
# Smart Open Services for European Patients

**Open eHealth initiative for a European large scale pilot of**

**Patient Summary and Electronic Prescription**

# Work Package 3.5 - Semantic Services

## Appendix F - Terminology Access Services

**D3.5.2**

**February 15th, 2010**

**Document Version: 0.0.4**

| | |
|---|---|
| WORK PACKAGE | **WP 3.5** |
| DOCUMENT VERSION | **0.0.4** |
| DATE | **15/02/2010** |

# Table of contents

For:

- Document Information
- Sub-Project Identification
- History of Alteration
- Referring Documents

Please refer to the main document:

***"Semantic Services Definition D3.5.2" - D3.5.2_epSOS_WP3_5_v0.0.5_20100223.doc***

# Appendix F - Terminology Access Services

## 1. General Description

### 1.1 Scenarios

Within the usage of the epSOS Terminology Access Services (epSOS TAS) one can distinguish a series of informational scenarios. The main use case of the service is the transactional translation scenario of each (coded) concept included in the original Patient Summary, ePrescription and eDispensation documents. This takes place at run-time, using a local or remote instance of the epSOS Master Translation/Transcoding Catalogue (epSOS MTC) through a standard interface. Users of the epSOS Terminology Access Services (HCPs, NCP employees, epSOS terminology curators) will further submit concepts or terms for integration into the epSOS MTC with the help of an epSOS terminology community platform.

Just as a reminder, the content of the epSOS MTC are build upon the contents of the epSOS Master Value Sets Catalogue. The contents of the epSOS MVC were chosen according to the use cases, and attempting to provide as much of a medical coverage as possible. The choice of terms is not within the scope of Appendix F. The reader is guided towards the main deliverable 3.5.2.

In terms of responsibilities, each Member State is responsible for mapping (either translating the display name if the code system is the same, or transcoding if the Member State is using a different code system).

Submitted terms will be carefully analyzed by the central entity yet to be defined detaining the master version of the epSOS MTC and will be included in future releases of the epSOS MTC. The NCPs will then be notified of the existence of a new version release and will update their locally used epSOS repository in the case of a distributed terminology access services model. Mapping concepts involves a complex decisional scenario in finding semantic equivalents between two code systems. Since commonly used official terminologies or controlled medical vocabularies are hierarchically organized, granularity issues are quite common in mapping projects. In such cases, semantic disambiguation of concepts using ontology will offer great support for curators. Language translation of display names or surface forms is also a common scenario in curation. In a more advanced setting, the granularity of mappings is set by the content of the epSOS Master Value Set Catalogue (epSOS MVC), and the ontology. epSOS TAS implements this granularity in its services, using Quality attribute of composite type Mapping.

At the moment, CDA expression of the data elements contains bindings to the various epSOS Value sets present in the epSOS MVC. Since each value set is used at the moment as a flat list of terms within the CDA document, the granularity within the ontology is not a current issue. The process of asking for a change or correction of the content of the epSOS MVC is centralized. There is only one master version of the epSOS MVC, and the processes of requesting a change or a correction by each MS is done through excel sheets. The CDA bindings assure that the terms are part of the value set in question and that they are used in the right context. The excel file, for the epSOS TAS, can be fed directly into an SQL server or an Oracle data base.

For a full description of each of the roles, please see the sections underneath.

## 1.2 Terminology Access Services Actors & Roles

The epSOS Terminology Access Services (epSOS TAS) stakeholders involved in the information exchange between paired member states NCPs can be categorized with respect to two main roles:

- epSOS TAS Producers and
- epSOS TAS Services Consumers

TAS producers include the epSOS MTC content authors/curators, subset developers, translators, mappers of the MTC content as well as administrators that will expose releases of the catalogue to the TAS consumers.

Consumers of the epSOS TAS include the health care providers, the actual beneficiaries of the services, but also software engineers that enable access to the terminology itself. However authors/curators manifest in their turn a general consumer role before creating and editing the MTC content. The complexity of user roles present in epSOS transactions and processes can be summarized in the following actor/role diagram:

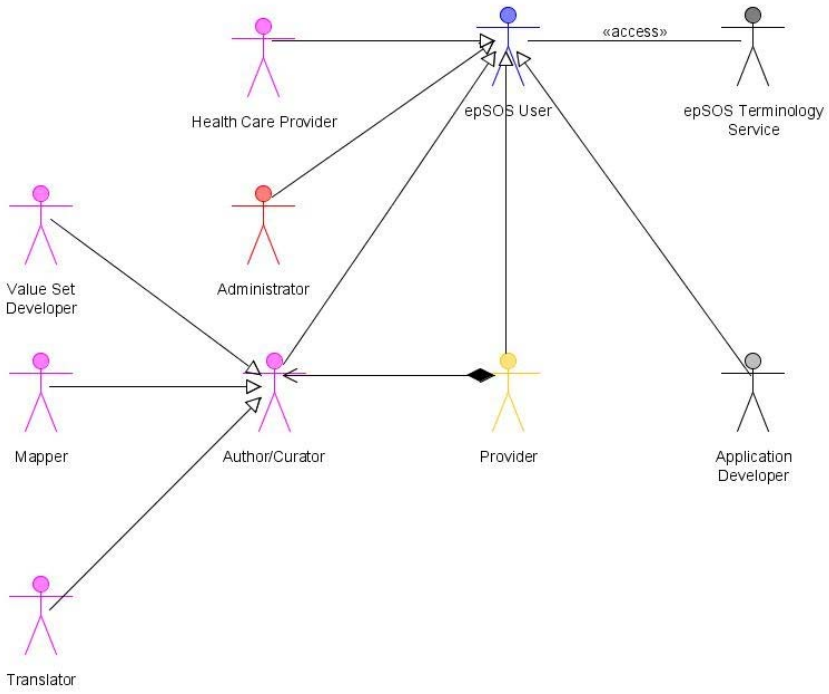



***Figure 1F*** *- TAS Actors and Roles (adapted from the HL7 CTS 2 SFM).*

The following descriptions were adapted from the HL7 CTS2 SFM in order to adapt the epSOS context.

### 1.2.1 epSOS Terminology Access Services User

Considering an abstract epSOS user as the main actor that accesses and eventually manipulates the content of the epSOS Master Translation/Transcoding Catalogue (epSOS MTC), then primary specializations of this actor are the epSOS MTC administrator, the epSOS MTC provider and the epSOS MTC enabled application developer. Providers of the epSOS MTC form a specialized group of

classification authors/curators that create and manage the content of each coding system and value set in the epSOS MTC. Subspecializations of terminology content authors include value set developers that define pertinent and complete subsets of the epSOS MTC with respect to a particular domain, mappers that map or cross reference the content of all coding system to the epSOS Master Value Sets Catalogue (epSOS MVC) within the epSOS MTC jurisdictional domain, and translators that have the main role of finding and documenting semantically equivalent expressions of the mapped elements in epSOS paired member state languages.

### 1.2.2 epSOS Health Care Provider

The epSOS HCP, as the main beneficiary of the epSOS Terminology Access Services (TAS), is an indirect epSOS TAS user which relies on the accuracy and the performance of the system built by the epSOS TAS providers and application developers in order to offer him a translation of the pivot documents as complete and as precise as possible.

### 1.2.3 epSOS Terminology Administrator

The epSOS Terminology Administrator is an actor responsible for ensuring the availability and overall maintenance of the epSOS Terminology Access Services (epSOS TAS). This includes, but is not limited to loading content such as the epSOS MTC into the epSOS TAS, and making available the required functionality to address the specific conformance profiles implemented by the epSOS TAS instance.

### 1.2.4 epSOS Terminology Application Developer

The epSOS Terminology Enabled Application Developer is an actor who is responsible for the development of software applications that make explicit use of controlled terminologies making up the content of the epSOS MTC.

### 1.2.5 epSOS Terminology Provider

The epSOS Terminology Provider is the abstract actor that includes the individuals or organization responsible for the development of a particular terminology or controlled vocabulary.

### 1.2.6 epSOS Terminology Author/Curator

The epSOS Terminology Author / Curator is an actor who is responsible maintaining the content of the epSOS Master Translation/Transcoding Catalogue (epSOS MTC), including but not limited to, the development of new concepts and associations that may be submitted to the Terminology Provider or the extension of an existing terminology with local concepts. This may also determine who can validate and perform quality control of terminology content present in the epSOS MTC. Terminology Authors / Curators may not necessarily belong to the Terminology Provider's organization. This function is part of the work package 3.9 (Development of proof of concept system for pilot phase) and 3.10 (Proof of concept testing) until it will be transferred to an appropriate European entity at the go-live phases of the project. The identification of this entity remains an open issue for the work package as a whole.

### 1.2.7 epSOS Terminology Value Set Developer

The epSOS 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 which are part of the epSOS MTC.

### 1.2.8 epSOS Terminology Mapper

The epSOS 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 that are found in the epSOS MTC.

### 1.2.9 epSOS Terminology Translator

The epSOS 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. The act of translation is defined as finding semantic equivalent counterparts for nominal phrases of a source language (dialect) in a target language (dialect). In other words translation is a written communication in a second language (dialect) having the same meaning as the written communication in a first language (dialect). A call for the National Linguistic Competence Centre was made to identify the person responsible for this task.

## 1.3 epSOS Terminology Access Services Profiles

The current specification distinguishes a series of profiles (collaborations) derived from use case aggregates of the identified actors/roles, through abstraction of the main procedures for access, creation and maintenance of the epSOS Master Translation/Transcoding Catalogue (epSOS MTC). 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. The profiles described below enable the isolation and detailing of a number of necessary and sufficient interfaces to the epSOS MTC, identified in a service oriented approach, in analogy to the service functional model of the HL CTS 2 specification.

### 1.3.1 epSOS Runtime Translation Profile

At runtime the epSOS Terminology Access Service include functionality that enable the ‘ad-hoc’ translation of the data element values contained in the pivot documents (ePrescription, eDispensation and Patient Summary). This functionality requires a two step process: an accurate mapping of conceptual entities used in the source documents between the source coding system and epSOS Master Value Sets Catalogue (epSOS MVC), and the accurate translation of the concept designations (human readable surface representation) from the source language into the target language.

### 1.3.2 epSOS Query/Browsing Profile

The functionality needed in the epSOS Query/Browsing profile must include methods to search, find and access concepts and their attributes, based on some search, filtering or mapping criteria such as key words or key terms included in the designation of the concept or concept code. Methods should also include filtering specific concept associations or navigation of associations for retrieving a value set or a specific concept. The epSOS Query/Browsing profile is a main use case in the majority of applications and for all direct terminology services users.

### 1.3.3 epSOS Authoring/Curation Profile

The epSOS authoring/curation profile is characterized by functionality needed to directly maintain the content of the epSOS MTC. Terminology authors create, maintain, map and translate the content of coding system and/or value sets of a specific domain. They are heavily dependent on the epSOS Query/Browsing profile for validating the content of the epSOS MTC and making decisions upon the addition of new or refinement of existing elements (concepts, concept attributes, associations between concepts, or association types, etc.) or making decisions pertinent to the maintenance of concept hierarchies (ontological) structure within a coding system or value set. These functions

are generally protected and reserved for classification curators, value set developers, mappers and human language translators within a terminology provider institution. Curators of a controlled vocabulary may also be external to the institutional provider that releases the specific language corpus, terminology, thesaurus, classification or controlled vocabulary.

### 1.3.4 epSOS Administrative Profile

This profile includes functionality to manage content as part of a terminology service: load, export, activate and retire terminologies. These functions are generally protected and accessible by service administrators with appropriate authorization.

### 1.3.5 epSOS Request Profile

The profile covers functionality giving all terminology users possibility to propose changes of the terminologies involved in epSOS, i.e. epSOS reference system, external code systems and mappings between them. It enables to create requests for change, update existing requests, and for terminology administrators also to accept or reject the requests. Requests can be related to concepts, designations, associations and mappings, or to code systems as a whole.

# 2 epSOS Terminology Access Services Specification

The functional specification of the epSOS Terminology Access Services derives from the analyses of the use cases of each profile described in 1.2. Each profile incorporates functions that define an application programming interface (API).

## 2.1 epSOS Domain Data Model

The functional components of the epSOS Terminology Access Services will operate on a broad spectrum of European and international terminology sources that are developed with specific use cases and purposes in mind. The release format of various terminologies may range from a simple flat list of concepts to complex multiple inheritance concept hierarchies. The basic entities required for an epSOS Ontology model in general are exposed in the following diagram:

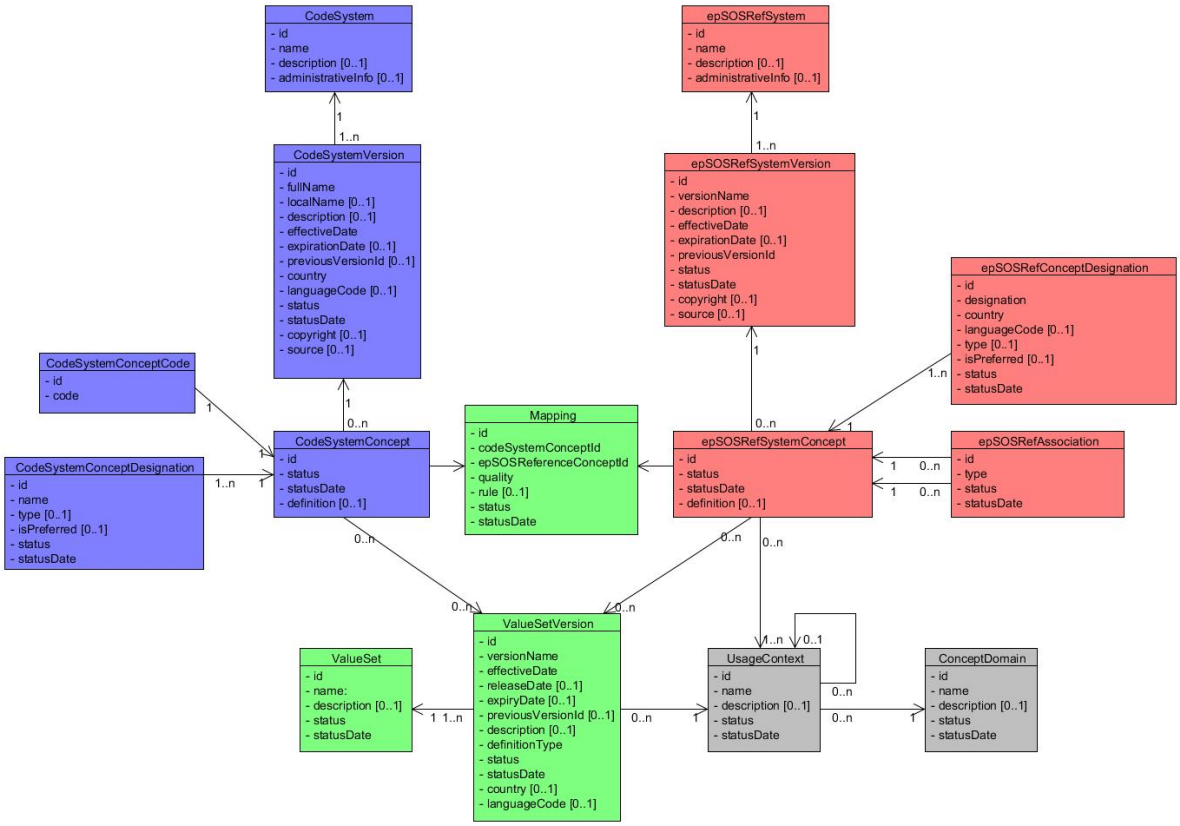



***Figure 2F*** *- The epSOS Ontology Model*

This model outlines the various logical entities that define the epSOS domain together with their attributes and usual cardinality between them. The purpose of the model is to illustrate the semantics of the key functional requirements of the epSOS Terminology Access Services. It is not intended to serve as an implementation blueprint. A detailed description of all composite data types resulting from the above domain model follows.

## 2.2 General Data Types

### 2.2.1 CodeSystem Type

A code system represents a collection of uniquely identifiable concepts with associated, representations, designations, associations, and meanings. Examples of code systems include ICD-9, ICD-10, SNOMED CT, LOINC, ATC, HL7, EDQM and many others. A given concept representation 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.

At a minimum, code systems have the following attributes:

- An identifier (**id**) that uniquely identifies the Code System. In HL7, this Id is in the form of an ISO OID.
- A name (**name**) that uniquely identifies the Code System.
- A description (**description**) that may include the code system uses and intent (optional).

Administrative information (**administrativeInfo**) is a placeholder to allow for additional information related to the overall Code System. This is separate from any specific version of the Code System (optional).

### 2.2.2 CodeSystemVersion Type

Although the actually value of a code will not change, the associated information may, e.g. associations, designations and concept properties. The Code System Version provides means to organize and manage variations to these elements over time. Typically (but not always), these variations will coincide with new versions of the owning code system.

Code System Versions are represented by attributes including:

- A version identifier (**id**) that uniquely identifies each version of a Code System. The full official name (**fullName**) of the Code System Version as assigned by the terminology provider at the time the version is being issued.
- A local name (**localName**), an alias of the **fullName** to which the Code system is normally referred to for the life of this version (optional).
- A **description** that may include the code system version uses and intent (optional).
- The start date (**effectiveDate**) when the version is valid for use.
- An expiration date (**expirationDate**) which marks the date when the version is no longer valid (optional).
- Identification of the previous version (**previousVersionId**) of the code system entity, which enables tracking of sequencing of versions, and identification of missing versions on a server instance (optional)..
- The country (**country**) where the Code System Version was issued.
- A language code (**languageCode**) of the official language in which the version was issued (optional). .
- A status (**status**) to identify the state of the Code System Version (mainly for curation)

- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).
- Copyright information (**copyright**) pertaining to the release (version) of the Code System (optional).
- An attribute (**source**) that identifies the authority or source of the code system in this version. i.e. IHTSDO, WHO etc (optional).

### 2.2.3 CodeSystemConcept Type

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. 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 their deletion.

Code System Concepts are represented by attributes including:

- A concept identifier (**id**) that uniquely identifies each concept of a Code System.
- A status (**status**) to identify the state of the Code System Version (mainly for curation).
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).
- A complete and correct definition of the concept viewed as a statement of the meaning of the concept represented by ‘genus proximum et differentia specifica’ (Aristoteles) (optional).

### 2.2.4 CodeSystemConceptCode Type

A Code System Concept Code defines a single "code value" that is used to represent a Code System Concept. A Code System Concept Code is the representation of a concept published by the author of a code system as part of the code system. It is intended to be used as the preferred unique identifier for that concept in that code system.
CodeSystemConceptCodes are represented by attributes including:

- A code identifier (**id**) that uniquely identifies each concept code of a Code System.
- A concept code (**conceptCode**) that according to terminology best practices is unique within the context of the Code System, although some code systems do allow reuse of codes over time.

Note on the difference between a "code" and a "designation". In many code systems the actual code value is also the identifier of the concept and is unique and definitional in nature within the context of a code system. Designations have no such restriction or intent and are merely alternate display formats.

### 2.2.5 CodeSystemConceptDesignation Type

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 type is minimally defined by the following attributes:

- A unique identifier (i**d**) for the designation.
- A name (**designation**) for the designation which is represented by a term or a complete nominal phrase. .
- A designation type (**type**) (optional).
- A 'preferred' attribute (**isPreferred**) that identifies whether a particular designation has a type of usage preference (optional).
- A status (**status**) to identify the state (mainly for curation).
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).

### 2.2.6 ValueSet Type

A Value Set represents a uniquely identifiable set of valid concept representations (codes and/or designations), 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 (**designation**) for the value set
- A description (**description**) for the value set (optional)
- A status (**status**) to identify the state (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)

### 2.2.7 ValueSetVersion Type

A Value Set Version represents a point in time view of a Value Set definition. The Value Set Version identifies the set of concepts that are available in the value set for any specific version of the

value set definition. The versioning of the value set reflects the fact that the value set definition may change over time. Value Set Versions can be created by aggregation of other value set versions.

A Value Set Version has the following properties:

- A Value Set Version identifier (**id**) that uniquely identifies the value set version.
- A Value Set Version name (**versionName**) that uniquely identifies the value set version.
- An effective date (**effectiveDate**) defined as the point in time when the value set version usage becomes valid.
- A release date (**releaseDate**) when the value set version was published or issued (optional).
- An expiration date (**expirationDate**) defined as the point in time when the value set version usage is no longer valid (optional).
- An identifier of the previous value set version (**previousVersionId**) (optional)
- A description (description) of the value set version that may include purpose and context usage of the value set version (**optional**).
- A definition type (**definitionType**) attribute, indicating the way (extensional or intensional) the value set version was created.
- A status (**status**) to identify the state (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)
- The country (**country**) where the Value Set Version was issued (optional)
- A language code (**languageCode**) of the official language in which the Value Set Version was issued (optional)

### 2.2.8 UsageContext Type

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. "Patient Summary", “ePrescription”, or “eDispensation” for example. The Usage Context has the following attributes:

- An identifier (**id**) that uniquely identifies the Usage Context.
- A name (**name**) that the Usage Context is normally referred to and uniquely identifies the Usage Context.
- A description (**description**) of the Usage Context that may include scope and purpose of the Usage Context (optional).
- A status (**status**) to identify the state (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)

### 2.2.9 ConceptDomain Type

Concept domains describe a semantic space. They are a named category of like concepts (a semantic type) that will be bound to one or more attributes in a static model. Concept Domains exist to specify the intent of the coded element while deferring the association of the element to a specific coded terminology until later in the model development process.

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 and is uniquely identified by.
- A description (description) that may include scope and purpose of the Concept Domain (optional).
- A status (**status**) to identify the state (mainly for curation).
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).

## 2.3 epSOS Specific Data Types

Similar data types are defined in the epSOS context for the purpose of defining the 'ad-hoc translational' functionality needed.

### 2.3.1 epSOSRefSystem Type

An epSOS Reference System type is a complex type, similar to the Code System type and represents the collection of uniquely identifiable concepts needed in the epSOS context, with their affiliated designations, associations, and meanings.

The epSOS Reference System has the following attributes:

- An identifier (**id**) that uniquely identifies the epSOS Reference System (OID).
- A name (**name**) that uniquely identifies the epSOS Reference System.
- A description (**description**) that may include the epSOS Reference System uses and intent. (optional).

Administrative information (**administrativeInfo**) which is a placeholder to allow for addi-tional information relating to the overall Code System, that is separate from any specific version of the epSOS Reference System (optional).

### 2.3.2 epSOSRefSystemVersion Type

An epSOS Reference System Version Type is a complex type, similar to the Code System Version type.

Attributes of an epSOS Reference System Version include:

- A version identifier (**id**) that uniquely identifies each version of an epSOS Reference System Version.
- The version name (**versionName**) of the epSOS Reference System Version as assigned by the epSOS terminology provider at the time the version is being issued.
- A **description** that may include the code system version uses and intent (optional).

- The start date (**effectiveDate**) when the version is valid for use.
- An expiration date (**expirationDate**) which marks the date when the version is no longer valid (optional).
- Identification of the previous version (**previousVersionId**) of the code system entity, which enables tracking of sequencing of versions, and identification of missing versions on a server instance.
- A status (**status**) to identify the state of the Code System Version (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)
- Copyright information (**copyright**) pertaining to the release (version) of the Code System (optional)
- An attribute (source) that identifies the authority or source of the code system in this version. i.e. IHTSDO, WHO etc. (optional)

### 2.3.3 epSOSRefSystemConcept Type

An epSOS Reference System Concept type is a complex type, similar to the Code System Concept type.

Attributes of an epSOS Reference System Concept include:

- A concept identifier (**id**) that uniquely identifies each concept of the epSOSRefSystem
- A status (**status**) to identify the state of the concept (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)
- A complete and correct **definition** of the concept viewed as a statement of the meaning of the concept represented by ‘genus proximum et differentia specifica’ (Aristoteles) (optional).

### 2.3.4 epSOSRefConceptDesignation Type

An epSOS Reference Concept Designation type is a complex type, similar to the Code System Concept Designation type.

Attributes of an epSOSRefConceptDesignation include:

- A unique identifier (**id**) for the designation
- A **designation** which is represented by a term or a complete nominal phrase.
- The country (**country**) where the designation is used and has a particular meaning
- The language code (**languageCode**) of the designation
- A designation type (**type**) (optional).
- A ‘preferred’ attribute (**isPreferred**) that identifies whether a particular designation has a type of usage preference (optional).
- A status (**status**) to identify the state (mainly for curation).

- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).

### 2.3.5 epSOS Mapping Type

A Mapping type is a complex type that represents the semantic equivalence assigned between an epSOS Reference System Concept and a Code System Concept of another vocabulary, classification or terminology.

The Mapping attributes include:

- A unique identifier (**id**) for the mapping.
- The **codeSystemConceptId** of a particular **CodeSystemVersion**.
- The semantic equivalent **epSOSRefConceptId** of the above **CodeSystemConceptId**.
- A **quality attribute** that describes if the semantic equivalence is given or if there are granularity differences still to be resolved ('narrower', 'broader', 'equivalent' attributes).
- A **mapping rule** (optional).
- A status (**status**) to identify the state (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation).

### 2.3.6 epSOSRefAssociation Type

An epSOS Reference System Association is a complex type that defines associations between epSOS Reference System Concepts. Associations are defined as directed semantic relationship triples, involving the association and the 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.

epSOS Reference System Association are minimally defined by:

- A unique identifier (**id**) for the association.
- A **association type**
- A status (**status**) to identify the state (mainly for curation)
- A status date (**statusDate**) to identify the date the status was set to its current value (for curation)

### 2.3.7 epSOSRequest Type

A Request is a complex type representing any textual description of terminology user needs and requests for change of either epSOS reference system or external code system content. Using these requests it could be asked for creation, modification or removal of :

- epSOS Reference System Concept
- epSOS Reference System Concept Designation (Country, Language)
- epSOS Reference System Association (Concept1, Concept2) <- Association

- Code System Concept (Code System)
- Code System Concept Designation (Code System Concept)
- Code System (Country, Language)
- Value Set
- Value Set Concept (Value Set)
- Mapping between epSOS Reference System Concept and Code System Concept (Concept1, Concept2)
- Other unspecified request.

Request is used to transfer information between terminology users and terminology provider. If terminology user considers there is a need to change epSOS MTC or Ontology, he/she **creates** a request describing required change and sends it to terminology provider. Terminology provider has to decide, whether the request is justified and if so, to **approve** the request. Otherwise the request will be **cancelled**. In case of approved request, the change is usually not put into effect immediately (e.g. it will be released in next version). For terminology user to be informed about actual execution of required change, the request is marked as **implemented**. Furthermore, if author of the request decides to recall it, he may **cancel** the request.

Life-cycle of the request is illustrated in figure 3F:

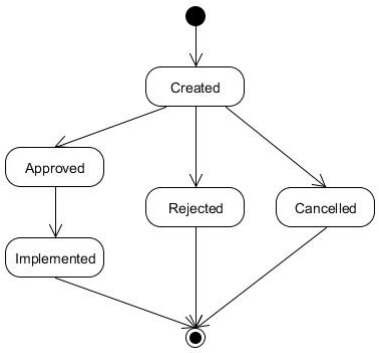



***Figure 3F** – Request life-cycle*

Request is intended to be created by any terminology user (e.g. Healthcare provider, application developer or value set developer). However, the request may be resolved and implemented only by terminology provider. Intermediate request statuses (such as 'pending clarification') will be needed to track a request lifecycle.

Domain model of the epSOS Request type is in figure:

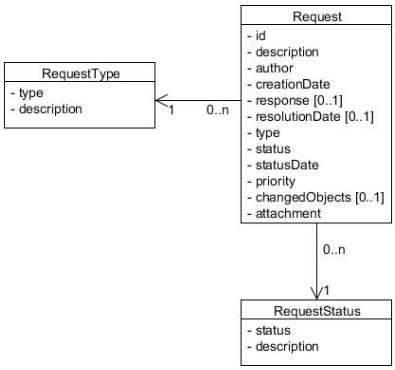



***Figure 4F** – Domain model for Request type*

Attributes of request include:

- A identifier (**id**) that uniquely identifies each request.
- A detailed textual description (**description**) of the request in form of plain text.
- An user (**author**) who created the request.
- A date (**creationDate**) when the request was created.
- A detailed textual description (**response**) of reply from terminology provider. It may contain either ways of request implementation or reasons for its rejection (optional).
- A resolution date (**resolutionDate**) which marks the date when the version is no longer valid (**optional**).
- A request type (**type**) expressing to what kind of object request is related.
- A status (**status**) to identify the state of the request.
- A status date (**statusDate**) to identify the date the status was set to its current value.
- A **priority** indicating how urgent the resolution of the request is.
- A list of objects (**changedObjects**) pointing to all objects influenced by request implementation (**optional**).
- An attachment containing any additional files related to the request, etc. code system for import (optional).

# 3 Standard epSOS TAS Interfaces Descriptions

The epSOS Reference System interacts with third party applications or services through a set of standard interfaces described below. In the design of the individual interfaces the functionality described in the HL7-OMG CTS2.0 specification has been adopted and completed where appropriate.

The four CTS2.0 standard interfaces (Administration, Query/Browse, Authoring/Curation and Association) where adapted to the requirements and specifics of the epSOS domain. Additional interfaces for the purpose of 'ad-hoc' run-time translation and request processing have been added to the specification. A synopsis of the epSOS standard interfaces is exposed in the following figure (interfaces in red where added to the standard CTS2.0 specification in the epSOS context):

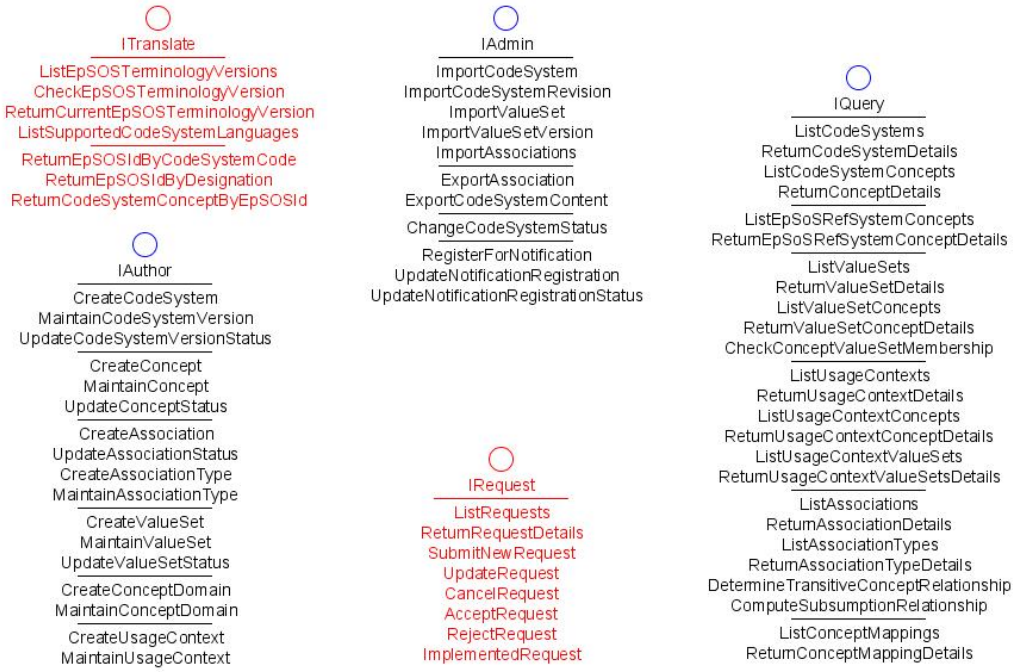



***Figure 5F*** *- The epSOS Standard Interfaces*

Each individual epSOS semantic service standard interface, along with a synopsis of the explicitly contained function signatures is described below.

## 3.1 Runtime Translation API (ITranslate Interface)

**Synopsis:**


### 3.1.1 List epSOS Reference System Versions

| | |
|---|---|
| **Function Signature** | + getEpSOSRefSystemVersions():epSOSRefSystemVersionId[] |
| **Description** | Retrieves a list of available versions of the epSOS reference system |
| **Inputs** | |
| **Outputs** | List of available versions of the epSOS reference system. |
| **Preconditions** | epSOS Terminology Access Services up and running<br>Content has been imported into the reference system |
| **Postconditions** | |
| **Exceptionalbehaviour** | No content has been imported to the epSOS reference system |
| **Comments** | |

### 3.1.2 Check EpSOS Reference System Version

| | |
|---|---|
| **Function Signature** | + isEpSOSRefSystemVersion(v : epSOSRefSystemVersionId):Boolean |
| **Description** | Validates if a particular version is present in the epSOS reference system |
| **Inputs** | epSOS reference system version identifier to be validated |
| **Outputs** | True if the epSOS reference system version is present, false otherwise |
| **Preconditions** | epSOS Terminology Access Services up and running<br>Content has been imported into the reference system |
| **Postconditions** | |
| **Exceptional behaviour** | No content has been imported to the epSOS reference system |

| | |
|---|---|
| **Comments** | |

### 3.1.3 Return Current epSOS Terminology Version

| | |
|---|---|
| **Function Signature** | +getCurrentEpSOSRefSystemVersion():epSOSRefSystemVersionId |
| **Description** | Retrieves the current epSOS reference system version identifier |
| **Inputs** | |
| **Outputs** | Current epSOS reference system version identifier |
| **Precondition** | epSOS Terminology Access Services up and running<br>Content has been imported into the reference system |
| **Postconditions** | |
| **Exceptional behaviour** | Unable to retrieve the current reference system version<br>No content has been imported to the epSOS reference system |
| **Comments** | |

### 3.1.4 List Supported CodeSystem Languages

| | |
|---|---|
| **Function Signature** | + getSupportedCodeSystemLanguages(c : CodeSystemId) : LanguageCode[] |
| **Description** | Retrieves a list of supported languages for a particular CodeSystem in the epSOS reference system |
| **Inputs** | Code System ID |
| **Outputs** | List of supported languages for the input Code System |
| **Precondition** | epSOS Terminology Access Services up and running<br>Content has been imported into the reference system |
| **Postconditions** | |
| **Exceptional behaviour** | Input CodeSystemId does not exist<br>No content has been imported to the epSOS reference system |
| **Comments** | |

### 3.1.5 Return epSOSId by CodeSystemCode

| | |
|---|---|
| **Function Signature** | + getEpSOSIdByCode(c : CodeSystemConceptCodeId, v : CodeSystemVersionId) : epSOSRefConceptId |
| **Description** | Retrieve the epSOS Concept Id for a particular Code System Concept Code |
| **Inputs** | Code System Concept Code Id<br>Code System Version Id |

| | |
|---|---|
| **Outputs** | epSOS Reference System Concept Id |
| **Precondition** | epSOS Terminology Access Services up and running<br>The CodeSystem and CodeSystemVersion contents have been imported into the reference system. |
| **Postconditions** | |
| **Exceptional behaviour** | CodeSystemConceptCodeId does not exist<br>CodeSystemVersionId does not exist<br>CodeSystemConceptCodeId is invalid for CodeSystemVersionId |
| **Comments** | |

### 3.1.6 Return epSOSId by Designation

725

| | |
|---|---|
| **Function Signature** | +getEpSOSIdByDesignation(d : DesignationId, v : CodeSystemVersionId) : epSOSRefConceptId |
| **Description** | Retrieve the epSOS Concept Id for a particular Code System Concept Designation |
| **Inputs** | Code System Concept Designation Id<br>Code System Version Id |
| **Outputs** | epSOS Reference System Concept Id |
| **Precondition** | epSOS Terminology Access Services up and running<br>The CodeSystem and CodeSystemVersion contents have been imported into the reference system. |
| **Postconditions** | |
| **Exceptional behaviour** | DesignationId does not exist<br>CodeSystemVersionId does not exist<br>DesignationId is invalid for CodeSystemVersionId |
| **Comments** | |

### 3.1.7 Return CodeSystemConcept by epSOSId

| | |
|---|---|
| **Function Signature** | +getConceptByEpSOSId(i:epSOSRefConceptId) : CodeSystemConcept |
| **Description** | Retrieve the CodeSystemConcept by epSOSId |
| **Inputs** | epSOS Reference Concept Id |
| **Outputs** | Code System Concept |
| **Precondition** | epSOS Terminology Access Services up and running<br>The CodeSystem and CodeSystemVersion contents have been imported into the reference system. |
| **Postconditions** | |
| **Exceptional behaviour** | epSOS Reference Concept Id does not exist |
| **Comments** | |

To demonstrate the usage of specified functions from ITranslate profile figure 6F illustrates the se-
730 quence corresponding to main epSOS use case. The “on-the-fly” translation of pivot documents shows the same process as Semantic Content Workflow described in the document *Semantic Services Definition D3.5.2 in figure 2.*

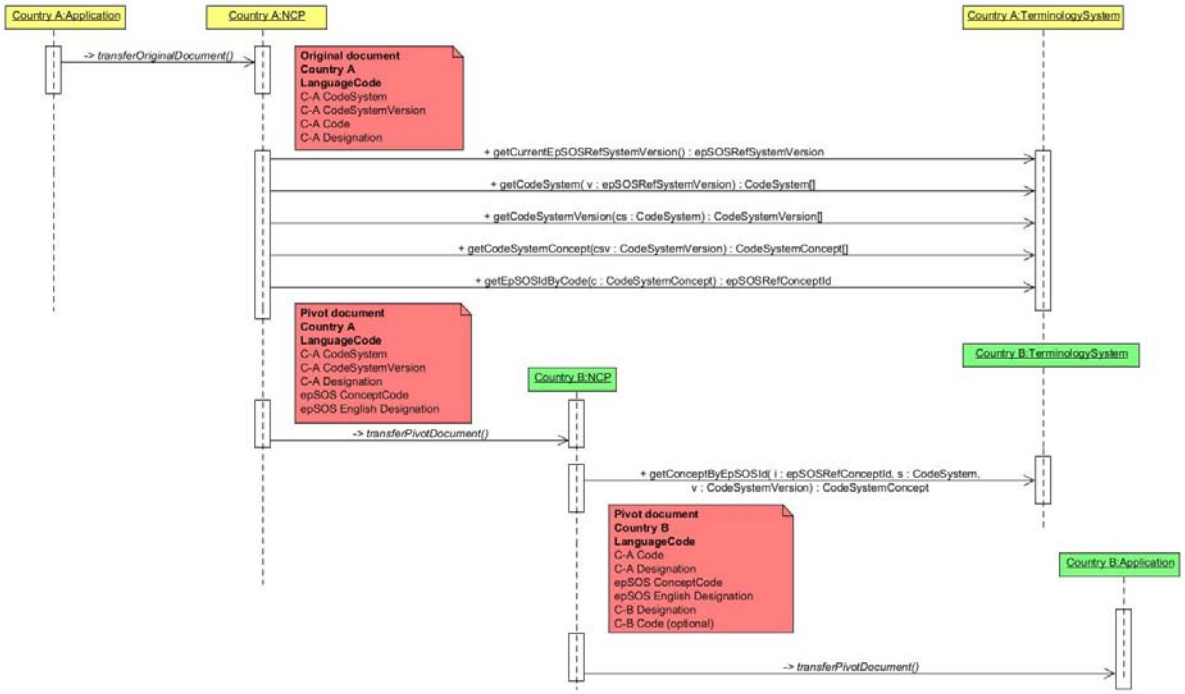



***Figure 6F*** *– Semantic Content Workflow*

## 3.2 Administrator API (IAdministration Interface)

**Synopsis:**


### 3.2.1 Import Code System

| | |
|---|---|
| **Function Signature** | +importCodeSystem (s : CodeSystem, [v : CodeSystemVersion, [s : Source, f : Format]]) : Boolean |
| **Description** | Installs a code system (terminology) into the epSOS semantic service reference system 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, associationss and so on, or some of these elements may be loaded using the Import Code System Revision operation. The actual contents may be supplied by value or reference, i.e. as a complete set of explicit content or as a reference to a location where the content can be separately obtained for loading. |
| **Inputs** | Code System<br>Code System Version (optional)<br>Code System Source (i.e. either: URI or File with Concepts, ConceptAssociations, Designations etc.) (optional)<br>Code System Format (ClaML) (optional) |
| **Outputs** | An acknowledgment indicating whether the terminology has been successfully loaded or not. |
| **Precondition** | Code System source is available in a format directly consumable by the import tools. |
| **Postconditions** | The code system is available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Code system not found at source location<br>Supplied contents cannot be processed (a number of specific errors can be defined) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

### 3.2.2 Import Code System Revision

| | |
|---|---|
| **Function Signature** | + importCodeSystemRevision ( i : CodeSystemId, v : CodeSystemVersion, s : Source, f : Format) : Boolean |
| **Description** | Installs either an entire new Code System Version or the necessary revision updates for an already loaded Code System Version into the epSOS semantic service reference system (content included by value or by reference to a location). Includes indicator as to whether intent is to replace whole code system or just replace some elements (codes, associations etc) |
| **Inputs** | Code System Id<br>Code System Version<br>Code System Version Source (i.e. either: URI or File with Concepts, ConceptAssociations, Designations etc.)<br>Code System Format (ClaML) |
| **Outputs** | An acknowledgment indicating whether the Code System Revision has been successfully loaded or not. |
| **Precondition** | Code System revision source is available in a format directly consumable by the import tools |
| **Postconditions** | The code system is available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Code System does not exist.<br>Code System Version not found at source location<br>Supplied contents cannot be processed (a number of specific errors can occur) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

745

### 3.2.3 Import Value Set

| | |
|---|---|
| **Function Signature** | + importValueSet ( v : ValueSet, [ver : ValueSetVersion, [s : Source, f : Format]]) : Boolean |
| **Description** | Installs a value set into the epSOS semantic service reference system for subsequent access by other service functions. This operation may also be used for the initial installation of a value set version. The value set contents can only be explicitly provided, intensional value sets (where the value set is defined as a computable expression) are not supported. |
| **Inputs** | Value Set<br>Value Set Version (optional)<br>Value Set Source (i.e. either: URI or File with Concepts, ConceptAssociations, Designations etc.) (optional)<br>Value Set Format (optional) |
| **Outputs** | An acknowledgment indicating whether the value set has been successfully loaded or not. |
| **Precondition** | Value set source is available in a format directly consumable by import tools. |
| **Postconditions** | The value set is available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Value set not found at source location |

| | |
|---|---|
| | Supplied contents cannot be processed (a number of specific errors can occur) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

### 3.2.4 Import Value Set Version

| | |
|---|---|
| **Function Signature** | + importValueSetVersion ( v : ValueSetId, ver : ValueSetVersion, s : Source, f : Format) : Boolean |
| **Description** | Installs a value set version into the epSOS semantic service reference system for subsequent access by other service functions. The value set contents can only be explicitly provided; intensional value sets (where the value set is defined as a computable expression) are not supported. |
| **Inputs** | Value Set Id<br>Value Set Version<br>Value Set Version Source (i.e. either: URI or File with Concepts, ConceptAssociations, Designations etc.)<br>Value Set Version Format |
| **Outputs** | An acknowledgment indicating whether the value set has been successfully loaded or not. |
| **Precondition** | Value set source is available in a format directly consumable by import tools. |
| **Postconditions** | The value set is available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Value Set does not exist.<br>Value Set Version not found at source location<br>Supplied contents cannot be processed (a number of specific errors can occur) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

### 3.2.5 Import Associations

755

| | |
|---|---|
| **Function Signature** | + importAssociations( a : epSOSRefSystemAssociation[])<br>+ importAssociations(s : Source, f : Format) |
| **Description** | Installs a set of associations into the epSOS semantic service reference system for subsequent access by other service functions. This may include the full set of associations between concepts from the epSOS reference system. The actual associations may be supplied by value or reference, i.e. as a complete set of explicit association or as a reference to a location where the associations can be separately obtained for loading. |
| **Inputs** | Array of Associations<br>OR<br>Associations Source<br>Associations Format |
| **Outputs** | An acknowledgment indicating whether the associations have been successfully loaded or not. |
| **Precondition** | Association source is available in a format directly consumable by import tools. |
| **Postconditions** | Associations are available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Association not found at source location<br>Supplied contents cannot be processed (a number of specific errors can occur) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

### 3.2.6 Export Associations

| | |
|---|---|
| **Function Signature** | + exportAssociations(v : epSOSRefSystemVersion, t : Target, f : Format, [r : Filter]) : Boolean |
| **Description** | Exports association instances between the epSOS reference System Concepts, eventually applying filter criteria |
| **Inputs** | Version of epSOS Reference SystemExport target URI or file<br>Export format<br>Export filter criteria (optional) |
| **Outputs** | Requested association instances of the code system.<br>An acknowledgment indicating whether the associations have been successfully exported or not. |
| **Precondition** | epSOS Semantic Service installed and running<br>Code System is available for querying in the epSOS reference system<br>Association instances are available for querying in the epSOS reference system |
| **Postconditions** | The association instances are exported to the required location or returned to the requestor |
| **Exceptional behaviour** | Version of epSOS Reference System does not exist |

| | |
|---|---|
| | Association instances do not exist<br>Invalid target/format/[filter criteria] |
| **Comments** | |

### 3.2.7 Import Mappings

| | |
|---|---|
| **Function Signature** | + importMappings( m : Mapping[])<br>+ importMappings(s : Source, f : Format) |
| **Description** | Installs a set of mappings into the epSOS semantic service reference system for subsequent access by other service functions. This may include the full set of mappings between concepts from the epSOS reference system to the epSOS reference system. The actual mappings may be supplied by value or reference, i.e. as a complete set of explicit mappings or as a reference to a location where the mappings can be separately obtained for loading. |
| **Inputs** | Array of Mappings<br>OR<br>Mappings Source<br>Mappings Format |
| **Outputs** | An acknowledgment indicating whether the mappings have been successfully loaded or not. |
| **Precondition** | Mapping source is available in a format directly consumable by import tools. |
| **Postconditions** | Mappings are available for access via the epSOS semantic service functions. |
| **Exceptional behaviour** | Mapping not found at source location<br>Supplied contents cannot be processed (a number of specific errors can occur) (Information pertaining to all failures is logged and reported for analysis and serviceability.) |
| **Comments** | |

### 3.2.8 Export Mappings

| | |
|---|---|
| **Function Signature** | + exportMappings(i : CodeSystemId, t : Target, f : Format, [r : Filter]) : Boolean |
| **Description** | Exports mapping instances between a code system and the epSOS reference System Concepts, eventually applying filter criteria |
| **Inputs** | Code System Id<br>Export target URI or file<br>Export format<br>Export filter criteria (optional) |
| **Outputs** | Requested mapping instances of the code system.<br>An acknowledgment indicating whether the mappings have been successfully exported or not. |
| **Precondition** | epSOS Semantic Service installed and running |

| | |
|---|---|
| | Code System is available for querying in the epSOS reference system<br>Mapping instances are available for querying in the epSOS reference system |
| **Postconditions** | The mapping instances are exported to the required location or returned to the requestor |
| **Exceptional behaviour** | Code System does not exist<br>Mapping instances do not exist<br>Invalid target/format/[filter criteria] |
| **Comments** | |

765

### 3.2.9 Export Code System Content

| **Function Signature** | + exportCodeSystem(i : CodeSystemId, t : Target, f : Format, [r : Filter]) : Boolean |
|---|---|
| **Description** | Exports a code system, subset criteria (specific set of concepts and/or associations/maps from the epSOS reference system by filtering the content and converting to the requested format for export. |
| **Inputs** | Code System Id<br>Export target URI or file(s)<br>Export format<br>Export filter criteria (optional) |
| **Outputs** | Requested code system artifacts exported.<br>An acknowledgment indicating whether the code system artifacts have been successfully exported or not. |
| **Precondition** | epSOS Semantic Service installed and running<br>Code System source is available for export |
| **Postconditions** | The code system is exported to the required location or is returned to the requestor |
| **Exceptional behaviour** | Code System does not exist<br>Invalid target/format/[filter criteria] |
| **Comments** | |

770

### 3.2.10 Change Code System Status

| **Function Signature** | + setCodeSystemStatus(i : CodeSystemId, s : Status) : Boolean |
|---|---|
| **Description** | Changes the state of a code system. (Includes inactivation, activation etc.) This allows an epSOS Reference System Administrator to manage the state of a given code system, thus managing the level of availability for access by other service functions. |
| **Inputs** | Code System Id<br>Code System Status |
| **Outputs** | An acknowledgment indicating whether the state of the code system has been successfully changed. |

| | |
|---|---|
| **Precondition** | Code system available in the epSOS Semantic Service reference system |
| **Postconditions** | The code system source state is changed to the requested value. |
| **Exceptional behaviour** | Code System does not exist<br>Code System state transition not allowed<br>Invalid Status |
| **Comments** | |

775

### 3.2.11 Register for Notification

| | |
|---|---|
| **Function Signature** | + addNotificationRegistration(a : Address, t : TargetType, I : TargetId, v : TargetVersion, d : Directions) : Boolean, NotificationId |
| **Description** | Register to be notified whenever a vocabulary element (code system, value set, concept or association) 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 to which the notification of the modification of the terminology term is sent to.<br>2. Notification Target Type (Code System, Value Set, Concept, Concept Association)<br>3. Notification Target Identifier<br>4. Notification Target Version<br>5. Notification Directions |
| **Outputs** | An acknowledgment indicating whether the terminology element notification request was successfully created.<br>Notification Identifier |
| **Precondition** | User or appropriate proxy (system administrator, etc) are authorized to access registry |
| **Postconditions** | A notification record has appropriately been added. |
| **Exceptional behaviour** | Identified element / version does not exist<br>Notification Directions not recognized |
| **Comments** | |

780 **3.2.12 Update Notification Registration**

| | |
|---|---|
| **Function Signature** | + updateNotificationRegistration(i : NotificationId; a : Address, t : TargetType, I : TargetId, v : TargetVersion, d : Directions) : Boolean |
| **Description** | Revises a notification entry for a particular vocabulary element. |
| **Inputs** | 1. Notification Entry Identifier<br>2. URL or other electronic address to which the notification of the modification of the terminology term is sent to.3. Notification Target Type (Code System, Value Set, Concept, Concept Association) |

| | |
|---|---|
| | 4. Notification Target Identifier<br>5. Notification Target Version<br>6. Notification Directions |
| **Outputs** | An acknowledgment indicating whether the terminology element notification revision request was successfully processed. |
| **Precondition** | User or appropriate proxy (system administrator, etc) are authorized to access registry |
| **Postconditions** | Notification records are updated appropriately. |
| **Exceptional behaviour** | 1. Notification Entry Identifier does not exist.<br>2. Identified element / version does not exist<br>3. Notification Directions not recognized |
| **Comments** | |

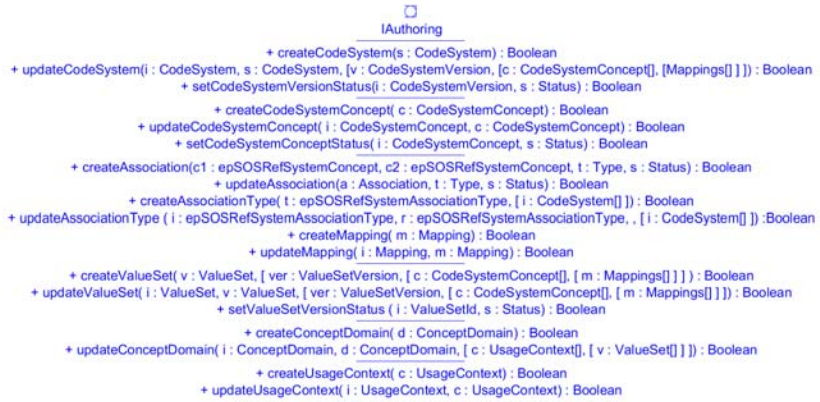
### 3.2.13 Update Notification Registration Status

785

| | |
|---|---|
| **Function Signature** | + updateNotificationRegistrationStatus (i : NotificationId, s : NotificationStatus) : Boolean |
| **Description** | Changes the status of a notification entry for a particular vocabulary element (suspend, reinstate, cancel, remove). |
| **Inputs** | 1. Notification Entry Identifier<br>2. Notification Status |
| **Outputs** | An acknowledgment indicating whether the notification status revision request was successfully processed. |
| **Precondition** | User or appropriate proxy (system administrator, etc) are authorized to access registry |
| **Postconditions** | Notification status is updated appropriately. |
| **Exceptional behaviour** | 1. Notification Entry Identifier does not exist.<br>2. Invalid state change |
| **Comments** | |

## 3.3 Authoring/Curation API (IAuthoring Interface)

**Synopsis:**




790

### 3.3.1 Create Code System

| | |
|---|---|
| **Function Signature** | + createCodeSystem(s : CodeSystem) : Boolean, CodeSystemId |
| **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** | Code System instance |
| **Outputs** | 1. An acknowledgment indicating whether the code system was created or not.<br>2. Code System Id, if the code system was successfully created. |
| **Precondition** | |
| **Postconditions** | The code system is available for access via the epSOS Terminology Access Services functions. |
| **Exceptional behaviour** | Code System already exists.<br>Invalid Code System instance |
| **Comments** | If the user wants to specify the Code System Id manually, this can be set as a "Code System Property".<br>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. |

795

### 3.3.2 Maintain Code System Version

| | |
|---|---|
| **Function Signature** | + updateCodeSystem(i : CodeSystemId, s : CodeSystem, [v : CodeSystemVersion, [c : CodeSystemConcept[], [Mappings[]]]) |

| | |
|---|---|
| **Description** | Update Code System meta-data properties and add concepts and properties to code system (Code System Supplement). |
| **Inputs** | Code System Id of the Code System to be updated<br>Code System (with updated properties)<br>Code System Version (with updated properties) (optional)<br>Code System Concepts in supplement (optional)<br>Mappings in supplement (optional) |
| **Outputs** | An acknowledgment indicating whether the code system supplement was updated or not. |
| **Precondition** | Code System to be updated exists<br>Code System Version to be updated exists |
| **Postconditions** | Supplement updated. |
| **Exceptional behaviour** | Code System does not exist<br>Code System version does not exist.<br>Invalid Code System properties<br>Invalid Code System Version properties<br>Invalid Concept properties<br>Invalid Mappings properties |
| **Comments** | This allows for maintenance of Code Systems Supplement properties, including creation and updating of Code System Versions, and assignment of Concepts and their properties to a Supplement. |

800

### 3.3.3 UpdateCodeSystemVersionStatus

| | |
|---|---|
| **Function Signature** | + setCodeSystemVersionStatus(i : CodeSystemVersionId, s : Status) |
| **Description** | Changes the status of a code system version (suspend, reinstate, cancel, remove). |
| **Inputs** | Code System Version Id<br>Status |
| **Outputs** | An acknowledgment indicating whether the code system status was updated or not. |
| **Precondition** | Code System Version exists |
| **Postconditions** | Status of the Code System Version is updated appropriately |
| **Exceptional behaviour** | Code System Version does not exist |
| **Comments** | |

### 3.3.4 Create Concept

805

| | |
|---|---|
| **Function Signature** | + createCodeSystemConcept( c : CodeSystemConcept) : Boolean, CodeSystemConceptId |

| | |
|---|---|
| **Description** | Create concept to be included in a Code System (epSOSRefSystem). The new concept is defined by the set of meta-data properties that describe it. |
| **Inputs** | CodeSystemConcept (epSOSRefSystemConcept) instance |
| **Outputs** | An acknowledgment indicating whether the concept was created or not.<br>Code System Concept Id (epSOSRefSystemConceptId) |
| **Precondition** | A sequencer that provides Concept Ids must be available if the Concept Id is not provided as an input. (default incremental Concept Id value) |
| **Postconditions** | The concept is available in the code system and is available for access via the epSOS Terminology Access Services functions. |
| **Exceptional behaviour** | Code System does not exist.<br>Code System version does not exist.<br>Concept already exists. |
| **Comments** | |

### 3.3.5 Maintain Concept

| | |
|---|---|
| **Function Signature** | + updateCodeSystemConcept( i : CodeSystemConceptId, c : CodeSystemConcept) : Boolean |
| **Description** | Update Concept meta-data properties. |
| **Inputs** | Code System Concept Id of the concept to be updated<br>Code System Concept (with updated properties) |
| **Outputs** | An acknowledgment indicating whether the concept was updated or not. |
| **Precondition** | |
| **Postconditions** | The concept is updated appropriately. |
| **Exceptional behaviour** | Code System Concept does not exist.<br>Invalid Code System Concept Properties<br>Invalid or Duplicate Code System Concept Code<br>Invalid or Duplicate Code System Concept Designation |
| **Comments** | |

### 3.3.6 Update Concept Status

| | |
|---|---|
| **Function Signature** | + setCodeSystemConceptStatus( i : CodeSystemConceptId, s : Status) : Boolean |
| **Description** | Changes the status of a code system concept (epSOSRefSystemConcept) (suspend, reinstate, cancel, remove). |
| **Inputs** | Code System Concept Id<br>Status |
| **Outputs** | An acknowledgment indicating whether the status revision request was successfully processed. |
| **Precondition** | Code System Concept exists. |
| **Postconditions** | Code System Concept status is updated appropriately. |
| **Exceptional behaviour** | Code System Concept does not exist |
| **Comments** | |

### 3.3.7 Create Association

| | |
|---|---|
| **Function Signature** | + createAssociation(c1 : epSOSRefSystemConcept, c2 : epSOSRefSystemConcept, t : Type, s : Status) : Boolean |
| **Description** | Create a new association between two epSOS Reference System Concepts |
| **Inputs** | Two epSOS Reference System Concepts (order of arguments determine association direction)<br>Type<br>Status |
| **Outputs** | An acknowledgment indicating whether the association was created or not.<br>The generated AssociationId. |
| **Precondition** | |
| **Postconditions** | The association is available for use via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Association already exists.<br>Invalid Association properties |
| **Comments** | |

### 3.3.8 Maintain Association

| | |
|---|---|
| **Function Signature** | + updateAssociation( a : Association, t : Type, s ; Status) : Boolean |
| **Description** | Update association properties. |
| **Inputs** | AssociationIdId of the association that is being updatedType |

| | |
|---|---|
| | Status |
| **Outputs** | An acknowledgment indicating whether the association was updated or not. |
| **Precondition** | |
| **Postconditions** | The updated association is available for use via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Invalid Association Type properties |
| **Comments** | |

### 3.3.9 Create Association Type

| | |
|---|---|
| **Function Signature** | + createAssociationType( t : epSOSRefSystemAssociation, [ i : CodeSystemId[] ]) : Boolean, epSOSRefSystemAssociationId |
| **Description** | Create a new association type (as intended by the association type class of the conceptual model), an instance of which 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 in the epSOSRefSystem. |
| **Inputs** | epSOSRefSystemAssociation<br>List of zero or more Code System Ids |
| **Outputs** | An acknowledgment indicating whether the association type was created or not.<br>The generated epSOSRefSystemAssociationId, if not supplied. |
| **Precondition** | |
| **Postconditions** | The association type is available for use via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Association Type already exists.<br>Invalid Association Type properties<br>Code System Id does not exist |
| **Comments** | Creating Associations types which can span across two different code systems is permitted by the interface, but should be used with care. It is not certain if semantic Associations 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. |

825

### 3.3.10 Maintain Association Type

| | |
|---|---|
| **Function Signature** | + updateAssociationType ( i : epSOSRefSystemAssociationId, r : epSOSRefSystemAssociation, , [ i : CodeSystemId[] ]) :Boolean |
| **Description** | Update or deprecate an Association type that may be used to link two concepts. |
| **Inputs** | Association Type Id |

| | |
|---|---|
| | Association Type instance (with updated properties)<br>List of zero or more Code System Ids |
| **Outputs** | An acknowledgment indicating whether the Association type was modified or not. |
| **Precondition** | |
| **Postconditions** | The updated Association type is available for access via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Association Type does not exist.<br>Invalid Association Type properties<br>Unrecognized Code System ID |
| **Comments** | Status change could be a separate operation, but seems unnecessary at this "meta" level. |

### 3.3.11 Create Mapping

| | |
|---|---|
| **Function Signature** | + createMapping( m : Mapping) : Boolean |
| **Description** | Create a new mapping between two Concepts (CodeSystemConcepts and epSORef-SystemConcepts) |
| **Inputs** | Mapping instance |
| **Outputs** | An acknowledgment indicating whether the mapping was created or not.<br>The generated MappingId, if not supplied. |
| **Precondition** | |
| **Postconditions** | The mapping is available for use via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Mapping already exists.<br>Invalid mapping properties |
| **Comments** | |

### 3.3.12 Maintain Mapping

| | |
|---|---|
| **Function Signature** | + updateMapping ( i : MappingId, m : Mapping) : Boolean |
| **Description** | Update mapping properties. |
| **Inputs** | MappingId of the mapping that is being updated<br>Mapping instace (with updated properties) |
| **Outputs** | An acknowledgment indicating whether the mapping was updated or not. |
| **Precondition** | |
| **Postconditions** | The updated mapping is available for use via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Invalid mapping properties |
| **Comments** | |

## 3.4 Value Set Authoring/Curation

### 3.4.1 Create Value Set

| | |
|---|---|
| **Function Signature** | + createValueSet( v : ValueSet, [ ver : ValueSetVersion, [ c : CodeSystemConcept[], [ m : Mappings[]]]) : Boolean, [ValueSetId] |
| **Description** | Create a Value Set (extensional) |
| **Inputs** | Value Set instance<br>Value Set Version instance (optional)<br>Code System Concept list (optional)<br>Mappings (optional) |
| **Outputs** | An acknowledgment indicating whether the value set was created or not.<br>Generated incremental Value Set Id, if not input |
| **Precondition** | A sequencer will provide the value set incremental id, if not provided in the input. |
| **Postconditions** | 1. The value set is available for access via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Value Set already exists.<br>Value Set Version already exists<br>Invalid Value Set properties<br>Invalid Value Set Version properties |
| **Comments** | Intensional ValueSets, defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time are not supported by the epSOS Terminology Services. |

840

### 3.4.2 Maintain Value Set

| | |
|---|---|
| **Function Signature** | + updateValueSet( i : ValueSetId, v : ValueSet, [ ver : ValueSetVersion, [ c : Code-SystemConcept[], [ m : Mappings[]]]) : Boolean |
| **Description** | Update properties or expression of a value set definition (extensional) |
| **Inputs** | Value Set Id to be updated<br>Value Set instance (with updated properties)<br>Value Set Version instance (with updated properties) (optional)<br>Code System Concept list (optional)<br>Mappings (optional) |
| **Outputs** | An acknowledgment indicating whether the value set was updated or not. |
| **Precondition** | |
| **Postconditions** | The updated value set is available for access via the epSOS Terminology Access Services functions.<br>A new value set version is created. |

| | |
|---|---|
| **Exceptional behaviour** | Value Set does not exists.<br>Concept not found.<br>Invalid Value Set properties<br>Invalid Value Set Version properties |
| **Comments** | Intensional ValueSets, defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time are not supported by the epSOS Terminology Access Services. |

845

### 3.4.3 Update Value Set Version Status

| | |
|---|---|
| **Function Signature** | + setValueSetVersionStatus ( i : ValueSetId, s : Status) : Boolean |
| **Description** | Changes the status of a value set version (suspend, reinstate, cancel, remove). |
| **Inputs** | Value Set Version Id<br>Status |
| **Outputs** | An acknowledgment indicating whether the status revision request was successfully processed. |
| **Precondition** | |
| **Postconditions** | Value Set Version status is updated appropriately. |
| **Exceptional behaviour** | Value Set Version does not exist.<br>Invalid state change |
| **Comments** | |

## 3.5 Concept Domain and Usage Context Authoring/Curation

### 3.5.1 Create Concept Domain

| | |
|---|---|
| **Function Signature** | + createConceptDomain( d : ConceptDomain) : Boolean, [ConceptDomainId] |
| **Description** | Create a Concept Domain. |
| **Inputs** | Concept Domain instance |
| **Outputs** | An acknowledgment indicating whether the concept domain was created or not.<br>Generated incremental Concept Domain Id, if not input |
| **Precondition** | A sequencer will provide the concept domain id, if not provided in the input. |
| **Postconditions** | The concept domain is available for access via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Concept Domain already exists.<br>Invalid Concept Domain properties |
| **Comments** | |

### 3.5.2 Maintain Concept Domain

| | |
|---|---|
| **Function Signature** | + updateConceptDomain( i : ConceptDomainId, d : ConceptDomain, [ c : UsageContext[], [ v : ValueSet[] ] ]) : Boolean |
| **Description** | Update properties of a Concept Domain, including bindings to value sets within usage contexts |
| **Inputs** | Concept Domain Id to be updated<br>Concept Domain instance (with updated properties)<br>UsageContext list (with updated properties) (optional)<br>ValueSet list (with updated properties) (optional) |
| **Outputs** | An acknowledgment indicating whether the concept domain was updated or not. |
| **Precondition** | |
| **Postconditions** | The updated concept domain is available for access via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Concept Domain does not exists.<br>Invalid Concept Domain properties<br>Invalid state transition<br>Unrecognized/invalid value set<br>Unrecognized/invalid usage context |
| **Comments** | A binding of a value set to a concept domain or usage context is a function relating a Value Set to ConceptDomains and/or Usage Contexts. |

### 3.5.3 Create Usage Context

| | |
|---|---|
| **Function Signature** | + createUsageContext( c : UsageContext) : Boolean, [UsageContextId] |
| **Description** | Create a Usage Context. |
| **Inputs** | Usage Context instance |
| **Outputs** | 1. An acknowledgment indicating whether the usage context was created or not.<br>2. Generated incremental Usage Context Id, if not input |
| **Precondition** | 1. A sequencer will provide the usage context id, if not provided in the input. |
| **Postconditions** | 1. The usage context is available for access via the epSOS Semantic Service functions. |
| **Exceptional behaviour** | Usage Context already exists.<br>Invalid Usage Context properties |
| **Comments** | |

860

### 3.5.4 Maintain Usage Context

| | |
|---|---|
| **Function Signature** | + updateUsageContext( i : UsageContextId, c : UsageContext) : Boolean |
| **Description** | Update properties of a Usage Context |
| **Inputs** | Usage Context Id of the Usage Context to be updated<br>Usage Context instance (with updated properties) |
| **Outputs** | An acknowledgment indicating whether the usage context was updated or not. |
| **Precondition** | |
| **Postconditions** | The updated usage context is available for access via the Semantic Service functions. |
| **Exceptional behaviour** | Usage Context does not exists.<br>Invalid Usage Context properties<br>Invalid State transition |
| **Comments** | |

865

## 3.6 Query / Browsing API (IQuery Interface)

**Synopsis:**


**Filter Criteria** are intended to specify the types of filtering mechanisms specific to the Code System Entities being queried. It is understood that the identification attributes for code system elements and epSOS reference system elements, such as concepts and value sets, are different, and, as such, different filter criteria may be applied to appropriately identify and filter the code system entity information being queried. As part of the technical specification, Filter Criteria will be specified explicitly in accordance with the developed Platform Independent Model.

**Query Control** - Query Control defines a placeholder for parameters that have the capacity of constraining the query results returned from a query retrieving code system or epSOS reference system elements. This allows a user to restrict the amount of data returned or sort the data. This is different from Filter Criteria, which specifies restrictions on the business data retrieved.

### 3.6.1 List Code Systems / Return Code System Details

| **Function Signature** | + getCodeSystem() : CodeSystem[] |
|---|---|
| **Description** | List the code systems available in the epSOS reference system matching the entered filter criteria. Filter criteria may be related to all code system attributes. As a result list of composite types CodeSystem is provided. From the composite type for a given code system all details (metadata) can be retrieved. |
| **Inputs** | Filter Criteria (related to e.g. name, description, administrativeInfo)<br>Query Control |
| **Outputs** | A set of zero or more code systems, together with metadata properties, available on the instance of the epSOS Semantic Service. |
| **Preconditions** | epSOS Terminology Access Services installed and running |

| | |
|---|---|
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to the technical specification. Should include a name for a simple means of retrieving ID where not known. |

885

### 3.6.2 List Code System Versions / Return Code System Version Details

| | |
|---|---|
| **Function Signature** | + getCodeSystemVersion(cs : CodeSystem) : CodeSystemVersion[] |
| **Description** | For a given code system, list all the code system versions available in the epSOS reference system matching the entered filter criteria. Filter criteria may be related to all code system version attributes.<br>As a result a list of composite types CodeSystemVersion is provided. From the composite type for a given code system version all details (metadata) can be retrieved. |
| **Inputs** | Code System<br>Filter Criteria (related to e.g. name, effectiveDate, status, language etc.)<br>Query Control |
| **Outputs** | A set of zero or more code system versions, together with metadata properties, available on the instance of the epSOS Semantic Service. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | Code System not found<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known.<br>For the versions of code system, an acceptable input should also be "current" or "latest". |

890 **3.6.3 List Code System Concepts / Return Code System Concept Details**

| | |
|---|---|
| **Function Signature** | + getCodeSystemConcept(csv: CodeSystemVersion) : Concept[] |
| **Description** | Returns the set of all concepts in the specified code system version, optionally filtered by input criteria, such as status or specific concept property, e.g. values. Filter criteria may be related to all code system concept attributes.<br>As a result list of composite types CodeSystemConcept is provided. From the composite type for a given concept all details (metadata) can be retrieved. |
| **Inputs** | Code System<br>Code System Version<br>Filter Criteria |

| | |
|---|---|
| | Query Control |
| **Outputs** | A set of zero or more concepts in the specified code system version matching the filter criteria. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | Code System not found.<br>Code System Version not found.<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known.<br>Returning all concepts in a code system version is generally impractical for large code sets. Indexing, query optimization is necessary.<br>For the versions of code system, an acceptable input should be "current" or "latest". |

### 3.6.4 List epSOS Reference System Versions / Return epSOS Reference System Version Details

| **Function Signature** | + getEpSOSRefSystemVersion() : EpSOSRefSystemVersion[] |
|---|---|
| **Description** | Lists all versions of the epSOS Reference System available in the epSOS reference system matching the entered filter criteria. Filter criteria may be related to all epSOS reference system version attributes.<br>As a result list of composite types EpSOSRefSystemVersion is provided. From the composite type for a given version all details (metadata) can be retrieved. |
| **Inputs** | Filter Criteria (related to e.g. name, effectiveDate, ststus, language etc.)<br>Query Control |
| **Outputs** | A set of zero or more epSOS reference system versions, together with metadata properties available on the instance of the epSOS Semantic Service. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known.<br>For the versions of code system, an acceptable input should be "current" or "latest". |

### 3.6.5 List epSOS Reference System Concepts / Return epSOS Reference System Concept Details

| | |
|---|---|
| **Function Signature** | + getEpSOSRefSystemConcept(esv: EpSOSRefSystemVersion) : Concept[] |
| **Description** | Returns the set of all concepts in the specified epSOS reference system version, optionally filtered by input criteria, such as status or specific concept property, e.g. values. Filter criteria may be related to all epSOS reference system concept attributes.<br>As a result list of composite types epSOSRefSystemConcept is provided. From the composite type for a given concept all details (metadata) can be retrieved. |
| **Inputs** | epSOS Reference System Version<br>Filter Criteria<br>Query Control |
| **Outputs** | The set of zero or more concepts in the specified epSOS reference system version matching the filter criteria. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | epSOS Reference System Version not found.<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to the technical specification. Should include name for a simple means of retrieving ID where not known.<br>Returning all concepts in a epSOS reference system version is generally impractical for large code sets. Indexing, query optimization is necessary.<br>For the versions of epSOS reference system, an acceptable input should be "current" or "latest". |

### 3.6.6 List Value Sets / Return Value Set Details

| | |
|---|---|
| **Function Signature** | + getValueSet() : ValueSet[] |
| **Description** | List the value sets available in the epSOS reference system, matching entered filter criteria. Filter criteria may be related to all value set attributes.<br>As a result a list of composite types ValueSet is provided. From the composite type for a given value set all details (metadata) can be retrieved. |
| **Inputs** | Filter Criteria (related to e.g. name, description, status)<br>Query Control |
| **Outputs** | A list of zero or more value sets, together with metadata properties available on the instance of the epSOS Semantic Service. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known. |

### 3.6.7 List Value Set Versions / Return Value Set Version Details

| | |
|---|---|
| **Function Signature** | + getValueSetVersion(vs : ValueSet) : ValueSetVersion[] |
| **Description** | For a given value set, list all the value set versions available in the epSOS reference system matching entered filter criteria. Filter criteria may be related to all value set version attributes.<br>As a result list of composite types ValueSetVersion is provided. From the composite type for a given value set version all details (metadata) can be retrieved. |
| **Inputs** | Value Set<br>Filter Criteria (related to e.g. name, effectiveDate, ststus, language etc.)<br>Query Control |
| **Outputs** | A list of zero or more code system versions, together with metadata properties, available on the instance of the epSOS Semantic Service. |
| **Preconditions** | epSOS Terminology Access Services installed and running |
| **Postconditions** | |
| **Exceptional behaviour** | Value set not found<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known.<br>For the versions of code system, an acceptable input should be "current" or "latest". |

910

### 3.6.8 List Value Set Concepts / Return Value Set Concept Details

| | |
|---|---|
| **Function Signature** | + getValueSetConcept(vsv: ValueSetVersion) : Concept[] |
| **Description** | Returns the set of all concepts in the specified value set version, optionally filtered by input criteria, such as status or specific concept property, e.g. values. Filter criteria may be related to all value set version and concept attributes.<br>As a result a list of composite types Concept (generalization of CodeSystemConcept and EpSOSRefSystemConcept) is provided. From the composite type for a given concept, all details (metadata) can be retrieved. |
| **Inputs** | Value Set Version<br>Filter Criteria<br>Query Control |
| **Outputs** | The set of zero or more concepts in the specified value set version matching the filter criteria. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | ValueSetVersion not found.<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a |

| | |
|---|---|
| | simple means of retrieving ID where not known.<br>Returning all concepts in a value set version is generally impractical for large code sets. Indexing, query optimization is necessary.<br>For the versions of value set, an acceptable input should be "current" or "latest". |

### 3.6.9 Check Concept Membership in Value Set

| **Function Signature** | + checkConceptValueSetMembershipt(c : Concept, vsv : ValueSetVersion) : Boolean |
|---|---|
| **Description** | Determine whether the supplied coded concept exists in the supplied value set.<br>As an input parameter composite types Concept (generalization of CodeSystemConcept and EpSOSRefSystemConcept) is provided. From the composite type for a given concept all details (metadata) can be retrieved. |
| **Inputs** | Concept<br>Value Set Version |
| **Outputs** | Return True if concept exists in value set, otherwise return False |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | ValueSetVersion not found.<br>Concept not found. |
| **Comments** | Definition of filter criteria is left to technical specification. |

### 3.6.10 List Usage Contexts / Return Usage Context Details

| **Function Signature** | + getUsageContext() : UsageContext[] |
|---|---|
| **Description** | Lists the usage contexts that are available in the epSOS reference system matching entered filter criteria. Filter criteria may be related to all usage context attributes.<br>As a result list of composite types UsageContext is provided. From the composite type for a given usage context all details (metadata) can be retrieved. |
| **Inputs** | Filter Criteria (related to e.g. name, description, status)<br>Query Control |
| **Outputs** | Listing of zero or more usage contexts on this instance of the epSOS Semantic Service that match the input filter criteria, together with associated metadata. |
| **Preconditions** | |

| | |
|---|---|
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to the technical specification.<br>When search attributes are applied, the result set is restricted to the usage contexts that match the search filter criteria. |

### 3.6.11 List Usage Context Concepts / Return Usage Context Concept Details

| | |
|---|---|
| **Function Signature** | + getUsageContextConcept(uc : UsageContext) : Concept[] |
| **Description** | Returns the set of all concepts in the specified usage context (independently on coding system or value set), optionally filtered by input criteria, such as status or specific concept property, e.g. values. Filter criteria may be related to all usage contexts and concept attributes.<br>As a result list of composite types Concept (generalization of CodeSystemConcept and EpSOSRefSystemConcept) is provided. From the composite type for a given concept all details (metadata) can be retrieved. |
| **Inputs** | Usage Context<br>Filter Criteria<br>Query Control |
| **Outputs** | The set of zero or more concepts in the specified usage context, matching the filter criteria. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Usage Context not found.<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification.<br>Returning all concepts in a usage context is generally impractical for large code sets. Indexing, query optimization is necessary. |

925

### 3.6.12 List Usage Context Value Sets / Return Usage Context Value Set Details

| | |
|---|---|
| **Function Signature** | + getUsageContextValueSet(uc : UsageContext) : ValueSet[] |
| **Description** | Returns the list of all value sets in the specified usage context (independently on coding system), optionally filtered by input criteria, such as status or specific value set property, e.g. name. Filter criteria may be related to all usage contexts and value set attributes.<br>As a result list of composite types ValueSet is provided. From the composite type for a given value set all details (metadata) can be retrieved. |

| | |
|---|---|
| **Inputs** | Usage Context<br>Filter Criteria<br>Query Control |
| **Outputs** | The set of zero or more concepts in the specified usage context matching the filter criteria. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Usage Context not found.<br>Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. |

930

### 3.6.13 List Associations / Return Association Details

| | |
|---|---|
| **Function Signature** | + getAssociation() : Association[] |
| **Description** | List the Association available in epSOS reference system that match a set of input filter criteria. |
| **Inputs** | Filter Criteria<br>Query Control |
| **Outputs** | A listing of zero or more associations that match the input filter criteria |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria. This is expected to include: Code System Version, epSOS Reference System Version, Concepts, Association Types, whether only direct associations are considered or whether the transitive closure of the relation are used.<br>There are potentially many different types of Associations that can be defined. In some cases, the associationss are explicitly defined, persisted and managed, in other cases, queries can be made based on algorithms.<br>This operation will allow retrieval of any type of Association, including lexical and rules based |

935 **3.6.14 List Association Types / Return Association Type Details**

| | |
|---|---|
| **Function Signature** | + getAssociationType() : AssociationType[] |
| **Description** | List association types that are available on a epSOS Semantic Service that match entered filter criteria (e.g. epSOS reference system version, association specific concepts). |

| | |
|---|---|
| **Inputs** | Filter Criteria<br>Query Control |
| **Outputs** | Returns a list of zero or more association types, together with associated metadata |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | |

### 3.6.15 Determine Transitive Concept Association

940

| | |
|---|---|
| **Function Signature** | + getTransitiveConceptAssociation(p : CodeSytemConcept, c: CodeSystemConcept, t : epSOSRefSystemAssociationType) : Boolean, [Association[] ] |
| **Description** | Determine whether there exists a transitive relationship between two concepts; if it exists, provide the association path.<br>Determine Transitive Concept Association determines if it is possible to navigate from a given Parent Concept to a given Child Concept in one or more association transitions. |
| **Inputs** | Parent Concept (EpSOSRefSystemConcept type)<br>Child Concept (EpSOSRefSystemConcept type)<br>epSOSRefSystemAssociationType |
| **Outputs** | Determine Transitive Concept Association will return the edge graph necessary to traverse from Parent Concept to Child Concept using the given Association Type, or null if the transitive closure does not exist. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid Parent Concept<br>Invalid Child Concept<br>Invalid Association Type |
| **Comments** | |

### 3.6.16 Compute Subsumption Association

| | |
|---|---|
| **Function Signature** | + getSubsumptionAssociation(p : CodeSyytemConcept, c: CodeSystemConcept, t : epSOSRefAssociationType) : Boolean, [ Mapping[] ] |
| **Description** | Subsumes tests whether the parent concept subsumes (is implied by) the child. Subsumes returns true if and only if a Child Concept can be determined to belong to the transitive closure of the hasSubtype association graph headed by Parent Concept. If the service supports subsumption involving qualifiers (compositional expressions), it |

| | |
|---|---|
| | must define the appropriate translation semantics. If the service doesn't support subsumption on concepts defined using compositional expression, it should raise the QualifiersNotSupported exception if presented with a parent concept or child concept containing qualifiers. |
| **Inputs** | Parent Concept (EpSOSRefSystemConcept type)<br>Child Concept (EpSOSRefSystemConcept type) |
| **Outputs** | Subsumes will return true if the child code can be determined to be a subtype of the parent. Subsumes will also return true if the child and parent codes are equivalent. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid Parent Concept<br>Invalid Child Concept |
| **Comments** | |

945

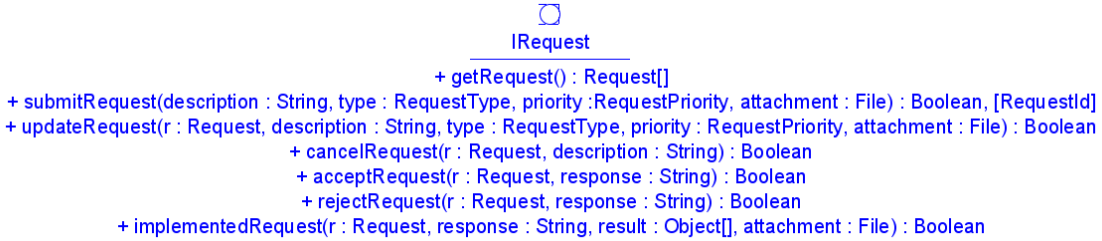
### 3.6.17 List Concept Mappings / Return Concept Mappings Details

| | |
|---|---|
| **Function Signature** | + getMapping() : Mapping[] |
| **Description** | List the Mappings available in epSOS reference system that match a set of input filter criteria. |
| **Inputs** | Filter Criteria (including concepts from code system or epSOS reference system)<br>Query Control |
| **Outputs** | A list of zero or more mappings that match the input filter criteria |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria. This is expected to include: Code System Version, epSOS Reference System Version, Code System Concepts, epSOS reference system version. |

950

## 3.7 Request processing API (IRequest Interface)

**Synopsis:**




### 3.7.1 List Requests / Return Request Details

955

| **Function Signature** | + getRequest() : Request[] |
|---|---|
| **Description** | List the requests available in the terminology system matching entered filter criteria. Filter criteria may be related to all request attributes.<br>As a result list of composite types Request is provided. From the composite type for each given request all details (metadata) can be retrieved. |
| **Inputs** | Filter Criteria (related to e.g. name, description, author, status, type, priority, effeciveDate)<br>Query Control |
| **Outputs** | A listing of zero or more request, together with metadata properties available on the instance of the terminology service. |
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Invalid filter criteria<br>Invalid query control |
| **Comments** | Definition of filter criteria is left to technical specification. Should include name for a simple means of retrieving ID where not known. |

### 3.7.2 Submit New Request

| **Function Signature** | + submitRequest(description : String, type : RequestType, priority : RequestPriority, attachment : File) : Request |
|---|---|
| **Description** | Set all input parameters describing the request and submit it as a new request.<br>As a result composite type Request is provided. |
| **Inputs** | Description (textual description of requirements for a new request)<br>Type<br>Priority<br>Attachment (any supported file format) |
| **Outputs** | Composite type Request containing new request object |

| | |
|---|---|
| **Preconditions** | |
| **Postconditions** | |
| **Exceptional behaviour** | Description not specified.<br>Incorrect request type |
| **Comments** | New request will be created with status Created. Author, Creation Date and Status date will be set according to internal system parameters. |

960

### 3.7.3 Update Created Request

| | |
|---|---|
| **Function Signature** | + updateRequest(r : Request, description : String, type : RequestType, priority : RequestPriority, attachment : File) : Boolean |
| **Description** | Modify any of public parameters of already created request (with status Created). Request can be modified only by the same user as submitted it (Author).<br>Return value indicates whether request was successfully modified. |
| **Inputs** | Request<br>Description (textual description of requirements for a new request)<br>Type<br>Priority<br>Attachment (any supported file format) |
| **Outputs** | True, if the request was modified according to input parameters, otherwise False. |
| **Preconditions** | Update shall be done only by the request Author. |
| **Postconditions** | |
| **Exceptional behaviour** | Request not found. |
| **Comments** | |

965

### 3.7.4 Cancel Request

| | |
|---|---|
| **Function Signature** | + cancelRequest(r : Request, description : String) : Boolean |
| **Description** | Change status of the request to Cancelled.<br>Requests in status Cancelled are no longer valid.<br>Textual description of the cancellation reason shall be provided.<br>Request can be cancelled only by the same user that submitted it (Author).<br>Return value indicates whether request was successfully modified. |
| **Inputs** | Request |
| **Outputs** | True, if status of the request was modified to Cancelled, otherwise False |
| **Preconditions** | Cancellation shall be done only by the request Author. |

| | |
|---|---|
| **Postconditions** | |
| **Exceptional behaviour** | Request not found. |
| **Comments** | |

### 3.7.5 Accept Request

| | |
|---|---|
| **Function Signature** | + acceptRequest(r : Request, response : String) : Boolean |
| **Description** | Change status of the request to Accepted.<br>Acceptation of the request means that proposed changes were approved and will be adopted in the future.<br>Response provides textual description of future results of adoption.<br>Request can be accepted only by the Terminology Administration user.<br>Return value indicates whether request was successfully modified. |
| **Inputs** | Request<br>Response (textual description of the way how request will be adopted, etc. what concepts will be modified or created) |
| **Outputs** | True, if status of the request was modified to Accepted, False otherwise |
| **Preconditions** | Acception shall only be done by an epSOS user with administrative rights. |
| **Postconditions** | |
| **Exceptional behaviour** | Request not found. |
| **Comments** | |

### 3.7.6 Reject Request

| | |
|---|---|
| **Function Signature** | + rejectRequest(r : Request, response : String) : Boolean |
| **Description** | Change status of the request to Rejected.<br>Rejection of the request means that proposed changes will not be adopted. Response provides a textual description of reasons for rejection. Request can be rejected only by the Terminology Administration user.<br>Return value indicates whether request was successfully modified. |
| **Inputs** | Request<br>Response (textual description of reasons for rejection) |
| **Outputs** | True, if status of the request was modified to Rejected, False otherwise |
| **Preconditions** | Rejection can only be done by an epSOS user with administrative rights. |
| **Postconditions** | |

| | |
|---|---|
| **Exceptional behaviour** | Request not found. |
| **Comments** | |

### 3.7.7 Implemented Request

| | |
|---|---|
| **Function Signature** | + implementedRequest(r : Request, response : String, result : Object[], attachment : File) : Boolean |
| **Description** | Change status of the request to Implemented.<br>By implementation of the request the author is informed that the requested changes were already done.<br>Response provides textual description of implemented results.<br>Request can be flagged to Implemented only by an epSOS Semantic Service user with administrative rights.<br>Return value indicates whether request flag was successfully modified. |
| **Inputs** | Request<br>Response (textual description of adoption results)<br>Results (list of objects changed according to request's requirements) |
| **Outputs** | True, if status of the request was modified to Implemented, False otherwise. |
| **Preconditions** | Implented flag can only be toggled by an epSOS user with administrative rights. |
| **Postconditions** | |
| **Exceptional behaviour** | Request not found. |
| **Comments** | |

980