given text string, meaning that the identifier is unique to the concept. Service software must not assume either model. For example, in SNOMED CT, the concept of "fever" has the fully specified name of "fever (finding)," a preferred name of "fever," and synonyms of "febrile" and "pyrexia." These are all designations for the concept of "fever." In the terminology model, designations are represented by the Designation class. Each Designation is a representation of the Concept and is assigned a unique designation identifier. In most instances, concept designations are human readable forms, but machine readable forms may also be present.

The Designation class is minimally defined by the following attributes:

- A unique identifier (id) for the designation
- A name (name) for the designation
- A description (description) for the designation
- A flag (isPreferred) indicating if the designation is preferred for the concept
- The language (language) in which the designation is expressed
- A format (format) for the designation
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)
- A type (designationKind) that describes the nature of the designation (e.g. human readable text vs BLOB, etc.).

Value Set

A value set represents a uniquely identifiable set of valid concept representations (codes), where any concept representation can be tested to determine whether or not it is a member of the value set. Value set complexity may range from a simple flat list of concept codes drawn from a single code system, to an unbounded hierarchical set of

possibly post-coordinated expressions drawn from multiple code systems. In the terminology model, a value set is represented by the ValueSet class. Value Set has the following properties:

- An identifier (id) that uniquely identifies the value set.
- A name (name) for the value Set
- A description (description) for the value set.
- An optional expression (ruleSetId) that defines (by value or reference) the algorithm to determine the members for “intensionally defined” value sets
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

Notes on use of Value Sets (and Value Set Versions):

1. Representation of “Intensional” Value Sets. These are defined by a superordinate concept, either immediately above or at a higher level, followed by the characteristic(s) that distinguish the concept from other concepts, i.e. using an algorithm (set of rules) that describe how to derive the actual concepts that belong in the Value Set. Since these rules can be arbitrarily complex, they are represented in the model in the form of an identifier of a rules set. This identifier has been included at both the Value Set and Value Set Version levels to allow for change of rules to be applied at whichever level is required. It is intended that the “owner” of the value set would define rules for when changes to a rule would result in a new value set vs a new version of a value set. By default, Intensional value sets contain the full set of designations defined by the intensional statement. **Examples** : a) Lymphozyte - A subtype of the cell type leukocyte with a function in the humoral or cytotoxic immune system, b) medicinal product - Any compound with a marketing authorization number granted by a national competent authority

2. Hierarchic and nested Value Sets. The model includes a “recursive” aggregation relationship on Value Set Version. This allows (as a convenience mechanism) for definition of Value Sets that “include” other Value Sets. This is explicitly restricted to inclusion of complete Value Sets (and not partial ones). Anything more complex would need to be described using the rule sets mentioned above. However this would allow for a Value Set to be defined that consists of a number of other complete Value Sets together with additional concepts intensionally or extensionally defined concepts. In all cases, at any given point in time, the definition of the Value Set Version MUST be able to resolve to an exact set of individual Concepts (together with their Designations where appropriate). This set might be hierarchical or a flat list.

3. Implementers should consider the workflow processes which are necessary to maintain and curate content in a distributed environment. Implementers should discuss how they would implement the workflow management and the coordination across nodes as a part of the implementation process, including how to bind functional components to workflow.

ValueSetVersion

A Value Set Version represents a point in time view of a Value Set. The Value Set Version identifies the set of concepts that are available in the value set for any specific version of the value set. It also allows for constraining the use of the Concept further by identifying specific Designations that may be used. Note that Value Set Version can be created by aggregation of other value set versions. A Value Set Version has the following attributes:

- An identifier (id) that uniquely identifies the value set version.
- An effective date (ValueSetVersion.effectiveDate) that identifies when the value set version became effective, i.e. is able to be used (bound to from information models).
- A date (ValueSetVersion.releaseDate) when the version of the value set was released (which allows for value set versions to be released in advance of their becoming "effective" or usable.
- An optional order (ValueSetVersion.versionOrder) that identifies the order in which the version should be applied
- A flag (ValueSetVersion.isComplete) that indicates whether the version of the value set is complete or not.
- An optional expression (ruleSetId) that defines (by value or reference) the algorithm to determine the members for “intensionally defined” value sets
- The different languages (supportedLanguages) supported by the Value Set Version
- A status to identify the state (mainly for curation)
- A status date to identify the date the status was set to its current value (for curation)

ConceptValueSetMembership

This provides the means to link a Value Set Version to the concepts which it includes. In “extensionally” defined value sets, this would be created manually, in “intensionally” defined Value Sets, this could optionally be used to record the result of applying the rule set at a given point in time (i.e. the date of the Value Set Version where this was required). As for ConceptCodeSystemVersionMembership, the absence of date attributes on this class deliberately imposes the restriction that a new concept being added would require a new value set version. This includes the following attribute:

- An optional date (effectiveDate) which would only be needed in “intensionally” defined Value Sets where the resulting concepts from applying the algorithm at a specific point in time were being explicitly instantiated and recorded.

DesignationValueSetVersionMembership

This identifies which Designations for Concepts may be used within a specific Value Set Version. This provides an additional level of flexibility for constraining the use of specific designations in specific circumstances. This includes the following attributes:

- An optional indicator (isDefault) which identifies whether the designation is the default designation representation for the concept in the value set.

- An optional date (effectiveDate) which would only be needed in “intensionally” defined Value Sets where the resulting concept designations from applying the algorithm at a specific point in time were being explicitly instantiated and recorded.

Concept Domain

Concept domains are intended to allow “late binding” of vocabulary. At design time, rather than referencing a specific value set, a given attribute (or datatype property) may reference a concept domain to provide an abstract description of “this is the type of stuff that goes here”. The same concept domain may be referenced in multiple designs, effectively saying “the same set of codes – whatever it is – that is used in place A should also be used here”. The Concept Domain has the following attributes:

- An identifier (id) that uniquely identified the Concept Domain.
- A name (name) that the Concept Domain is normally referred to.
- A description (description) that describes the Concept Domain.

Usage Context

A Usage Context identifies an environment or set of conditions in which a Value Set may be used. This can be at various levels of specificity, e.g. "Canadian pediatric psychiatry", "psychiatry", "disease codes" and so on. This is used in conjunction with Concept Domain to allow realms to bind to specific Value Sets in specific

circumstances. These may be "universal" or owned by a specific Jurisdictional Domain (Realm). (From a modeling perspective, we have classed universal as the highest level Jurisdictional Domain). The Usage Context has the following attributes:

- An identifier (id) that uniquely identified the Usage Context.
- A name (name) that the Usage Context is normally referred to.
- A description (description) that describes the Usage Context.

Value Set Context Binding

This provides a mechanism to bind a concept domain and a particular value set in a defined usage context. Note that the broader the context, the more the interoperability. At runtime, the operating context is identified and then used together with the concept domain referenced in the specification to resolve to the particular value set used. The Value Set Context Binding has the following attributes:

- An effective Date for which the binding may be used.

Business Scenarios

Scenario Actors

Actors will use the CTS 2 service for different purposes. These different actors can be generalized into a basic *Terminology User* an Actor that is simply an individual, organization, or application that requires access to terminology content for some purpose. Specializations of the *Terminology User* actor participate in additional operational specific scenarios that are defined by this Service Functional Model to address the Scope that is outlined in section 2.1.2. Actors described in this section are not necessarily human actors, but also include organizations and systems Figure3.1-2 outlines the specializations and composition of the different actors used in this specification. These actors are described below.

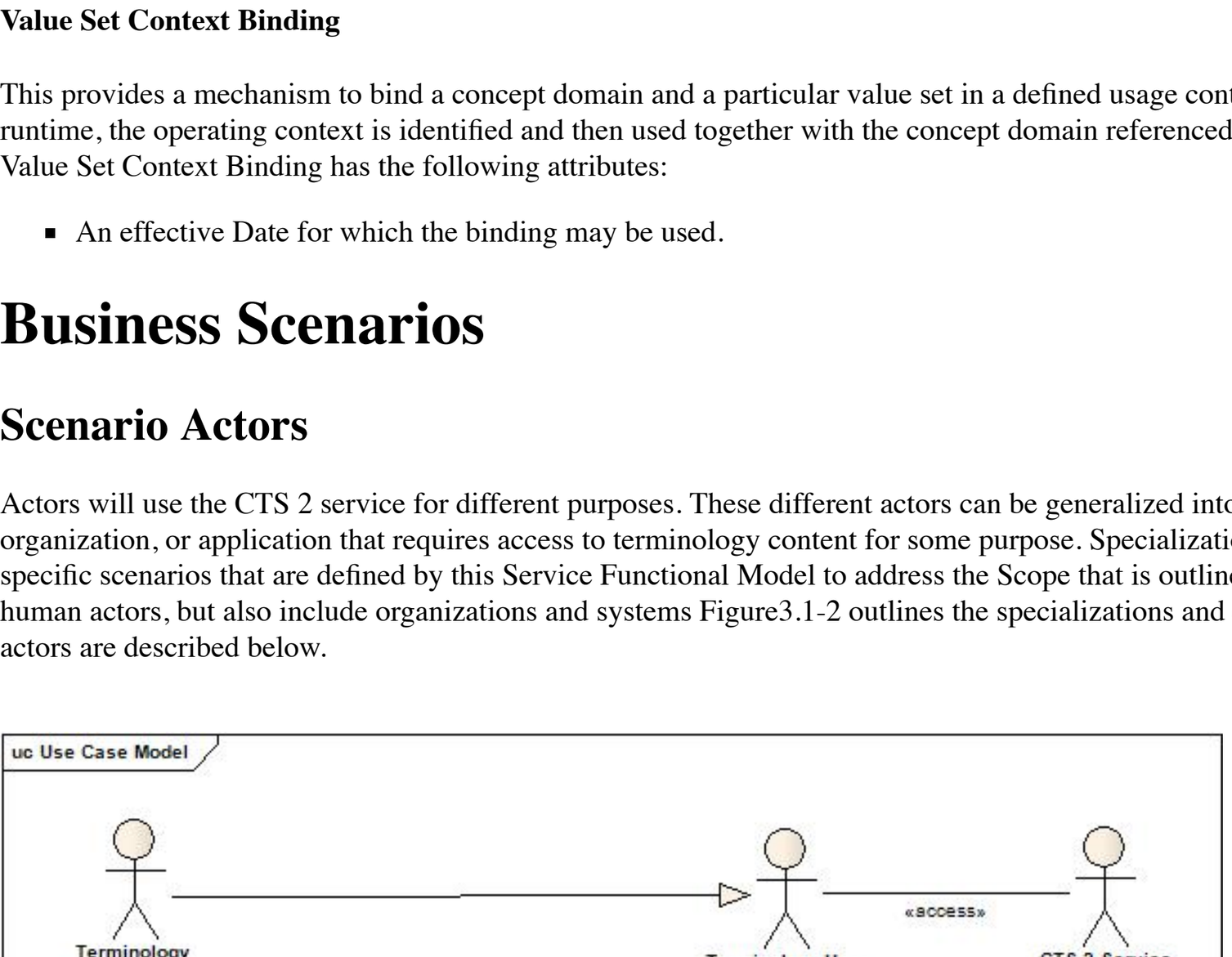


Figure 3.1-2

The following actors take a role in the CTS 2 scenarios.

- **CTS 2 Service**

The *CTS* 2 *Service* is a specific implementation of the CTS 2 Terminology Server.

- **Terminology User**

A *Terminology User* is an actor such as a subject matter expert, terminologist or terminology enabled application. *Terminology User* activities include, but are not limited to, querying for specific concept codes and browsing or comparing value sets. Specializations of the *Terminology User* actor follow below.

- **Terminology Administrator**

The *Terminology Administrator* is an actor responsible for ensuring the availability and overall maintenance of the terminology server. This includes, but is not limited to loading content into the terminology server, and making available the required functionality to address the specific conformance profiles implemented by the Terminology Server instance.

- **Terminology Enabled Application Developer**

A *Terminology Enabled Application Developer* is an actor who is responsible for the development of software applications that make explicit use of controlled terminologies.

- **Terminology Author / Curator**

A *Terminology Author / Curator* is an actor who is responsible maintaining terminology content, including but not limited to, the development of new concepts that may be submitted to the *Terminology Provider* or the extension of an existing terminology with local concepts. This may also who can validation and quality control of terminology content. Terminology Authors / Curators may not necessarily belong to the *Terminology Provider's* organization.

- **Terminology Human Language Translator**

A Terminology Human Language Translator is an actor with domain knowledge who is also familiar with the languages and dialects which they are responsible for translating.

- **Terminology Mapper**

A *Terminology Mapper* is an actor (human or system) that is responsible for creating or maintaining specialized associations, or "mappings" between concepts from different code systems.

- **Terminology Provider**

The *Terminology Provider* is the actor the individuals or organization that is responsible for the development of Terminology Content. ion of the *Terminology User* actor.

- **Terminology Value Set Developer**

A *Terminology Value Set Developer* is an actor with specific domain knowledge, as well as expertise in controlled terminologies who develops and maintains domain-or application-specific terminology value sets.

Primary Scenarios

Primary scenarios are tied to one or more conformance profiles. Note, that as an aid to reading this specification, Actors that are identified in the text are italicized. In addition, when a scenario references another scenario, that referenced scenario is in bolded italics.

Administrative Scenarios

The administration scenarios are intended to provide the functional operations necessary for terminology administrators to be able to access and make available terminology content obtained from a Terminology Provider. Terminology Administrators are required to interface with Terminology Provider systems in order to obtain the terminology content, then load that terminology content on local Terminology Servers.

Import Content

A *Terminology Administrator* is required to make available new terminology content from a *Terminology Provider* available to *Terminology Users* through a *Terminology Server*. This may or may not include the removal of a previously loaded terminology content from the terminology server. To accomplish this, the *Terminology Administrator* may be required to convert the content from the format provided by the *Terminology Provider* to a format that the *Terminology Server* is capable of importing (the conversion being out of scope of this service). Example of terminology content that may be available for loading into the *Terminology Server* include but is not limited to:

- Source terminologies (complete sources and deltas)
- Value sets

These content sources may either be new sources, or updated versions of an previously existing content sources.

Associated Functional Models: Import Terminology, Import Terminology Revision

Export Content

A *Terminology Administrator* wants to is required to export a terminology or terminology subset from the *Terminology Server*. This may require filtering of the exported content. In cases where standard or common terminology interchange formats are available (such as HL7 Vocabulary MIF or LexGrid), conversion of content on the CTS 2 server into said standard would be considered.

Associated Functional Models: Export Terminology

Remove Content

A *Terminology Administrator* is required to remove a terminology or terminology version from the terminology service, rendering it unavailable for subsequent access by other service functions.

Associated Functional Models: Change_Terminology_Status

Change Content Status

A *Terminology Administrator* is required to activate or inactivate a given terminology, thus changing its availability for access by other terminology service functions.

Associated Functional Models: Change Terminology Status

Update Notification

A *Terminology User* has a dependency on a specific terminology element that is available to a Terminology Server. The *Terminology User* is interested in knowing when this terminology element is modified in any way, and would like to receive an electronic notification in the event of that change to that terminology element Associated Functional Models: Register for Notification

Update Notification Management

A *Terminology User* is required to update the notification information pertinent to their notification account.

Associated Functional Models: Update Notification Registration, Update Notification Registration Status

Content Dependency Notification

A *Terminology Administrator* is required to run a dependency check to compare updated content for a given code system, against the version of that code system currently used by the *Terminology Administrator's* organization. For example, to provide a list of all terminology elements which are somehow affected by upgrading to a newer version of a terminology Associated Functional Models: Register for Notification, Update Notification Registration, Update Notification Registration Status

Search / Query Scenarios

The scenarios in this section describe the ability to query code system and value set. These scenarios attempt to outline the information requirements for querying. The detailed function models in section 5.2 call out the distinct functional requirements.

A given CTS 2 implementation will be required to advertise the specific search algorithms that is supports.

In each scenario below, the *Terminology User* may need to specify additional information pertaining to the query. This information may include:

- The ability to determine the status of metadata or contents of a code system, value set as it existed in a specified *version*, where *version* represents a meta-data component used to filter the result set of the query.

NOTE: Details of the available meta-data requirements will be identified as part of the Binding Document and Model harmonization activity.

Code System Search / Query

This section outlines Search / Query operations pertaining to Code Systems.

Resolve Available Code Systems

A *Terminology User* wants to determine what code systems are available through a specific instance of a Terminology Service. The *Terminology User* is interested in seeing a listing of the available code systems, as well as the details pertaining to each code systems available through a specific Terminology Service instance.

Associated Functional Models: List Code Systems, Return Code System Definition

Retrieve Coded Concepts from Code System

A *Terminology User* wants to browse or query the content of a specific code system. The *Terminology User* is interested in seeing a listing of specific coded concepts, associated attributes, as well as the metadata pertaining to each coded concept that meets some search criteria. For example, after a retrieval of concepts has been performed, the result set could be fed to a terminology browsing GUI

Associated Functional Models: List Code System Concepts, Return Coded Concept Definition

Validate Concept in Code System

A *Terminology User* wants to validate that a given concept exists in a given code system.

Associated Functional Models: Return Coded Concept Definition

Identify Concept Language Translations

A *Terminology User* wants to determine what (if any) alternate language representations exist for a given Concept.

Associated Functional Models: Return Coded Concept Definition

Resolve Concept Representations

A *Terminology User* wants to determine what (if any) alternate representations exist for a given Coded Concept. Examples of alternate representations for a concept may include abbreviations, or synonyms.

Associated Functional Models: Return Coded Concept Definition

Compare Code System Versions

A *Terminology User* wants to determine what differences exist between different versions or instances of a code system.

Note that the Service calls just return the required Code systems and concept information. Carrying out the side by side comparison would be a client function.

Associated Functional Models: Return Code System Definition, List Code System Concepts

Value Set and Concept Domain Search / Query

This section outlines Search / Query operations pertaining to Value Sets and Concept Domains.

Resolve Available Value Sets

A *Terminology User* wants to determine what value sets are available through a specific instance of a Terminology Service. The *Terminology User* is interested in seeing a listing of the available value sets that match some search criteria, as well as the details pertaining to each value set available through a specific *CTS 2 Service* instance.

Associated Functional Models: List Value Sets, Return Value Set Definition

Retrieve Coded Concepts from Value Set

A *Terminology User* wants to browse or query the content of one or more value sets. The *Terminology User* is interested in seeing a listing of specific coded concepts, as well as the details pertaining to each coded concept in any of the given value sets. For example, the *Terminology User* may want to search for some criteria over a set of value sets.

Associated Functional Models: List Value Set Contents, Return Coded Concept Definition

Resolve Available Concept Domains

A *Terminology User* wants to determine what Concept Domains and Usage Contexts are available through a specific instance of a Terminology Service. The *Terminology User* is interested in seeing a listing of the available concept domains that match some search criteria.

Associated Functional Models: List Concept Domains, Return Concept Domain Definition, List Usage Contexts, Return Usage Context Definition

Retrieve Coded Concepts for Concept Domain

A *Terminology User* wants to browse or query the content of one or more value sets and contained concept codes that are related to a specific Concept Domain (and optionally Usage Context). The *Terminology User* is interested in seeing a listing of specific coded concepts, as well as the details pertaining to each coded concept in any of the given Concept Domains. (Note that this involves finding the value set bindings and then querying the value sets, although the service should hide that from the user to some degree).

Associated Functional Models: List Concept Domain Bindings, Return Coded Concept Definition, Return Usage Context Definition, List Value Set Contents

Validate Coded Concept in Value Set or Concept Domain

A *Terminology User* wants to validate that a given concept exists in a given value set or Concept Domain and Usage Context.

Associated Functional Models: Check Concept Value Set Membership, Check Concept Domain Membership

Compare Value Set Versions

A *Terminology User* wants to determine what differences exist between different versions of a value set.

Value Sets can be defined as either enumerations of concepts (Enumerated Value Set), or by expression syntax that defines the content of the Value Set.

In the case of an Enumerated Value Set, the specific Value Set version identifier can be used as a compare point for the two value sets.

For Intensionally defined Value Sets, the compare point is either the Code System version when the Value Set definition is bound to a specific Code System version, or the date when the Value Set definition is bound to a code system with no specific version specified.

Note that the Service calls just return the required Value Set information. Carrying out the side by side comparison would be a client function.

Associated Functional Models: Return Value Set Definition, List Value Set Contents

Resolve Concept Representations

A *Terminology User* wants to determine what (if any) alternate representations exist for a given Coded Concept in a value set. Examples of alternate representations for a concept may include abbreviations, or synonyms.

Associated Functional Models: List Value Set Contents

Authoring / Curation Scenarios

This section outlines the requirements of terminology systems that provide the capability of making changes to terminology elements such as code system or value sets. This includes both the direct modification of terminology content for use by individuals responsible for terminology authoring and curation. Such functionality includes:

- adding new concepts into a code system
- adding new relationships into a code system
- extending a code system with local terms
- creating or modifying value sets
- modifying other code system content and attributes

In addition to direct modification of terminology content, this section also specifies functionality with the capability of creating structured change requests for consideration by terminology maintainers. This functionality is key in allowing Terminology Providers to solicit feedback pertaining to terminology structure and content from Terminology Users in a controlled and structured manner.

Code System Authoring / Curation

This section outlines the business scenarios specific to terminology systems that provide the capability of making changes to code system components which include coded concepts, representations (textual), Associations or Relationships, and value sets.

Create Code System

A *Terminology Author* is required to create a new Code System to contain a set of new coded concepts. The Code System is created by defining the set of meta-data properties that describe it.

Associated Functional Models: Create Code System

Maintain Code System

As part of ongoing terminology maintenance, a *Terminology Author* is required to perform maintenance to the defining characteristics of an existing code system.

Associated Functional Models: Maintain Code System, Update Code System Version Status

Create Concept

A *Terminology Author* is required to create a concept to be included in a Code System.

For example, as part of providing *Terminology Service* infrastructure to another department, a *Terminology Author* is required to add additional concept codes to a code system to represent the domain concepts that are important to the new department.

The new concept is defined by the set of meta-data properties that describe it, which may include its proper placement via association binding within the hierarchy of the Code System.

Associated Functional Models: Create Concept

Maintain Concept

A *Terminology Author* is required to maintain a concept. This includes but is not limited to functionality such as:

- making updates to the associated concept attributes,
- changing the presentation,
- changing preferred name,
- changing synonymy,
- technical corrections to the concept
- modifying the associations bound to concepts
- deprecating a concept
- deprecating a concept and superseding it with another concept

These types of changes result in a new version of the of the code system being modified.

Associated Functional Models: Maintain Concept, Update Concept Status

Value Set Authoring / Curation

This section outlines the business scenarios specific to terminology systems that provide the capability of creating and maintaining sub-sets of codes residing in code systems on CTS 2 servers, otherwise known as value sets.

Create Value Set by Intension

A *Terminology User* is required to create a dynamic value set that is defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time.

For example, an intensional value set might be expressed as, SNOMED CT concepts that are children of the SNOMED CT concept “Diabetes Mellitus.”

Note: When creating an intensionally defined value set, the *Terminology User* may or may not bind the value set definition to a specific version of the Code System(s) from which the concepts are being drawn.

If the value set expression is bound to a specific version of the Code System(s), the value set will always resolve the same set of concept codes for any given version of the value set.

If the value set expression is **not** bound to a specific version of the Code System(s), the value set will resolve a different set of concept codes as the version of the Code System changes.

Implicit definitions can specify:

- All active unique identifiers from a given coding system.
- All unique identifiers that participate in a specified relationship with a given local code in a coding system, which may or may not include the specified code itself.
- The transitive closure of a specified transitive relationship with a given code, including or excluding the code itself.
- A reference to another value set
- Other mechanisms that have to have external human or computational resolution.
- Unions, intersections and exclusions of any of the above
- Nested value set definition in which a value set entry references another value set (a child value set). There is no preset limit to the level of nesting allowed within value sets. Value sets cannot contain themselves, or any of their ancestors (i.e., they cannot be defined recursively). Nested value sets are always intensionally defined in HL7.

Associated Functional Models: Create Value Set by Intension

Create Value Set by Extension

A *Terminology User* is required to create an enumerated (static) value set that is comprised of an explicitly enumerated set of codes.

For example, A *Terminology Author* is interested in creating a value set based on the SNOMED-CT code system. The *Terminology Author* builds the value set by selecting the individual concepts that best represent the concepts that are required for the value set.

Associated Functional Models: Create Value Set by Extension

Maintain Value Set (Definition)

A *Terminology User* is required to maintain (i.e. remove or update) a value set (by definition).

For example, a *Terminology User* identifies an error in how a value set (by definition) is defined. A *Terminology Author* re-defines the value set to be accurate to the understanding of the *Terminology User*.

Associated Functional Models: Maintain Value Set (Intension), Update Value Set Status

Maintain Value Set (Enumeration)

A *Terminology User* is required to maintain (i.e. add, remove or update) an enumerated coded concept in value set.

For example, a *Terminology User* identifies a concept code that is not included in an enumerated value set. The *Terminology Author* browses the code system to select the concept code identified by the *Terminology User*. The *Terminology Author* selects the concept code to be included in the existing value set

Note: In order to create a coded concept in a value set it must first exist in the code system.

Associated Functional Models: Maintain Value Set (Extension), Update Value Set Status

Concept Domain and Usage Context Authoring / Curation

This section outlines the business scenarios specific to terminology systems that provide the capability of creating and maintaining Concept Domains and associated Usage Contexts, which provide a further level of reusability for Value Sets.

Create Concept Domain

A *Terminology Author* is required to create a Concept Domain.

Associated Functional Models: Create Concept Domain

Create Usage Context

A *Terminology User* is required to create a Usage Context within a Jurisdictional Domain. (It is assumed that Jurisdictional Domains themselves would be created as part of system configuration.

Associated Functional Models: Create Usage Context

Maintain Concept Domain

A *Terminology Author* is required to maintain a concept domain, including the ability for a *Terminology User* to identify the value set bindings for specific usage contexts.

Associated Functional Models: Maintain Concept Domain

Maintain Usage Context

A *Terminology User* is required to maintain a Usage Context.

Associated Functional Models: Maintain Usage Context

Change Request Processing

Although Change Request processing is within the overall process of Terminology authoring and management, it is out of scope of the Terminology Service. There are many existing systems that manage change request processes.

Association Scenarios

The scenarios in this section describe the ability to create, query and maintain associations between coded concepts. These coded concepts may or may not come from the same Code System. As such, these scenarios can describe intra-code system associations, or concept relationships, as well as inter-code system associations across different systems (aka concept maps). The premise for this is that behavioral and information requirements and functions for concept relationships and concept mapping are similar, although the context of use and potentially some information elements for each are different. These scenarios attempt to outline the information requirements for associations. Since the behavior is similar for both types of association, a single set of operations are defined. Where an information model distinction exists, specific semantic profiles could be defined. In each scenario below, the Terminology Mapper may need to specify additional information pertaining to the source / target association of interest. This information may include:

- The version of the source and target Code Systems being used to create the association, or,
- The code system of interest, whether that pertains to a single code system or more than one code system
- The version of the source and target code systems being used to create the association, or,
- The cardinality of the association, i.e.: if the concept association is one-to-one, one-to-many, many-to-one, or many-to-many.

Additionally, the type of associations may include, but are not limited to:

- if the source concept is an *exact match* to the target concept,
- if the source concept is *equivalent* to the target concept,
- if the source concept is *broader than* the target concept,
- if the source concept is *narrower than* the target concept
- Other examples are *generic-to-brand name*, *ingredient-variant-of*, etc.

Association Administrative Scenarios

Enumerate Code System Coded Concept Relationship Types

A *Terminology User* wants to determine the set of concept relationship types that are available on a terminology server.

Associated Functional Models: List Concept Relationship Types, Return Concept Relationship Type Definition

Identify / Retrieve Concept Associations for a Single Concept

A *Terminology User* wants to identify all the associations that exist for a given concept. This includes both direct and indirect relationships, and may be depth limited where appropriate. This includes concept relationships (associations for the concept that are within its native code system) or concept maps (associations between the specified concept code system and another code system) or both. Returns a set of triples: the source, the target and the association

Associated Functional Models: List Concept Relationships, Return Concept Relationship Definition

Identify / Retrieve Associations Between Two or More Coded Concepts

A *Terminology User* is required to provide a listing of the concept associations that exist between coded concepts. For example, these associations may be required as part of a government regulatory compliance review or audit.

This includes concept relationships (associations for the concept that are within its native code system) or concept maps (associations between the specified concept code system and another code system) or both. Returns a set of triples: the source, the target and the association.

Associated Functional Models: List Concept Relationships, Return Concept Relationship Definition

Import Coded Concept Associations

A *Terminology User* is required to make new coded concept associations available through a Terminology Server. This may or may not include the removal of previously loaded coded concept associations from the terminology server. To accomplish this, the Terminology User may be required to convert the content from the format to a format that the Terminology Server is capable of importing.

Associated Functional Models: Import Terminology, Import Terminology Revision

Export Coded Concept Associations

A *Terminology User* wants to export coded concept associations from the Terminology Server. This may require filtering of the content and converting the format of the export.

Associated Functional Models: Export Terminology

Remove Coded Concept Associations

A *Terminology User* is required to remove coded concept associations or coded concept association versions from the terminology service, rendering them unavailable for subsequent access by other service functions.

Associated Functional Models: Update Concept Relationship Status

Change Status of Coded Concept Associations

A *Terminology User* is required to activate or inactivate coded concept associations, thus changing their availability for access by other terminology service functions.

Associated Functional Models: Update Concept Relationship Status

Register for Association Update Notification

A *Terminology User* wants to receive notification that an element of an association has changed and thus may require review.

Associated Functional Models: Register For Notification, Update Notification Registration, Update Notification Registration Status

Compute Transitive Closure

A Terminology User

Reasoning / Subsumption Trace

A Terminology User

Association Search / Query Scenarios

This section outlines Search / Query operations specific to associations and association content.

Resolve Available Associations

A *Terminology User* wants to determine what associations are available on the terminology service by browsing a list of available associations on the CTS 2 instance. The service differentiates between coded concept relationships and coded concept maps available for any specified concept.

Associated Functional Models: List Concept Relationships

Validate Associations

A *Terminology User* wants to validate that a given association or set of associations are available on the CTS 2 service instance based upon specific search criteria.

Associated Functional Models: List Concept Relationships

Retrieve Association Metadata

A *Terminology User* wants to retrieve metadata on available associations in the CTS 2 service instance. This may include metadata regarding the code system(s) employed, versions, authoring / curation content or additional data hosted on the CTS server designated to be used by external systems (i.e.: XML encoded or OWL formatted mapping rule content).

Associated Functional Models: Return Concept Relationship Definition

Compare Association Versions

A *Terminology User* wants to compare two or more versions of an association on a CTS 2 service instance by viewing each association version's identifying information or metadata. Retrieval is provided by the service, the actual comparison will be the responsibility of the service client application.

Associated Functional Models: Return Concept Relationship Definition

Request / Retrieve Association Instance

A *Terminology User* would like to request or retrieve an association when the metadata for such is retrieved and viewed from a CTS 2 instance.

Associated Functional Models: Return Concept Relationship Definition

Association Author / Curation Scenarios

Create / Maintain an Association between Coded Concepts

A *Terminology User* wants to create or maintain (i.e. remove or update) an association between coded concepts. For example, these associations may be required to map a local Code System to standard Code Systems in order to be compliant with regulatory reporting policies.

(Note that only "create" operations have been identified, since it is viewed that any changes to a relationship definition (other than state changes, which are included in a separate operation) are in fact definitions of new relationships (i.e. the rules for the relationship define the relationship, and are not something separate or merely properties. It is possible that the technical specification may introduce update operations, but at this level would not add meaning)

Search criteria may be accompanied by a "match algorithm code" that determines how the search text will be applied. The table below provides an example set of match algorithms. **NOTE:** This match algorithm list is not exhaustive. It is permissible for service implementations to extend the list below with additional, custom match algorithms as appropriate, although implementers are strongly encouraged to register the algorithm code to ensure interoperability.

Match Algorithm Code	Description
IdenticalIgnoreCase	The lower case representation of the target text must match the lower case representation matchText exacty.
Identical	The target text must match the matchText exactly.
StartsWithIgnoreCase	The lower case representation of target text must begin with the lower case representation of matchText.
StartsWith	The target text must begin with the matchText.
EndsWithIgnoreCase	The lower case representation of the target text must end with the lower case representation of matchText.
EndsWith	The target text must end with the matchText.
ContainsPhraseIgnoreCase	The lower case representation of the target text must contain the lower case representation of the matchText.
ContainsPhrase	The target text must contain the matchText.
WordsAnyOrderIgnoreCase	The target text must contain all of the words in the match text, but in any order.
WildCardsIgnoreCase	The match text may contain zero or more 'wild cards', designated by an asterisk (*). Wild cards match 0 of more characters in the target string. The escape character is a backslash('\') meaning that the matchText "a*b*' would match any string that begins with the string "a*b".
RegularExpression	The match text may contain regular expressions, as defined in XML Schema Part 2: Datatypes.
NYSIIS	New York State Identification and Intelligence System phonetic encoding

Associated Functional Models: Create Concept Relationship

Create Relationship Type

A *Terminology Author* is required to create a new relationship type that may be used to link two concepts.

Associated Functional Models: Create Concept Relationship Type

Maintain Relationship Type

A *Terminology Author* is required to modify or deprecate a relationship type that may be used to link two concepts.

Associated Functional Models: Maintain Concept Relationship Type

Create Lexical Association

A *Terminology User* wants to instantiate an association between two sets of coded concepts using a set of lexical rules (matching algorithms) to generate the associations.

Note: This operation is likely not an automated process, as many terminologies do not possess the ontological structures necessary to instantiate additional associations automatically. This however should not preclude the automated creation of associations for terminologies with the ontological infrastructure necessary to enable reasoning engines to effectively instantiate new associations.

Associated Functional Models: Create Lexical Relationship Between Coded Concepts

Create Rules Based Association

A *Terminology User* wants to instantiate an association between two sets of coded concepts using a set of description logic or inference rules that either assert or infer mappings between two Code Systems.

NOTE: This operation is likely not an automated process, as many terminologies do not possess the ontological structures necessary to instantiate additional associations automatically. This however should not preclude the automated creation of associations for terminologies with the ontological infrastructure necessary to enable reasoning engines to effectively instantiate new associations. These associations may be subject to human review to verify validity.

Associated Functional Models: Create Rules Based Relationship Between Coded Concepts

Assumptions and Dependencies

Dependencies on other Service Frameworks

As a service specification, the original CTS specified service discovery APIs. We assume that for CTS 2 a service discovery framework is available to aid with discovery and query of the CTS 2 service, and that the service is queriable by the common service metadata attributes outlined in the Service Discovery framework, in addition to terminology service specific metadata outlined in section 7.

CTS 2 offers a robust set of API requirements, many of which should be restricted to specific user classes. This specification outlines a set of functional profiles that specify the types of operations users may perform as part of a profile. CTS 2 assumes that security, identity management, and auditing services are available that can implement the necessary user role based access requirements outlined in section 6.

CTS Backwards Compatibility

Message API Support (MAPI)

In the original CTS, the Message API component is specific to HL7. Its primary purpose is to allow a wide variety of message processing applications to create, validate and translate CD-derived data types in a consistent and reproducible fashion. It is assumed that this level of functionality will remain specific to HL7, and as such will me managed by developing a profile specific to HL7.

General CTS API Support

Unless otherwise indicated, it is assumed that CTS 2 provides the functional coverage required for backwards compatibility to CTS. It is assumed that areas where CTS 2 compatibility with CTS will vary include areas such as:

- HL7 Datatypes - Where the version of the datatypes has been updated since CTS was developed.
- Service Discovery - The CTS service discovery APIs are no longer needed assuming the existence of Service Discovery infrastructure.
- Separating MAPI APIs into an HL7 specific terminology profile.

HL7 Datatypes

As an HL7 specification, CTS 2 will make use of the HL7 Datatypes where possible. Recognizing that they HL7 Datatypes are an evolving standard, CTS 2 technical implementations of CTS 2 will be required to indicate what version of the HL7 Datatypes that are currently implemented. This restricts complete backward compatibility to the original CTS, as CTS 2 implementations will be implementing a more current version of the HL7 Datatypes.

In the event that the development of the CTS 2 Technical Specification identifies gaps in the HL7 Datatypes, CTS 2 will specify its requirements and feed those requirements to HL7 for inclusion in the datatypes specification.

Considerations for Technical Specification

Functional Overloading

CTS 2 services need to support terminologies with very different designs. Some terminologies are well designed, whereas some others are not. The terminologies that are well designed have concept uniqueness, concept permanence, unique identifiers, formal definitions, and track history well supporting a 'graceful evolution'.

Terminologies that are not well designed lack one or more of these good design practices, and need additional modifications in the functional definitions of CTS 2 functions. For example, terminologies that reuse concept identifiers among different domains (e.g. 'M' may mean male, million or meter) need the domain identifier in addition to the concept identifier to uniquely identify a concept. For example, in the conceptual model above, and additional Concept Identifier has been included in addition to the Concept Code itself (which in well behaved systems should be unique)

The objective of CTS 2 is to support various terminologies within a single terminology service, and not to standardize terminology design. However, to support terminologies with varying designs, different semantic profiles will need to be used to cope with some of these variances. In some cases, the CTS2 functions have to be include additional optional input parameters (for the terminologies with non-standard designs) to return the output. Thus, some of the input parameters of a given CTS 2 function vary based on the design of the terminology that is being queried. The CTS 2 function will have required and optional input parameters - in general the required parameters apply to all terminologies, but the optional parameters apply only to some terminologies with non-standard designs.

However, specifying input parameters as just required or optional can lead to confusion in implementation, which may lead to difficulty for the user to understand which combinations of optional parameters are to be used for a specific terminology. As the number of optional parameters for a function increases linearly, the number of combinations of input parameters can increase exponentially.

For a function with n optional input parameters, up to 2^n combinations of input parameters are possible. It then becomes hard for the implementer or user to decipher which

combinations are valid and which combinations should be used for different terminologies. In reality, most of these combinations are invalid and cannot be used. In addition, such a function is hard to automate, and requires human intervention to specify the parameters that are applicable to each terminology. Thus, specifying required and optional parameters alone can lead to unnecessary and confusing combinatorial explosion and may lend poorly to automation.

There are several solutions that are possible to this problem. One is the use of different semantic profiles for different terminologies (or classes of terminologies - see further discussion below), as long as the behavior of the operations is not affected (other than validation of mandatory and optional inputs).

Another is 'function overloading'. This is similar to overloading a method in Object Oriented Programming (OOP). A given method in OOP may have many overloaded variants - each with different inputs, but all performing the same basic function and returning the same output. When the technical specification is produced, the resolution to this issue must be defined explicitly. Functional overloading requires more effort to create the functional variants, but this reduces the combinatorial explosion of optional input parameters, avoids creation of invalid combinations of input parameters, and provides a solution to tie a specific terminology to a specific variant for each function. This also lends better to automation than specifying the input parameters as just optional or required. Functional overloading still cannot be automated at runtime, because we need a way to tie a specific variant of a function to a specific terminology. This is achieved through a metadata discovery service.

Metadata Discovery

The metadata discovery service helps the calling program to discover the available variants of a CTS 2 function, the input parameters required for each function, and the terminologies that each variant applies to. For example, a function that returns the descriptions of a given concept will require the concept identifier as a coded datatype (ConceptCode, CodeSystemId) as the input. However, if a given terminology reuses the same concept code to represent what are effectively different concepts, a way to disambiguate them is needed. The conceptual model provides a separate unique identifier, however an additional user friendly mechanism is needed. At the level of the code system itself, as well as the identifier, the code name/description will allow a human user to differentiate them, e.g. if "m" is used for "male" and "meter" in the same code system. Over and above that, when using value sets and binding information model instances to them, the "concept domain" may be used in to enable clear distinction in these cases. The metadata discovery service will cover all of these variants.

However, creating a relationship between a given terminology and a functional variant can be daunting given the number of CTS 2 functions and the number of terminologies supported. This can be overcome by grouping the terminologies together based on common design and structural characteristics. This is achieved by using semantic profiles. A given semantic profile groups together terminologies with similar designs. Many such semantic profiles are thus possible. The metadata discovery function will list the variants applicable to different semantic profiles, rather than different terminologies.

The CTS 2 authors define the semantic profiles as a set of design characteristics, and assign the better known terminologies into these profiles. At the technical specification/implementation level, the CTS 2 functions will then have overloaded variants based on common design characteristics, rather than those based on individual terminologies. The metadata discovery service will provide the available variants of each CTS 2 function and the different semantic profiles they apply to (rather than the different terminologies).

Functions that do not have overloaded variants will just have the single (default) variant returned by the metadata discovery service. Terminologies that are not yet classified need to be classified into a semantic profile. This is discussed in detail under the Semantic Profiles section.

By using functional overloading, metadata discovery and semantic profiles together, the combinatorial explosion and invalid combinations are avoided, the ambiguity and the amount of effort required are reduced, and automation is made easier.

Detailed Functional Model for each Interface

Administration Functions

Import Terminology

Description	Installs a terminology into the terminology service for subsequent access by other service functions. This operation is used for the initial install of the overall terminology structure itself. This may include the full set of concepts, relationships and so on, or some of these elements may be loaded using the Import Terminology Revision operation.
Inputs	1. Code System Source ID 2. Code System ID 3. Code System Version 4. Code System Source Location (URI) OR 1. Code System Contents
Outputs	1. An acknowledgment indicating whether the terminology has been successfully loaded or not.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Terminology source is available in a format directly consumable by CTS 2 import tools.
Postconditions	1. The terminology is available for access via the CTS 2 service functions.
Exception Conditions	1. Terminology source is not consumable by CTS 2 import tools. 2. Terminology not found at source location 3. Supplied contents cannot be processed (may be a number of specific errors) (Information pertaining to all failures is logged and reported for analysis and serviceability.)
Aspects left to Technical Specification	
Relationship to	

levels of conformance	Administration
Miscellaneous notes	
Other relevant content	
Associated Scenario	Import Content, Import Coded Concept Associations

Import Terminology Revision

Description	Installs either an entire new version or the necessary revision updates for an already loaded terminology into the terminology server repository (content included by value or by reference to a location). Includes indicator as to whether intent is to replace whole terminology or just replace some elements (codes, relationships etc).
Inputs	1. Code System Revision Source ID 2. Code System ID 3. Current Code System Version 4. New Code System Version 5. Scope of Replacement (encoded and description) 6. Code System Location (URI) .. OR .. 7. Updated Contents
Outputs	1. An acknowledgment indicating whether the terminology revision has been successfully loaded or not.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Existing Terminology is loaded into the terminology service. 3. Existing Terminology must be active. 4. Terminology revision source is available in a format directly consumable by CTS 2 import tools
Postconditions	1. The terminology revision is available for access via the CTS 2 service functions.
Exception Conditions	1. Terminology does not exist. 2. Terminology is not active. 3. Terminology source is not consumable by CTS 2 import tools. 4. Terminology not found at source location 5. Supplied contents cannot be processed (may be many different error conditions) (Information pertaining to the failures is logged and reported for analysis and serviceability.)
Aspects left to Technical Specification	Detailed specification of the mechanism to recognize the specific information structures that may be imported.
Relationship to levels of conformance	Administration
Miscellaneous notes	Terminology revisions may be available as either: 1. complete code systems 2. set of deltas to be applied sequentially to the previous version. In either case, all previous versions/iterations should be available until specifically removed.
Other relevant content	Depending on the scope of the actual import, many different information structures may be included. One particular area (concept relationships is expanded in more detail below. This would include elements such as: 1. Relationship Identifier (if known) 2. Relationship Description 3. Relationship Source Code System Id and Concept Code 4. Relationship Target Code System Id and Concept Code 5. Relationship Type 6. Relationship Restrictions 7. Relationship Cardinality 8. Relationship Group 9. Relationship Order 10. Relationship is Reciprocal 11. Relationship is Refinable 12. Relationship is Transitive 13. Relationship is Cyclic 14. Relationship is Inheritable 15. Relationship Curation / Authoring Information 16. External systems coded concept relationship data hosted on the CTS server (e.g. XML encoded or OWL formatted mapping rule content).
Associated Scenario	Import Content, Import Coded Concept Associations

Export Terminology

Description	Exports a terminology, terminology subset or map from the Terminology Server by filtering the content and converting to the requested format for export.
Inputs	1. Code System ID 2. Code System Version 3. Code System subset criteria 4. Code System map 5. Export Format 6. Target Location (URI)
Outputs	1. Code System Location (URI) or 1. Code System Content (depending on whether the return will be the actual content or a location in which the content is placed for subsequent retrieval.)
Invariants	
Precondition	1. CTS 2 Service installed and running 2. Existing Terminology is loaded into the terminology service 3. Existing Terminology must be active 4. Terminology source is available for export
Postconditions	1. The terminology is exported to the required location or is returned to the requestor
Exception Conditions	1. Terminology does not exist. 2. Terminology is not active. 3. Invalid criteria/format input 4. Unrecognized location input 5. Terminology source is not exportable by CTS 2 export tools.
Aspects left to Technical Specification	
Relationship to levels of conformance	Administration
Miscellaneous notes	
Other relevant content	
Associated Scenario	Export Content

Change Terminology Status

Description	Changes the state of a code system. (Includes inactivation, activation, removal etc.) This allows a *Terminology Administrator* to manage the state of a given terminology, thus managing the level of availability for access by other terminology service functions.
Inputs	1. Code system identifier 2. Code system version 3. Code system status
Outputs	1. An acknowledgement indicating whether the state of the source terminology has been successfully changed.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service.
Postconditions	1. The terminology source state is changed to the requested value.
Exception Conditions	1. Terminology does not exist 2. Terminology state transition not allowed
Aspects left to Technical Specification	1. Definition of actual state values / model
Relationship to levels of conformance	Administration
Miscellaneous notes	
Other relevant content	
Associated Scenario	Change Content Status, Remove Content

Register for Notification

Description	Register to be notified whenever a vocabulary element (code system, value set, concept or relationship) is modified in any way. The Notification Directions parameter is a placeholder for allowing specific directions to be given for the notification (e.g. whether it is required immediately or may be delayed)
Inputs	1. URL or other electronic address which to send the terminology element modification notification to. 2. Notification Target Type (Code System, Value Set, Concept, Concept Relationship) 3. Notification Target Identifier 4. Notification Target Version 5. Notification Directions
Outputs	1. An acknowledgement indicating whether the terminology element notification request was successfully created. 2. Notification Identifier
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Identified element must be present in the terminology service. 3. User or appropriate proxy (system administrator, etc) are authorized to access registry
Postconditions	1. Notification records are updated appropriately
Exception Conditions	1. Identified element / version does not exist 2. Notification Directions not recognized
Aspects left to Technical Specification	
Relationship to levels of conformance	Administration
Miscellaneous notes	Subsequent notifications do not require a confirmation. Where appropriate, however, negative feedback on the channel (unable to deliver message, unable to connect), should result in attempts to retransmit and/or the placement of a temporary hold on notifications until connection problem is corrected.
Other relevant content	
Associated Scenario	Update Notification, Content Dependency Notification

Update Notification Registration

Description	Revises a notification entry for a particular vocabulary element.
Inputs	1. Notification Entry Identifier 2. URL or other electronic address which to send the terminology element modification notification to. 3. Notification Target Type (Code System, Value Set, Concept, Concept Relationship) 4. Notification Target Identifier 5. Notification Target Version 6. Notification Directions
Outputs	1. An acknowledgment indicating whether the terminology element notification revision request was successfully processed.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Notification Entry exists. 3. Identified element must be present in the terminology service. 4. User or appropriate proxy (system administrator, etc) are authorized to access registry
Postconditions	1. Notification records are updated appropriately.
Exception Conditions	1. Notification Entry Identifier does not exist. 2. Identified element / version does not exist 3. Notification Directions not recognized
Aspects left to Technical Specification	
Relationship to levels of conformance	Administration

Miscellaneous notes	Subsequent notifications do not require a confirmation. Where appropriate, however, negative feedback on the channel (unable to deliver message, unable to connect), should result in attempts to retransmit and/or the placement of a temporary hold on notifications until connection problem is corrected.
Other relevant content	
Associated Scenario	Update Notification Management, Content Dependency Notification

Update Notification Registration Status

Description	Changes the status of a notification entry for a particular vocabulary element (suspend, reinstate, cancel, remove).
Inputs	1. Notification Entry Identifier 2. Notification Status
Outputs	1. An acknowledgment indicating whether the terminology element notification status revision request was successfully processed.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Notification Entry exists. 3. User or appropriate proxy (system administrator, etc) are authorized to access registry
Postconditions	1. Notification status is updated appropriately.
Exception Conditions	1. Notification Entry Identifier does not exist. 2. Invalid state change
Aspects left to Technical Specification	
Relationship to levels of conformance	Administration
Miscellaneous notes	
Other relevant content	
Associated Scenario	Update Notification Management, Content Dependency Notification

Search / Access

Code System Search / Access

List Code Systems

Description	List the code systems available on this instance of the CTS 2 Service that match entered filter criteria.
Inputs	1. Code System Filter Criteria
Outputs	1. A listing of zero or more code systems, together with metadata properties available on the instance of the terminology service.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	None.
Exception Conditions	1. Invalid filter criteria
Aspects left to Technical Specification	Definition of filter criteria. Should include name for a simple means of retrieving ID where not known.
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Resolve Available Code Systems

Return Code System Details

Description	Returns the details (metadata) for a given code system available on the terminology service
Inputs	1. Code System Identifier 2. Code System Version
Outputs	1. Details pertaining to a specific code system (metadata or attributes for the code system, e.g. description)

Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service.
Postconditions	None.
Exception Conditions	1. Code System Identifier not found. 2. Code System version not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Resolve Available Code Systems, Compare Code System Versions

List Code System Concepts

Description	Returns the set of all concepts in the specified code system, optionally filtered by input criteria, such as state or specific concept property values.
Inputs	1. Code System Identifier 2. Code System Version 3. Filter Criteria
Outputs	1. The set of zero or more concepts in the specified code system matching the filter criteria.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service.
Postconditions	None.
Exception Conditions	1. Code System Identifier not found. 2. Code System version not found. 3. Invalid filter criteria
Aspects left to Technical Specification	Definition of filter criteria
Relationship to levels of conformance	Query/Search
Miscellaneous notes	Returning all concepts in a code system is generally impractical for large code sets. Indexing, query optimization is necessary.
Other relevant content	
Associated Scenario	Retrieve Coded Concepts from Code System, Compare Code System Versions

Return Coded Concept Details

Description	Returns the details for the known attributes (metadata) of a coded concept
Inputs	1. Code System Identifier 2. Code System Version 3. Coded Concept Identifier 4. Coded Concept Version
Outputs	1. The details of the attributes (metadata) of the coded concept
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service. 3. Coded concept must exist.
Postconditions	None.
Exception Conditions	1. Code System Identifier not found. 2. Code System Version not found. 3. Coded Concept Identifier not found. 4. Coded Concept Version not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	For versions, an acceptable input should be "current" or "latest"

Other relevant content	
Associated Scenario	Retrieve Coded Concepts from Value Set, Resolve Concept Representations, Retrieve Coded Concepts from Code System, Validate Concept in Code System, Identify Concept Language Translations

List Concept Relationship Types

Description	List relationship types that are available on a terminology server that match entered filter criteria (e.g. code system, code system version).
Inputs	1. Filter Criteria
Outputs	1. Returns a list of zero or more relationship types, together with associated metadata
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	None.
Exception Conditions	1. Invalid Filter Criteria.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Enumerate Code System Coded Concept Relationship Types

Return Concept Relationship Type Details

Description	Returns the details for the known attributes (metadata) of a relationship type.
Inputs	1. Concept Relationship Type Id
Outputs	1. Full metadata for the Relationship type
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Relationship Type exists
Postconditions	None.
Exception Conditions	1. Relationship Type Id not found
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Enumerate Code System Coded Concept Relationship Types

Value Set Search / Access

List Value Sets

Description	Lists the value sets that are available to the CTS 2 service.
Inputs	1. Value Set Filter Criteria
Outputs	1. Listing of zero or more value sets on this instance of the terminology server that match the input filter criteria, together with associated metadata.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	None.
Exception Conditions	1. Invalid filter criteria
Aspects left to Technical Specification	
Relationship to levels of	

conformance	Query/Search
Miscellaneous notes	When search attributes are applied, the result set is restricted to the value sets that match the search filter criteria. Examples include: 1. restricting to matching properties such as: 1. Value Set Name 2. value Set version 3. Code Systems that comprise the values of the value set 4. Metadata attributes/properties of the value set
Other relevant content	
Associated Scenario	Resolve Available Value Sets

Return Value Set Details

Description	Look up detailed information (metadata) for a given value set.
Inputs	1. Value Set Identifier 2. Value Set version
Outputs	1. Details (metadata) pertaining to the specified value set (resolved meta data or attributes for the value set.)
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Value Set must be loaded into the terminology service. 3. Value Set must be active.
Postconditions	None.
Exception Conditions	1. Terminology service not available. 2. Value Set Identifier not found. 3. Value Set version not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	For version, an acceptable input should be "current" or "latest".
Other relevant content	
Associated Scenario	Resolve Available Value Sets, Compare Value Set Versions

List Value Set Contents

Description	Lists out the contents (entries) of a given value set, filtering based on input criteria.
Inputs	1. Value Set Identifier 2. Value Set version (Optional) 3. Filter criteria
Outputs	1. A list of zero or more entries for the given value set
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Value Set must be loaded into the terminology service.
Postconditions	None.
Exception Conditions	1. Value Set Identifier not found. 2. Value Set version not found. 3. Invalid filter criteria
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	Value sets may not be finite (e.g. the set of all reals between 1 and 10) Obviously we don't want to list them all.
Other relevant content	
Associated Scenario	Retrieve Coded Concepts from Value Set, Compare Value Set Versions, Resolve Concept Representations

Check Concept Value Set Membership

Description	Determine whether the supplied coded concept exists in the supplied value set
	1. Code System Identifier 2. Concept Identifier

Inputs	3. Value Set Identifier 4. Value Set Version
Outputs	1. Return True if coded concept exists in value set 2. Return False if coded concept does not exist in value set
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service. 3. Value Set must be loaded into the terminology service. 4. Identified Concept present and part of identified code system
Postconditions	None.
Exception Conditions	1. Code System Identifier not found. 2. Concept code not found. 3. Value Set Identifier not found. 4. Value Set version not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Validate Coded Concept in Value Set

Concept Domain and Usage Context Search / Access

List Concept Domains

Description	Lists the concept domains that are available to the CTS 2 service.
Inputs	1. Concept Domain Filter Criteria
Outputs	1. Listing of zero or more concept domains on this instance of the terminology server that match the input filter criteria, together with associated metadata.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	None.
Exception Conditions	1. Invalid filter criteria
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	When search attributes are applied, the result set is restricted to the concept domains that match the search filter criteria. Examples include: 1. restricting to matching properties such as: 1. Concept Domain Name 2. Value Sets that are bound to the Concept Domain 3. Usage Contexts in which the Concept Domain is bound to value sets 4. Metadata attributes/properties of the Concept Domain
Other relevant content	
Associated Scenario	Resolve Available Concept Domains

Return Concept Domain Details

Description	Look up detailed information (metadata) for a given concept domain.
Inputs	1. Concept Domain Identifier
Outputs	1. Concept Domain details (resolved meta data or attributes for the concept domain)
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Concept Domain must be present in the terminology service.

Postconditions	None.
Exception Conditions	1. Terminology service not available. 2. Concept Domain Identifier not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Resolve Available Concept Domains

List Usage Contexts

Description	Lists the usage contexts that are available to the CTS 2 service.
Inputs	1. Usage Context Filter Criteria
Outputs	1. Listing of zero or more usage contexts on this instance of the terminology server that match the input filter criteria, together with associated metadata.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	None.
Exception Conditions	1. Invalid filter criteria
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	When search attributes are applied, the result set is restricted to the usage contexts that match the search filter criteria. Examples include: 1. restricting to matching properties such as: 1. Usage Context Name 2. Concept Domains that are bound to value sets in the Usage Context 3. Metadata attributes/properties of the Concept Domain
Other relevant content	
Associated Scenario	Resolve Available Concept Domains

Return Usage Context Details

Description	Look up detailed information (metadata) for a given usage context.
Inputs	1. Usage Context Identifier
Outputs	1. Detailed Usage Context description (resolved meta data or attributes for the usage context)
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Usage Context must be present in the terminology service.
Postconditions	None.
Exception Conditions	1. Terminology service not available. 2. Usage Context Identifier not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Resolve Available Concept Domains

List Concept Domain Bindings

Description	Lists out the contents (value set bindings) of a given concept domain in usage contexts, filtering based on input criteria.
Inputs	1. Concept Domain Identifier 2. Filter criteria

Outputs	1. A list of zero or more value sets
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Concept Domain must be loaded into the terminology service.
Postconditions	None.
Exception Conditions	1. Concept Domain Identifier not found. 2. Invalid filter criteria
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Retrieve Coded Concepts for Concept Domain

Check Concept Domain Membership

Description	Determine whether the supplied coded concept exists in the supplied concept domain, optionally within specific usage contexts
Inputs	1. Code System Identifier 2. Concept Identifier 3. Concept Domain Identifier 4. List of Usage Context Identifiers (optional)
Outputs	1. Return True if coded concept exists in specified concept domain and usage contexts 2. Return False if coded concept does not exist in specified concept domain and usage contexts
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service. 3. Concept Domain must be loaded into the terminology service. 4. Identified Concept present and part of identified code system
Postconditions	None.
Exception Conditions	1. Code System Identifier not found. 2. Concept code not found. 3. Concept Domain Identifier not found. 4. Usage Context Identifier not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Query/Search
Miscellaneous notes	
Other relevant content	
Associated Scenario	Validate Coded Concept in Concept Domain

Authoring/Curation

Code System Authoring/Curation

Create Code System

Description	Create a new Code System to contain a set of new coded concepts. The Code System is created by defining the set of meta-data properties that describe it.
Inputs	1. Code System Name 2. Code System Version 3. Code System properties
Outputs	1. An acknowledgment indicating whether the code system was created or not. 2. Code System Id, if the code system was successfully created.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	1. The code system is available for access via the CTS 2 service functions.

Exception Conditions	1. Code System already exists.
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	1. If the user wants to specify the Code System Id manually, it can be specified as a "Code System Property". 2. The service will denote that the code system exists when applicable. We will let the individual service administrator treat this exception as an error (refuse to create the codesystem) or a warning (create codesystem anyway). 3. Since there is a separate "import" operation for loading code sets, the semantics of the "create" should be to just create the code system and then create the Concepts. This implies the need for a "release" or "publish" capability, which will be handled by an "Update Code System Version Status" operation.
Other relevant content	
Associated Scenario	Create Code System

Maintain Code System Version

Description	Update Code System meta-data properties.
Inputs	1. Code System Id 2. Code System Name 3. Code System Version 4. Code System properties 5. Set of zero or more Concepts (Ids, Properties, Designations) 6. Set of zero or more Concept Relationships
Outputs	1. An acknowledgment indicating whether the code system was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service.
Postconditions	1. The updated code system is available for access via the CTS 2 service functions.
Exception Conditions	1. Code System does not exist. 2. Code System version does not exist. 3. Invalid Concept Properties 4. Invalid Concept Details 5. Invalid Concept Relationships
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	This allows for maintenance of a Code Systems properties, including creation and updating of Code System Versions, and assignment of Concepts to a Version. Note that the set of concept IDs could either be a complete new full set or deltas only, which needs to be identified explicitly. Entering as new Version Id causes creation of a new version (assuming the state model allows it at the time) Concepts can be assigned and/or unassigned from Code System Versions, as can Concept Properties, Designations and Concept Relationships.
Other relevant content	
Associated Scenario	Maintain Code System

Update Code System Version Status

Description	Changes the status of a code system version (suspend, reinstate, cancel, remove).
Inputs	1. Code System Identifier 2. Code System Version 3. Status
Outputs	1. An acknowledgment indicating whether the status revision request was successfully processed.
Invariants	

Precondition	1. CTS 2 Service installed and running. 2. Code System Version exists.
Postconditions	1. Code System Version status is updated appropriately.
Exception Conditions	1. Code System Version Identifier does not exist. 2. Invalid state change
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Maintain Code System

Create Concept

Description	Create concept to be included in a Code System. The new concept is defined by the set of meta-data properties that describe it, which may include its proper placement via association binding within the hierarchy of the Code System.
Inputs	1. Code System ID 2. Code System Version 3. Concept Code 4. Concept Identifier (optional) 5. Concept Properties 6. (Allowable) Concept Relationships 7. Concept Designations
Outputs	1. An acknowledgment indicating whether the concept was created or not. 2. Concept Identifier
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service. 3. A sequencer that provides Concept Identifiers must be available if the Concept Identifier is not provided as an input.
Postconditions	1. The concept is available in the code system and is available for access via the CTS 2 service functions.
Exception Conditions	1. Code System does not exist. 2. Code System version does not exist. 3. Concept already exists.
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	1. The service will denote that the concept exists when applicable. We will let the individual service administrator treat this exception as an error (refuse to create the concept) or a warning (create concept anyway). Generated ID can be used for disambiguation in code sets where codes are reused. The input Code System Version is the version of the code system in which the concept code is first created Entered Concept Properties and Relationships become the "allowable" or "Defined Concept Properties / Relationships" in the model and also associated with the Code System Version in which they were created.
Other relevant content	
Associated Scenario	Create Concept

Maintain Concept

Description	Update Concept meta-data properties.
Inputs	1. Code System Id 2. Concept Code 3. Concept Id 4. Jurisdictional Domain (optional) 5. Concept Properties 6. (allowable) Concept Relationships

	7. Concept Designations
Outputs	1. An acknowledgment indicating whether the concept was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System must be loaded into the terminology service. 3. Concept exists on system
Postconditions	1. The concept is updated appropriately.
Exception Conditions	1. Code System does not exist. 2. Concept does not exist. 3. Unrecognized Jurisdictional Domain 4. Invalid Concept Properties 5. Invalid Concept Designation
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	Updates include but is not limited to functionality such as: 1. making updates to the associated concept attributes, 2. changing the presentation 3. changing preferred name 4. changing synonymy 5. technical corrections to the concept 6. modifying the associations bound to concepts 7. in keeping with good vocabulary practice, codes or identifiers for concepts cannot be reused. Additionally, in hierarchical Code Systems, it may be necessary to re-associate any concepts related to the concept being deprecated to prevent a part of the code system hierarchy from being orphaned Code System Version not input here. A set of Concept changes can be made then assigned as a new Code System Version before it is published (using Maintain Code System Version and then Update Code System Version Status) Note that Concept Version is NOT explicitly separately created and maintained. This would be automatic by the system. JurisdictionalDomain allows for localization. Entered Concept Properties and Relationships become the "Defined Concept Properties / Relationships" in the model and also associated with the Code System Version in which they were created. They can also be deprecated through use of status codes.
Other relevant content	
Associated Scenario	Maintain Concept

Update Concept Status

Description	Changes the status of a code system concept (suspend, reinstate, cancel, remove).
Inputs	1. Code System Identifier 2. Concept Code 3. Status
Outputs	1. An acknowledgment indicating whether the status revision request was successfully processed.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Concept code exists.
Postconditions	1. Code System Concept status is updated appropriately.
Exception Conditions	1. Code System Identifier does not exist. 2. Concept Code does not exist 3. Invalid state change
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Maintain Concept

Create Concept Relationship Type

Description	Create a new relationship type that may be used to link two concepts. A list of code system IDs can be supplied if the intent is to restrict use to specific code systems. The default is availability to all code systems present on the server.
Inputs	1. Relationship Type ID (optional) 2. Relationship Type Name 3. Relationship Type Properties 4. List of zero or more Code System IDs
Outputs	1. An acknowledgment indicating whether the relationship type was created or not. The ID will be generated if not supplied.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Relationship Type does not already exist.
Postconditions	1. The relationship type is available for use via the CTS 2 service functions.
Exception Conditions	1. Relationship Type already exists. 2. Invalid Relationship Type properties 3. Unrecognized Code System ID
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	1. Creating relationships across two different code systems is permitted by the interface, but should be used with care. It is not certain if semantic relationships can be reliably created/maintained at present even when a reference terminology (to which the queried terminologies are mapped) is available. Note that given that there will often be complex business rules associated at this level, this may also be handled by configuration rather than explicit interface operation.
Other relevant content	
Associated Scenario	Create Relationship Type

Maintain Concept Relationship Type

Description	Update or deprecate a relationship type that may be used to link two concepts.
Inputs	1. Relationship Type ID 2. Relationship Type Name 3. Relationship Type Properties 4. List of zero or more Code System IDs 5. Relationship Type Status
Outputs	1. An acknowledgment indicating whether the relationship type was modified or not.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. The relationship type already exists. 3. Invalid Relationship Type properties
Postconditions	1. The updated relationship type is available for access via the CTS 2 service functions.
Exception Conditions	1. Relationship Type does not exist. 2. Invalid Relationship Type properties 3. Invalid Status 4. Unrecognized Code System ID
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring

Miscellaneous notes	Status change could be a separate operation, but seems unnecessary at this "meta" level.
Other relevant content	
Associated Scenario	Maintain Relationship Type

Value Set Authoring/Curation

Create Value Set by Intension

Description	Create a dynamic Value Set that is defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time.
Inputs	1. Value Set Name 2. Value Set Id (optional) 3. Value Set Version 4. Value Set Properties 5. Value Set Rule Set 6. Value Set Date Lock (boolean)
Outputs	1. An acknowledgment indicating whether the value set was created or not. 2. Generated Value Set Id, if not input
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. A sequencer to provide the value set id, if not provided in the input.
Postconditions	1. The value set is available for access via the CTS 2 service functions.
Exception Conditions	1. Value Set already exists. 2. Invalid Value Set properties 3. Unrecognized/invalid rule set
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	1. Example rule set expression: an intensional value set might be expressed as, “SNOMED CT concepts that are children of the SNOMED CT concept “Diabetes Mellitus”. Note that identification of the target code set(s) is part of the rule set. 2. When creating an intensionally defined value set, the Terminology User may or may not lock the date of the evaluation (effectively binding the value set definition to a specific version of the Code System(s) from which the concepts are being drawn). This would ensure that the value set would always resolve to the same set of concept codes for any given version of the value set. If the value set expression is not locked in time, then it will resolve to a different set of concept codes as the version of the Code System changes. 3. Note that the model allows for the rule set to be defined at the overall Value Set level or at the Version level, to allow for flexibility. The Service consumer needs to be able to indicate which one is appropriate. Entering a new Version Id causes creation of a new version (assuming the state model allows it at the time) The extent to which the rule set allows selection of specific designations would need to be defined (but is an implementation issue), e.g. could specify the language.
Other relevant content	
Associated Scenario	Create Value Set by Intension

Create Value Set by Extension

Description	Create an enumerated (static) value set that is comprised of an explicitly enumerated set of codes.
Inputs	1. Value Set Name 2. Value Set Id (optional) 3. Value Set Version 4. Value Set Properties 5. Enumerated set of concepts
Outputs	1. An acknowledgment indicating whether the value set was created or not. 2. Generated Value Set Id, if not input
Invariants	
Precondition	1. CTS 2 Service installed and running.

	2. A sequencer to provide the value set id, if it's not provided as an input
Postconditions	1. The value set is available for access via the CTS 2 service functions.
Exception Conditions	1. Value Set already exists. 2. Invalid value set properties 3. Concept not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Create Value Set by Extension

Maintain Value Set (Intension)

Description	Update properties or expression of a value set by definition.
Inputs	1. Value Set Id 2. Value Set Name 3. Value Set Version 4. Value Set Properties 5. Value Set Rule Set 6. Value Set Date Lock (boolean)
Outputs	1. An acknowledgment indicating whether the value set was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	1. The updated value set is available for access via the CTS 2 service functions. 2. A new value set version is created.
Exception Conditions	1. Value Set does not exists. 2. Invalid Value Set properties 3. Unrecognized/invalid rule set
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	See notes on Create operation
Other relevant content	
Associated Scenario	Maintain Value Set (Definition)

Maintain Value Set (Extension)

Description	Update properties or concepts of an enumerated (static) value set.
Inputs	1. Value Set Id 2. Value Set Name 3. Value Set Version 4. Value Set Properties 5. Enumerated set of concepts
Outputs	1. An acknowledgment indicating whether the value set was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	1. The updated value set is available for access via the CTS 2 service functions. 2. A new value set version is created.
Exception Conditions	1. Value Set does not exist. 2. Concept not found.
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	

Other relevant content	
Associated Scenario	Maintain Value Set (Enumeration)

Update Value Set Status

Description	Changes the status of a value set version (suspend, reinstate, cancel, remove).
Inputs	1. Value Set Identifier 2. Value Set Version 3. Status
Outputs	1. An acknowledgment indicating whether the status revision request was successfully processed.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Value Set Version exists.
Postconditions	1. Value Set Version status is updated appropriately.
Exception Conditions	1. Va;ue Set Version Identifier does not exist. 2. Invalid state change
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Maintain Value Set (Definition), Maintain Value Set (Enumeration)

Concept Domain and Usage Context Authoring/Curation

Create Concept Domain

Description	Create a Concept Domain.
Inputs	1. Concept Domain Name 2. Concept Domain Id (optional) 3. Concept Domain Properties
Outputs	1. An acknowledgment indicating whether the concept domain was created or not. 2. Generated Concept Domain Id, if not input
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. A sequencer to provide the concept domain id, if not provided in the input.
Postconditions	1. The concept domain is available for access via the CTS 2 service functions.
Exception Conditions	1. Concept Domain already exists. 2. Invalid Concept Domain properties
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Create Concept Domain

Maintain Concept Domain

Description	Update properties of a Concept Domain, including bindings to value sets within usage contexts
	1. Concept Domain Id 2. Concept Domain Name

Inputs	3. Concept Domain Properties 4. Concept Domain Status 5. Set of zero or more pairs of Value Set Id and Usage Context Id
Outputs	1. An acknowledgment indicating whether the concept domain was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	1. The updated concept domain is available for access via the CTS 2 service functions.
Exception Conditions	1. Concept Domain does not exists. 2. Invalid Concept Domain properties 3. Invalid state transition 4. Unrecognized/invalid value set 5. Unrecognized/invalid usage context
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Maintain Concept Domain

Create Usage Context

Description	Create a Usage Context.
Inputs	1. Usage Context Name 2. Usage Context Id (optional) 3. Usage Context Properties
Outputs	1. An acknowledgment indicating whether the usage context was created or not. 2. Generated Usage Context Id, if not input
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. A sequencer to provide the usage context id, if not provided in the input.
Postconditions	1. The usage context is available for access via the CTS 2 service functions.
Exception Conditions	1. Usage Context already exists. 2. Invalid Usage Context properties
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Create Usage Context

Maintain Usage Context

Description	Update properties of a Usage Context
Inputs	1. Usage Context Id 2. Usage Context Name 3. Usage Context Properties 4. Usage Context Status
Outputs	1. An acknowledgment indicating whether the usage context was updated or not.
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	1. The updated usage context is available for access via the CTS 2 service functions.

Exception Conditions	1. Usage Context does not exists. 2. Invalid Usage Context properties 3. Invalid State transition
Aspects left to Technical Specification	
Relationship to levels of conformance	Authoring
Miscellaneous notes	
Other relevant content	
Associated Scenario	Maintain Usage Context

Concept Relationships

List Concept Relationships

Description	List the concept relationships available on an instance of the CTS 2 Service that match a set of input filter criteria
Inputs	1. Filter Criteria
Outputs	A listing of zero or more concept relationships that match the input filter criteria
Invariants	
Precondition	1. CTS 2 Service installed and running.
Postconditions	none
Exception Conditions	1. Invalid filter criteria
Aspects left to Technical Specification	Definition of filter criteria. This is expected to include: Code System Ids and/or Code System Version Ids, Concept Codes / Ids, Relationship Types, whether only direct associations are considered or whether the transitive closure of the relation are used. There are potentially many different types of relationships that can be defined. In some cases, the relationships are explicitly defined, persisted and managed, in other cases, queries can be made based on algorithms (such as those defined in the descriptions of the Create Lexical and Rules Based Relationships. To some degree, these are also both more targeted "query" operations in that they return a list of associations based on the input rules, as well as allowing for persisting of the relationships if so required. For the purposes of the SFM, this operation will allow retrieval of any type of relationship, including lexical and rules based, but the technical specification may define a number of more explicit operations.
Relationship to levels of conformance	Associations
Miscellaneous notes	
Other relevant content	
Associated Scenario	Resolve Available Associations, Validate Associations, Identify / Identify / Retrieve Concept Associations Between for a Single Concept, Identify / Retrieve Associations Between Two or More Coded Concepts

Return Concept Relationship Details

Description	Look up detailed information (metadata) for a given concept relationship
Inputs	1. Concept relationship identifier
Outputs	All available concept relationship information (resolved meta data or attributes for the concept relationship.) Including: 1. Code system identifier(s) 2. Code system version(s) 3. Code system name(s) and description(s) 4. Concept codes / identifiers / names 5. Authoring / curation information 6. External systems coded concept relationship data hosted on the CTS server (i.e.: XML encoded or OWL formatted relationship rule content).
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Concept relationship must be loaded into the terminology service.
Postconditions	None.
Exception Conditions	1. Concept relationship does not exist.
Aspects left to	

Technical Specification	
Relationship to levels of conformance	Associations
Miscellaneous notes	
Other relevant content	
Associated Scenario	Identify / Retrieve Associations Between Two or More Coded Concepts, Retrieve Association Metadata, Identify / Identify / Retrieve Concept Associations Between for a Single Concept, Compare Association VersionsCompare Association Versions, Request / Retrieve Association Instance

Update Concept Relationship Status

Description	Update the status of a concept relationship (active, inactive, cancelled etc). This allows a Terminology User to activate or inactivate a given concept relationship, thus changing its availability for access by other terminology service functions
Inputs	1. Concept relationship identifier. 2. Status
Outputs	An acknowledgement indicating whether the concept relationship status has been successfully updated.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Concept relationship must be loaded into the terminology service.
Postconditions	
Exception Conditions	
Aspects left to Technical Specification	
Relationship to levels of conformance	Associations
Miscellaneous notes	
Other relevant content	
Associated Scenario	Change Status of Coded Concept Associations, Remove Coded Concept Association

Create Concept Relationship

Description	Relates a single specific coded concept within a specified code system (source)to a corresponding single specific coded concept (target) within the same or another code system, including identification of a specified relationship type.
Inputs	1. Source code system identifier. 2. Target Code system identifier. 3. Source Concept code. 4. Target Concept code. 5. Relationship Type 6. Relationship Properties
Outputs	Concept Relationship Id
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Code System(s) must be loaded into the terminology service. 3. Source Coded Concept must exist. 4. Target Coded Concept must exist.
Postconditions	A concept relationship of the specified type is created between two coded concepts
Exception Conditions	1. Source Code System does not exist. 2. Target Code System does not exist. 3. Source Coded concept not found. 4. Target Coded concept not found. 5. Unrecognized Relationship Type 6. Invalid relationship properties
Aspects left to Technical Specification	
Relationship to levels of conformance	Associations
Miscellaneous notes	See notes under "Import Terminology Revision" for a list of sample properties for Concept Relationships
Other relevant content	
Associated Scenario	Create/Maintain Association between Coded Concepts

Create Lexical Relationship Between Coded Concepts

Description	Relates a set of zero or more coded concepts within a specified code system (source)to a corresponding set of zero or more coded concepts (target) within that system or another code system using a set of lexical rules (matching algorithms) to generate the relationships. The "Source Search Criteria" allows for identification of a subset of the Source Code System to apply the matching algorithm to, if required (this may include limiting the version of the code system).
Inputs	1. Source Code System Identifier 2. Target Code System Identifier 3. Source Search Criteria (optional) 4. Match Algorithm Code
Outputs	List of zero or more Concept Relationship Ids (together with identification of each pair of Concepts that were linked as a result of the matching.
Invariants	
Precondition	1. CTS 2 Service installed and running. 2. Source Code System must be loaded into the terminology service. 3. Target Code System must be loaded into the terminology service.
Postconditions	A set of zero or more Concept Relationships are created, each between a coded concept from the source code system and a coded concept in the target code system.
Exception Conditions	1. Source Code System does not exist. 2. Target Code System does not exist. 3. Unrecognized / Invalid algorithm. 4. Unrecognized / Invalid search criteria. 5. (Warning) No relationships matched the input algorithm.
Aspects left to Technical Specification	This may be a very time consuming operation across large code sets, and may potentially produce large result sets. Management of result set size is envisaged to require further consideration and probably additional input parameters.
Relationship to levels of conformance	Associations
Miscellaneous notes	Even though this is expressed as a "create" operation, it is essentially also a "return" operation in that it identifies which concepts are related by the input rules.
Other relevant content	

Match Algorithm Code	Description
IdenticalIgnoreCase	The lower case representation of the target text must match the lower case representation matchText exactly.
Identical	The target text must match the matchText exactly.
StartsWithIgnoreCase	The lower case representation of target text must begin with the lower case representation of matchText.
StartsWith	The target text must begin with the matchText.
EndsWithIgnoreCase	The lower case representation of the target text must end with the lower case representation of matchText.
EndsWith	The target text must end with the matchText.
ContainsPhraseIgnoreCase	The lower case representation of the target text must contain the lower case representation of the matchText.
ContainsPhrase	The target text must contain the matchText.
WordsAnyOrderIgnoreCase	The target text must contain all of the words in the match text, but in any order.
WildCardsIgnoreCase	The match text may contain zero or more 'wild cards', designated by an asterisk (*). Wild cards match 0 of more characters in the target string. The escape character is a backslash('\') meaning that the matchText "a*b*' would match any string that begins with the string "a*b".
RegularExpression	The match text may contain regular expressions, as defined in XML Schema Part 2: Datatypes.
NYSIIS	New York State Identification and Intelligence System phonetic encoding

Associated Scenario	Create Lexical Association

Create Rules Based Relationship Between Coded Concepts

Description	Relates a set of zero or more coded concepts within a specified code system (source)to a corresponding set of zero or more coded concepts (target) within that system or another code system using a set of description logic or inference rules that either assert or infer relationships. The "Source Search Criteria" allows for identification of a subset of the Source Code System to apply the matching algorithm too, if required (this may include limiting the version of the code system).
Inputs	1. Source Code System Identifier 2. Target Code System Identifier 3. Source Search Criteria (optional) 4. Description Logic 5. Inference Rules
Outputs	List of zero or more Concept Relationship Ids (together with identification of each pair of Concepts that were linked as a a result of the matching.
Invariants	
Precondition	Specified code system(s) is/are loaded and available on the terminology server
Postconditions	A set of zero or more Concept Relationships are created, each between a coded concept from the source code system and a coded concept in the target code system.
Exception	1. Source Code System does not exist. 2. Target Code System does not exist. 3. Unrecognized / Invalid description logic.

Conditions	4. Unrecognized / invalid inference rules. 5. Unrecognized / Invalid search criteria. 6. (Warning) No coded concepts satisfy the description logic or inference rules.
Aspects left to Technical Specification	This may be a very time consuming operation across large code sets, and may potentially produce large result sets. Management of result set size is envisaged to require further consideration and probably additional input parameters.
Relationship to levels of conformance	Associations
Miscellaneous notes	Even though this is expressed as a "create" operation, it is essentially also a "return" operation in that it identifies which concepts are related by the input rules.
Other relevant content	These relationships are subject to human review to verify validity.
Associated Scenario	Create Rules Based Association

Profiles

Introduction

A profile is a named set of cohesive capabilities. A profile enables a service to be used at different levels and allows implementers to provide different levels of capabilities in differing contexts. Service-to-service interoperability will be judged at the profile level and not the service level. Note that through the use of profiles, there are no “optional” interfaces. Conditions that might otherwise merit this optionality should be addressed via a dedicated profile.

A set of profiles may be defined that cover specific functions, semantic information and overall conformance. The SSDF explains in detail the meaning of each of these types of profile. In brief, they are as follows:

- **Functional Profile:** a named list of a subset of the operations defined within this specification which must be supported in order to claim conformance to the profile.
- **Semantic Profile:** identification of a named set of information descriptions (e.g. semantic signifiers) that are supported by one or more operations.
- **Conformance Profile:** this is a combination of a set of functional and semantic profiles taken together to give a complete coherent set of capabilities against which conformance can be claimed. This may optionally include additional constraints where relevant.

CTS 2 Functional Profiles

Minimal CTS 2 Query Profile

The Minimal CTS 2 Profile specifies basic query only functional coverage necessary for a service to declare itself as being a minimally conformant CTS 2 service. The minimal CTS 2 includes capabilities for searching and query terminology content. If only this profile were supported, other means would be necessary for defining and structuring code systems and concepts, metadata, importing and exporting code systems and so on.

Profile	Member Operations
Minimal CTS 2 Profile	List Code Systems (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Code_Syst
	Return Code System Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Code_System_Definition)
	List Code System Concepts (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Code_System_Concepts)

Return Coded Concept Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Coded_Concept_Definition

List Value Sets (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Value_Sets)

Return Value Set Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Value_Set_Definition)

List Value Set Contents (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Value_Set_Contents)

Check Concept Value Set Membership (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Check_Concept_Value_Set_Memb

List Concept Domains (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Concep

Return Concept Domain Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Concept_Domain_Definitio

List Concept Domain Bindings (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Concept_Domain_Bindings)

Check Concept Domain Membership (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Check_Concept_Domain_Member

List Usage Contexts (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Usage_C

Return Usage Context Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Usage_Context_Definition

List Concept Relationships (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Concept_Relationships)

Return Concept Relationship Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Concept_Relationship_Def

List Concept Relationship Types (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#List_Concept_Relationship_Types)

	Return Concept Relationship Type Definition (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Return_Concept_Relationship_Typ

Vocabulary Facilitator Profile

The Vocabulary Facilitator Profile is intended to support the ability for Vocabulary Facilitators to create, modify, package and submit change requests to a Terminology Provider. Change requests to the terminology do not modify the terminology content directly, but result in a collaborative community consensus recommendation to the Terminology Provider that outlines a requested modification to the source terminology. These change requests can then be reviewed by the Terminology Provider, and when appropriate, included in the next release of the source terminology. Although this functionality is within the overall process scope, managing change requests is out of scope for this service at this time.

Terminology Administration Profile

The Terminology Administration profile builds on the minimal profile and is intended to provide the functional operations necessary for terminology administrators to be able to access and make available terminology content obtained from a Terminology Provider. Terminology Administrators are required to interface with Terminology Provider systems in order to obtain the terminology content, then load that terminology content on local Terminology Servers.

Profile	**Member Operations**
Terminology Administration Profile	Import Terminology (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Import_Terminology)
	Import Terminology Revision (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Import_Terminology_Revis
	Export Content (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Export_Con
	Change Terminology Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Change_Terminology_Statu

	Register for Notification (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Register_for_Notification)
	Update Notification Registration (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Update_Notification_Regist
	Update Notification Registration Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Update_Notification_Regist
	Minimal CTS 2 Profile (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Minimal_CTS_2_Profile)

Terminology Authoring Profile

Terminology authors require the capability to robustly query and access terminology content, as well as directly modify the terminology content. The Terminology Authoring profile is intended to provide the functional operations necessary for terminology authors to analyze the existing terminology content, as well as directly edit terminology content.

Profile	Member Operations
Terminology Authoring Profile	Create Code System (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_C
	Maintain Code System Version

(http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maintain_Code_System_Versi
Update Code System Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Up
Create Concept (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_Concep
Maintain Concept (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maintain_C
Update Concept Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Update
Create Value Set By Intension (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#
Create Value Set by Extension (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_Value_Set_by_Extensi
Maintain Value Set (Intension) (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maintain_Value_Set_%28Inter
Maintain Value Set (Extension) (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maintain_Value_Set_%28Exte
Update Value Set Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Updat
Create Concept Domain (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create
Maintain Concept Domain (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Ma
Create Usage Context (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_

Maintain Usage Context (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maint
Terminology Administration Profile (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Terminology_Administration_
Create Concept Relationship (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#C
Update Concept Relationship Status (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Update_Concept_Relationship
Create Concept Relationship Type (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_Concept_Relationship_
Maintain Concept Relationship Type (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Maintain_Concept_Relationsh
Create Lexical Relationship Between Coded Concepts (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_Lexical_Relationship_
Create Rules Based Relationship Between Coded Concepts (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Create_Rules_Based_Relation

CTS 2 Semantic Profiles

Semantic profiles are created to group together vocabularies with similar designs. Vocabularies grouped under a single semantic profile can be queried using the same functional variants of CTS2 functions. This approach provides the following advantages:

- It allows the CTS2 author to focus on a set of design attributes of terminologies and support them using functional variants, rather than having to focus on individual terminologies while authoring the standard.
- It allows the implementer to implement functional variants of CTS2 functions based on the semantic profiles they want to support rather than to create or implement functional variants for the terminologies that are to be supported by their implementation.
- It allows terminology authoring organizations to classify their terminology under a semantic profile and insulates them from the complexities of functional variants of CTS2 functions.

These intrinsic qualities of terminologies allow the functional profiles to be implemented in accordance with the properties of the classes of these terminologies. The following Semantic Profiles for terminologies are are defined currently:

Mature Terminology Profile

Profile	Sample Terminology Criteria	Sample Terminologies Classified Under Profile	Notes
Mature Terminology Profile	▪ Unique identifiers for all concepts ▪ Unique identifiers for all designations ▪ Unique identifiers for all relationships ▪ Identifiers are never reused.	▪ SNOMED CT, all versions ▪ ICD 9 CM ▪ ICD 10 CM ▪ LOINC ▪ RxNorm ▪ MEDCIN ▪ NDF / NDF-RT ▪ CPT	Terminologies in the Mature Terminology Profile make an attempt to conform to many of terminology best practices that are, for example outlined in *Desiderata for Controlled Medical Vocabularies in the Twenty-First Century*, *James J. Cimino*.

Developing Terminology Profile

Profile	Sample Terminology Criteria	Sample Terminologies Classified Under Profile	Notes
Developing Terminology Profile	▪ Codes that are not globally unique within the terminology, but need further qualification, e.g. an additional identifier. ▪ The concepts can be uniquely identified by combining the concept code and an additional identifier.	▪ Some HL7 Vocabulary tables (where the additional identifier may refer to a concept domain) ▪ Locally developed terminology sources or code sets	Terminologies in the Developing Terminology Profile are either developed using adhoc techniques, or have degraded over time.

CTS 2 Conformance Profiles

Conformance Interoperability

The capabilities defined within the CTS 2 service functional model have been attributed to different functional profiles. The purpose of functional profiles is to group together functions to form cohesive levels of operational capability against which implementations can be tested for conformance. Thus, interoperability between CTS 2 implementations is assured within a specified conformance profile. In other words, two CTS 2 implementations that conform to the Terminology Authoring profile will be able to interoperate using the functions described in that profile.

These profiles serve to educate the purchasing and implementation communities, allowing for implementation variation while still promoting interoperability. Service Level Agreements made between organizations are then testable because they are informed by these profiles. Governance of these agreements is less ambiguous and more enforceable due to precise functional levels of interoperability that may be expected.

Implementation of this functional specification should explicitly deal with the different interoperability roles that CTS 2 may fill using these conformance profiles. The business rules enforced by an organization’s purchasing, implementation, and governance arms should be discussed, and the ways in which CTS 2 facilitates that enforcement should be made clear.

Conformance Assertion

Implementations of CTS 2 conform to a specified conformance profile, which is a combination of a functional and semantic profile. That is, conformance to a specific profile is asserted to against the quality metric of a specified semantic profile in association with the specified functional profile.

There are currently four different functional profiles defined. Each profile can be implemented according to either the **Mature Terminology** or **Developing Terminology** semantic profiles, providing up to eight possible levels of conformance to CTS 2.

	Mature Terminology Semantic Profile	Developing Terminology Semantic Profile
Minimal Query CTS 2 Functional Profile	Minimal Query CTS 2 - Mature Terminology Conformance Profile	Minimal Query CTS 2 - Developing Terminology Conformance Profile

Terminology Administration Functional Profile	Terminology Administration - Mature Terminology Conformance Profile	Terminology Administration - Developing Terminology Conformance Profile
Terminology Authoring Functional Profile	Terminology Authoring - Mature Terminology Conformance Profile	Terminology Authoring - Developing Terminology Conformance Profile

The Services Framework Functional Model

The Services Framework Functional Model identifies common underlying enterprise infrastructure such as naming, directory, security, etc. that may be assumed and referenced by this Functional Model.

Note that the Services Framework Functional Model is being developed in parallel with other service Functional Models; candidate functionality for the Framework should be submitted to the Infrastructure subgroup for evaluation.

CTS 2 compliant service instances are intended to be healthcare middleware services and to work within the context of supporting infrastructure services that may exist within an enterprise. As a result, a number of underpinning capabilities have been intentionally omitted from the scope of this specification. These include (but are not limited to) capabilities such as identity management, security and record location services.

The CTS 2 specification, by design, can be used as a means to integrate a new capability into a service-oriented architecture, or can be used to provide a service interface to access content in legacy applications. It is not intended as a replacement of any single system, but instead to act as a companion component that facilitates interoperability with data sharing partners through a standardized set of APIs.

CTS 2 serves as a simplifying resource for the organization, as it provides a single point of access for all terminology resources.

Relationship to Information Content

The following principles shall be followed for specifying the information model to be used by the services being specified in this Service Functional Model:

1. SFMs shall provide a conformance profile supporting HL7 content where relevant
2. We shall not preclude the use of non-HL7 content
3. SFMs will reuse to the maximum extent possible the content models as defined in other standards (for example, HL7 RMIMs)
4. Information content representations shall be represented in platform-agnostic formalisms (e.g., UML)
5. SFMs may identify content at varying levels of granularity, depending upon the functions being specified. (For example, the Common Terminology Service will deal with different granularity of information than the Resource Location and Update Service).
6. Conformance Profiles may be balloted or adopted after the release of the initial SFM to address specialized business needs. (realm-specific profiles, domain-specific profiles, etc.)
7. Details about semantics specific to this SFM appear in other sections of this document

Recommendations for Technical RFP Issuance

This section includes Identification of topics requiring elaboration in candidate solutions provided through the OMG RFP process. These may be service-specific, deployment related, or non-functional.

HL7 Support

As the next version of the HL7 CTS specification, CTS 2 will be required to support the use of terminology in HL7 environments, including HL7 models.

HL7 Model Interchange Format (MIF) Support

Responders to the RFP should explicitly discuss how CTS 2 will support the import of terminology content represented in HL7 MIF into the CTS 2 service, as well as the export of content from the CTS 2 service into a MIF representation. The reponder should provide a mapping from the implementatino model of CTS 2 to the HL7 vocabulary MIF.

Semantic Signifiers: Disparate Terminologies

While defining the semantics of payloads sent through CTS 2 is beyond the scope of this publication, the ability of CTS 2 to notify a service partner about the nature of the capabilities of that implementation of CTS 2 is essential to fulfilling terminology service interoperability.

CTS 2 could conceivably be used to access and maintain a great variety of terminology sources, including SNOMED, ICD, and RxNorm (to name a few). To create true terminological interoperability between organizations it is essential to provide a scalable and extensible terminology model that can be included in the description of and access to the terminology resources available on any given terminology service.

Though a limited number of semantic signifiers have been included in this document as a mechanism of defining the necessary behaviors of a terminology, it is expected that HL7, HL7 member organizations, terminology providers, and terminology users will be producing representations that will be supported within a given CTS 2 implementation.

Semantic Signifiers: HL7 Terminologies

Where terminology content exposed through CTS 2 is from an HL7 domain it is necessary to includes support for Concept Domains, Binding Realms and domain bindings.

RFP submitters should take the requirement for Domain and Binding Realm descriptions as a starting point to discuss the additional physical information descriptions. The usage of the two should be described and modeled so as to paint a complete picture of the issue of semantic description and discovery through the CTS 2 interface.

Additionally, Semantic Signifiers should allow for the use of some sort of logical operators in describing their hierarchy or aggregation. For example, Boolean Operators

(AND, OR, NOT) should be available in creating query parameters.

This should be discussed in detail by RFP Submitters.

Conformance Profiles and Service Level Agreements

The capabilities defined within the CTS 2 SFM have been attributed to specific conformance profiles. The purpose of a conformance profile is to group together functions to form cohesive levels of operational capability against which implementations can be tested for conformance. Thus, interoperability between CTS 2 implementations is assured within a conformance profile. In other words, two CTS 2 implementations that conform to the Authoring profile will be able to interoperate using the functions outlined in that profile.

These profiles serve to educate the purchasing and implementation communities, allowing for implementation variation while still promoting interoperability. Service Level Agreements made between organizations are then testable because they are informed by these profiles. Governance of these agreements is less ambiguous and more enforceable due to precise functional levels of interoperability that may be expected.

Implementation of this functional specification should explicitly deal with the different interoperability roles that CTS 2 may fill using these conformance profiles. The business rules enforced by an organization's purchasing, implementation, and governance arms should be discussed, and the ways in which CTS 2 facilitates that enforcement should be made clear.

Operationalizing CTS 2: Considerations in Implementation

Optimization

Structured terminologies can be quite large in nature in both the number of concepts, designations, associations, and other attributes that further describe terminology content. As such, efficiently accessing and querying terminology content is critical.

Responders to the RFP should discuss optimization strategies for accessing and updating specific terminologies.

Versioning

To ensure consistent representation of vocabulary versions, the CTS 2 committee recognizes the following approaches to versioning vocabulary entities:

- A vocabulary specification may use a numeric or decimal representation of the version number, that is, the version identifier may be that string containing only numeric digits and decimal points.
- A vocabulary specification may use a release date to represent the version.

Responders to the RFP should contrast these two versioning mechanisms, and discuss how each are supported by the proposed solution, including the recommended string and date formats respectivey.

Internationalization

Responders to the RFP will discuss what effect, if any, localization and internationalization of terminologies will have on technical implementations of CTS 2?

Service Description and Discovery

Because CTS 2 exists as a service between organizations, CTS 2 should be considered a perfect candidate to benefit from service description and discovery, such as what terminologies are available on any given CTS 2 implementation, ad the specific profiles implemented by that service implementation.

Responders to the RFP should explicitly discuss this deployment case, how to better describe CTS 2 to improve service discovery.

Federated Terminologies

As implementers strive to organize CTS 2 within and between institutions,it is likely that a federation of terminology sources and terminology servers will develop. These service interfaces will occupy various information and domain levels within and between organizations. Common federation patterns are likely to emerge, such as a mesh or a hierarchical structure. However, other deployment scenarios are desirable as well. Special attention should be paid to implementation in a non-homogeneous environments.

Responders to the RFP discuss how the implementation would support federated terminologies, and how it would allow for a hierarchical service topology to satisfy most deployment requirements.

Terminology Structure Considerations

Section 2.4.2 (http://biomedgt.org/apelcore/index.php/HL7_Common_Terminology_Services_2_Service_Functional_Model_%28SFM%29#Terminology_Structure_Considerations) of this document outlines a minimal terminology model and attributes for terminologies entities. This model represents the minimal classes, attributes and associations necessary to represent conceptual terminologies.

Responders to the RFP should provide a detailed implementation model that can represent terminology sources that adhere to terminology best practices, and discuss a strategy for representing less mature terminologies in a format that allows them to be consistently accessed by the appropriate CTS 2 functions in accordance with the required semantic profiles.

Terminology Maintenance

CTS 2 specifies a set of service functions specific to the maintenance of terminology sources. These service functions in and of themselves are expected to influence a broader set of application level functions that comprise a terminology maintenance solution.

Such a terminology maintenance solution would include workflow and process that would not only allow terminology providers to efficiently and accurately maintain the terminology content, but may also include the ability for terminology users to submit, review and modification of terminology content *change requests*. These change requests would be considered by the terminology providers for inclusion into the next version of the terminology.

Responders to the RFP should discuss how CTS 2 query and maintenance functionality would be included in a broader terminology maintenance solution that includes workflow and change request processing.

Appendix A - Relevant Standards

HL7 Common Terminology Services (http://informatics.mayo.edu/LexGrid/downloads/CTS/specification/ctsSpec/cts.htm)

The Common Terminology Services (CTS) specification was developed as an alternative to a common data structure.The HL7 Common Terminology Services (HL7 CTS) is an Application Programming Interface (API) specification that is intended to describe the basic functionality that will be needed by HL7 Version 3 software implementations to query and access terminological content. It is specified as an API rather than a set of data structures to enable a wide variety of terminological content to be integrated within the HL7 Version 3 messaging framework without the need for significant migration or rewrite.Instead of specifying what an external terminology must look like, HL7 has chosen to identify the common functional characteristics that an external terminology must be able to provide. As an example, an HL7 compliant terminology service will need to be able to determine whether a given concept code is valid within the particular resource. Instead of describing a table keyed by the resource identifier and concept code, the CTS specification describes an Application Programming Interface (API) call that takes a resource identifier and concept code as input and returns a true/false value. Each terminology developer is free to implement this API call in whatever way is most appropriate for them.There are two layers between HL7 Version 3 message processing applications and the target vocabularies. The upper layer, the Message API communicates with in terms of vocabulary domains, realms, coded attributes and other artifacts of the RIM and HL7 messaging model. The lower layer, the Vocabulary API communicates in terms of coding system, concept codes, designations, and other vocabulary related entities.**NOTE**: THE CTS II specification is an extension of the original HL7 Common Terminology Services approved standard

The Lexical Grid (http://informatics.mayo.edu/)

The Lexical Grid is a proposal for standard storage of terminologies and ontologies. The LexGrid Model defines how terminologies should be formatted and represented programmatically. It also defines several different server storage mechanisms and a XML format.

Appendix B – Glossary

This glossary is split into two separate subsections. The first subsection is entitled **Terminology Core Principles**, and is intended to include terms that are specific to terminologies and terminology services that help define the semantics of terminology principles used in this specification.

The terms used in the **Terminology Core Principles** are intended to align with the corresponding definitions for terminology elements contained in the balloted **Core Principles and Properties of HL7 Version 3 Models** document.

NOTE: In the case of any discrepancy between the terms in the **Terminology Core Principles** section below and the balloted **Core Principles and Properties of HL7 Version 3 Models** document, the **Core Principles and Properties of HL7 Version 3 Models** document is assumed to be correct.

Terminology Core Principles

Code System

A *Code System* is defined as a collection of codes with associated designations, meanings and associations. The persistent representation of a Code Systems include meta-data about the code system itself, as well as the contents of the Code System.

Examples of *Code Systems* include ICD-9 CM, SNOMED CT, LOINC, and CPT. To meet the requirements of a *Code System* as defined by HL7, a given *Concept Code* must resolve to one and only one meaning within the *Code System*. Given this definition, each table in the HL7 Version 2 standard represents a different *Code System* since *Concept Codes* are sometimes used in different tables to have different meanings. For example, the *Concept Code* “M” in the gender *Code System* means “Male”, while “M” in the marital status *Code System* means “Married”

Concept Relationship / Map

A concept relationship is an association between two or more concepts (concept map is usually used where the relationship crosses two different code systems). The endpoints of a concept relationship are source and target concepts, implying a direction of the relationship from a source to a target, which has bearing on the meaning of the relationship and the concepts it connects. A concept relationship is definitional, in that the relationship gives meaning to the concepts associated. For example, a relationship between a parent and child concept indicates that the child concept is a refinement or an example of the parent concept in a concept hierarchy. A concept relationship can also define other characteristics of a concept, as in relationships between concepts in different parent-child hierarchies where the child may have a different set of relationship than that one or more of its hierarchical parents.

Nested Value Sets

When a Value Set Entry references another Value Set, the child value set is referred to as a *Nested Value Set*. There is no preset limit to the level of nesting allowed within value sets. Value sets cannot contain themselves, or any of their ancestors (i.e. they cannot be defined recursively).

Intensional Value Sets can be defined by either fixing the Value Set definition to a specific version of the Code System (when the Code System supports versioning), or by decoupling the Value Set definition from the version of the code system. This seemingly subtle variation can have very significant impact on the final list of concepts which the Value Set ultimately resolves to.

When the Value Set definition is tied to the version of the Code System, the value set content will remain fixed when instantiated. When the Value Set definition is independent of Code System version, the content of the Value Set can vary as the Value Set is resolved against different versions of the Code System.

Value Set

A *Value Set* represents a uniquely identifiable set of valid concept representations, where any concept representation can be tested to determine whether or not it is a member of the value set.

Value set complexity may range from a simple flat list of concept codes drawn from a single code system, to an unbounded hierarchical set of possibly post-coordinated expressions drawn from multiple code systems.

Value sets exist to constrain the content for a coded element in an HL7 static model or data type property. Value sets cannot have null content, and must contain at least one concept representation where any given concept is generally (but not required to be) represented by only a single code within the Value Set.

Sub-value Sets

A *sub-value set* is a sub-set of a parent *Value Set*.

Value Set Specification

Value sets can be specified in two ways, either by enumeration (*extension*), or definition (*intention*). **Extensional Value Set Representation (Enumeration)**

Extensional Value Set

An extensionaly defined, enumerated set consists of a list of unique identifiers and the corresponding globally unique id. Whenever possible, it is strongly recommended that the value set identifier match the code used in the code system itself.

Note, that while all of the entries in a value set may be valid at the time that the value set was defined, it is quite possible that subsequent versions of a code system to retire some of the codes in the value set. For this reason, it is important that both explicit and implicit undergo the “value set binding” set described in a subsequent section.

> From ISO (http://www.tc215wg3.nhs.uk/pages/pdf/vote0204.pdf), an extensional definition is a description of a concept by enumerating all of its subordinate concepts under one criterion of subdivision.

Value sets defined by extension are comprised of an explicitly enumerated set of codes. The simplest case is when the value set consists of only one code.

Code Value	Description
M	Male
F	Female
U	Unspecified

More complex variations might re 1000 late to hierarchical coding systems such as the following fictitious example, where “Level” represents the nesting level for a particular Code Value:

Code Value	Level	Description
1123123	1	Education
1343434	2	Diabetic Education
1445455	2	Stroke Education
2135534	1	Counseling
2344566	2	Emotional
3456663	2	Daily Living

Intensional

Intensional Value Set Definition (Definition)

An intensional value set definition describes the contents of a value set in a way that the definition can be resolved (ideally computationally) to a set of permissible values and unique identifiers. While the construction rules can potentially be quite elaborate and possible specific to individual coding schemes, HL7 has identified a core set of rules that appear to be useful in most circumstances. For the sake of value set definition interoperability, HL7 strongly recommends that these algorithms be used whenever possible.

For example, an intensional value set definition might be defined as, “ SNOMED CT concepts that are children of the SNOMED CT concept “Diabetes Mellitus.”

> From ISO (http://www.tc215wg3.nhs.uk/pages/pdf/vote0204.pdf), an intensional definition describes the intension of a concept by stating the superordinate concept and the delimiting characteristics.

Implicit definitions can specify:

- All active unique identifiers from a given coding system.
- All unique identifiers that participate in a specified relationship with a given local code in a coding system, which may or may not include the specified code itself.
- The transitive closure of a specified transitive relationship with a given code, including or excluding the code itself.
- A reference to another value set
- Other mechanisms that have to have external human or computational resolution.

- Unions, intersections and exclusions of any of the above
- Nested value set definition in which a value set entry references another value set (a child value set). There is no preset limit to the level of nesting allowed within value sets. Value sets cannot contain themselves, or any of their ancestors (i.e., they cannot be defined recursively). Nested value sets are always intensionally defined in HL7.

Nested Value Sets

When a Value Set Entry references another Value Set, the child value set is referred to as a *Nested Value Set*. There is no preset limit to the level of nesting allowed within value sets. Value sets cannot contain themselves, or any of their ancestors (i.e. they cannot be defined recursively).

Intensional Value Sets can be defined by either fixing the Value Set definition to a specific version of the Code System (when the Code System supports versioning), or by decoupling the Value Set definition from the version of the code system. This seemingly subtle variation can have very significant impact on the final list of concepts which the Value Set ultimately resolves to.

When the Value Set definition is tied to the version of the Code System, the value set content will remain fixed when instantiated. When the Value Set definition is independent of Code System version, the content of the Value Set can vary as the Value Set is resolved against different versions of the Code System.

Specification

Actor

In the Unified Modeling Language (UML), an actor is something or someone who supplies a stimulus to the system. An actor cannot be controlled by the system and is defined as being outside the system. An actor is often thought of as a role, rather than an actual person. A single person in the real world can be represented by several actors if they have several different roles and goals in regards to a system. Source: http://en.wikipedia.org/wiki/Actor_(UML)