INTERNATIONAL HEALTH TERMINOLOGY
STANDARDS DEVELOPMENT ORGANISATION

SNOMED Clinical Terms Technical Specification

SNOMED CT API Outline Specification

DISCUSSION DRAFT

Date 20080220
Version 2.01

Document Properties

Filename:	IHTSDO_APISPEC_DRAFT-20080220_v2-01.doc
Title:	SNOMED Clinical Terms Technical Specification
Creating Author:	David Markwell
Subject*:	IHTSDO, API Specification, Draft

* Subject should be filled in as 3 keywords. The first keyword should be a structural or organizational entity, e.g. “IHTSDO”. The second keyword should be the process the document is related to, e.g. a “Meeting”. The third keyword should be an object, e.g. an “Agenda”.

Amendment History

Version	Date	Editor	Comments
2.00	20030117	David Markwell	Internal use only
2.01	20080220	Kent Spackman	Conversion to IHTSDO standard format, Removal of “internal only” restriction Addition of glossary

Purpose of this document

This specification builds on previously identified requirements for SNOMED Terminology Services (see the Technical Implementation Guide). It provides an outline specification for an Application Programming Interface to these services. The intention of this document is to inform and assist the development, use and evaluation of such services by third parties. The goal of this specification is not to constrain alternative approaches but rather to encourage the development and widespread use of services that enable consistent and effective use of SNOMED CT.

Status

This is a draft originally created for consideration by the SNOMED International Editorial Board. In this draft the section on proposed user-interface services is incomplete.
The author of this paper is David Markwell as technical consultant to SNOMED International.

Table of Contents

1 Introduction

1.1 Background

During the initial development of SNOMED CT the Design Team considered the idea of a SNOMED CT Application Programming Interface (API) in general terms. This consideration built on the pre-existing Clinical Terms Version 3 API specification produced by the NHS.

The SNOMED International Editorial Board indicated a desire to consider adoption of a more general standard for a terminology API. As a result, a review of the suitability of the proposed HL7 Central Terminology Services (CTS) was undertaken. The July 2002 version of the HL7 CTS was reviewed in a white paper discussed by the SNOMED International Editorial Board at its September 2002 meeting. That white paper made the following recommendation:

Based on the material reviewed the author recommends the SNOMED International Editorial Board to develop a SNOMED CT API. The result of this development should be a technically detailed description of the services outlined in this paper (and any other services deemed important to SNOMED CT implementation). At this stage the specification should **not** assume a particular technical solution.

The specification of the proposed SNOMED CT API should explicitly state its relationship to the HL7 CTS at three levels:

- At the scope level noting the differences and overlaps in scope and objectives;
- At the HL7 CTS property level to ensure that appropriate codified names are available to access SNOMED CT attributes in an unambiguous manner.
- At the individual service level:
  - Describing SNOMED CT services in terms of the set of HL7 CTS services required to deliver a particular SNOMED CT service
  - Indicating how to deliver each HL7 CTS service using an appropriate SNOMED CT service.

The recommendation was accepted by the SNOMED International Editorial Board and this document represents the next step.

1.2 Rationale for a SNOMED CT API specification

SNOMED CT is a powerful terminological resource which offers a wide range of potential benefits to patients, health professionals and health service providers. However, to deliver these benefits SNOMED CT must be made available to users in ways that facilitates its use in a busy clinical environment. This requires application software that is designed to utilize SNOMED CT in ways that simplify rather than complicate the process of recording and subsequently retrieving clinical information. The Technical Implementation Guide outlines the services required and some advice on optimization of these services. This document extends this outline to identify the framework of an agreed, publicly available common specification of essential and desirable API requirements for implementation of SNOMED CT.

1.3 Objectives

This outline API specification should encourage the development of effective SCT Enabled Applications in a competitive environment by:

- Encouraging developers of terminology server software to provide appropriate accessible functionality to enable consistent and effective use of the full range of features provided by SNOMED CT.
- Enabling developers of healthcare applications to exercise an informed choice between in-house development of terminology services and use of third-party products.

This outline API specification should minimize the overall effort of development of SCT Enabled Applications and to assist developers to avoid known pitfalls by:

- Adding technical clarity and precision to the description of services in the Technical Implementation Guide.
- Identifying key areas for evaluation and comparison of terminology servers in terms of the completeness and effectiveness with which they enable access to SNOMED CT.

Finally the outline API specification is also intended to inform the SNOMED International Editorial Board decisions about the future development of SNOMED CT by:

- Providing a vendor-neutral point of reference for determining the likely functional and developmental consequences of any proposed changes to the SNOMED CT design.

1.4 Scope of the SNOMED CT API

The Technical Implementation Guide specifies three main groups of services required by a SCT Enabled Application. These are:

- Terminology services [07_A-6]
  - To access SNOMED CT terminological resources.
- Record services [07_A-7]
  - To enter, store, retrieve or communicate SNOMED CT encoded information.
- Mapping services [07_A-8]
  - To map information recorded using SNOMED CT to administrative and epidemiological classifications.

Terminology and mapping services are generalizable to all SCT Enabled Applications and are thus within the scope of this API specification. In contrast, record services are highly dependent on the nature of a particular application and are outside the scope of this API specification. However, this specification is directly relevant to those developing record service applications as these applications are clients of terminology and mapping services.

1.5 Terminology server architecture

The Technical Implementation Guide divides terminology services into two general types:

- Reference Services
  - Services that do not include a user-interface.
  - The client application may use reference services to undertake many different functions.
  - For some of these functions the client application will populate an appropriate user interface component
- User Interface (UI) Services
  - Services that include one or more user-interface components that can be used in and programmatically accessed by the client application.
  - Another possible type of UI service is a SNOMED CT browser with an API for returning selected data to a client application.

In this specification these services are further subdivided to separately identify services in each of the above categories that only required for cross-mapping. The rationale for this is that the requirements for cross-mapping services are more variable than those for other types of service.

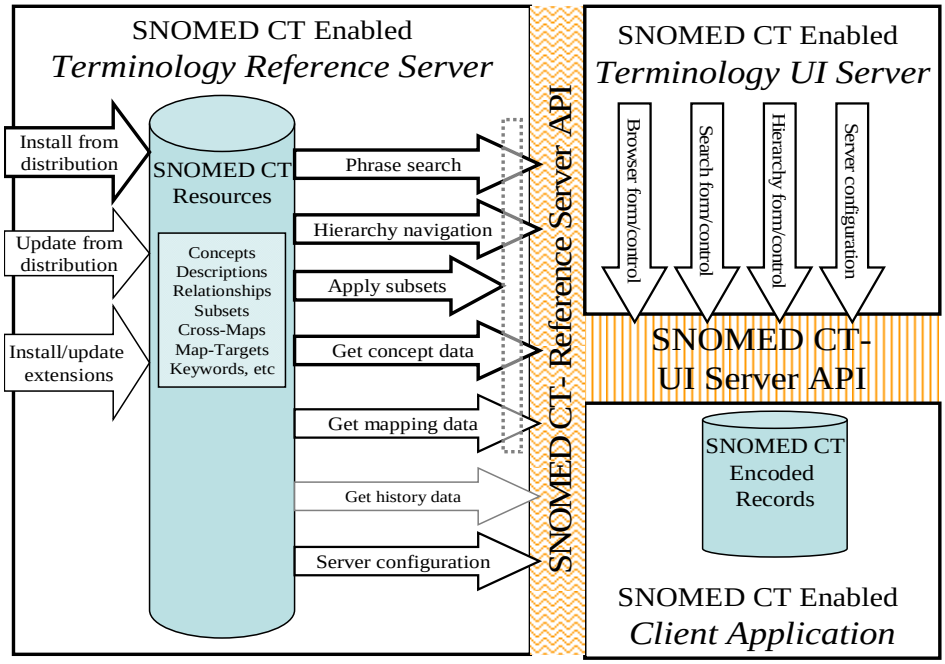

Figure 1. Client application with Terminology Reference and UI servers

1.6 Programming interface technology

An API may be expressed in various ways. The general principals may be delivered using various different types of interface mediated by various different technologies. Potential technologies include:

- Active-X / COM / DCOM in Microsoft Window environments
- Java / IDL
- CORBA
- Various web-based technologies using XML, SOAP, etc.

Decisions about which technologies to support depend on the intended functionality, performance, accessibility, ease of use and support requirements for maintenance or updates.
This document expresses the API specification in a generic manner without specifying a particular system architecture or programming environment. Therefore, the specification is not directly testable in a given implementation environment. Instead it identifies the key issues relevant to implementation in a variety of environments in a manner that can be readily applied to or compared against specific implementations.

2 Overall API design issues

2.1 Representation of API services and exchanged data

2.1.1 API service definitions

At its simplest level an API service returns some data and/or undertakes an action. The action undertaken may be an internal change in the state of the server or some external manifestation communication with other software or hardware (e.g. modification of a screen display). Completion of an API service may also require provision of some data.

The service definition identify:

- Data required by a service
- Actions the server is required to take when a service is requested
- Data to be returned by a service

2.1.2 Data types, structures and collections

The nature of the data required or returned may be simple (e.g. a value expressed in a particular data type) or complex (e.g. a set of values in a compound data structure or the data content of a programmable object).

This specification identifies data contents and general types of data that must be submitted and returned to one or more services. These are considered under three headings each of which is associated with different general functional requirements.

- Data item
  - A single named value that can be expressed using a simple data type in most programming environments.
- Data structure
  - A predefined set of logically related named elements.
  - Each named element is predefined as a "data item", another "data structure" or a "data collection".
  - A server must allow the values of individual named elements to be read by the client.
  - A server must allow the values of some specified named elements to be changed by the client but must not allow read-only values to be amended.
- Data collection
  - An ordered set containing an arbitrary number of members.
  - All the members of a collection are of the same predefined member type.
  - The member type for a collection is a named "data item" or "data structure".
  - A server must be able to:
    - Return a count of members in a data collection;
    - Iterate through a members of a collection;
    - Return individual members and/or sub-collections matching appropriate keys.

This document does not require particular technical representations of data structures or data types. The server developer should use an appropriate representation available within their technical implementation environment for each of the general data types or data structures specified.

2.1.3 Submitting data required by a service

The specification describes the complete sets of data required for full support of each service. This includes some mandatory data that is required whenever a service is requested and some optional data that is required only to enable particular variants of a service.

There are various ways in which data could be made available to a service. These include:

- Explicitly passing the data or a reference to the data (e.g. as parameters to a server call).
- By setting a property or variable accessible to the server prior to one or more service requests which require that data.
- As part of a stream of data associated with an instruction.

The specification does not require the data to be made available to the server in a particular way as this is technology dependent. However, some advice is provided in respect of the differences in handling configuration data which may be more or less similar across a range of service request and data which is likely to be different in respect of each service.

2.1.4 Returning data from a service

The specification describes the complete sets of data to be returned by each service.

There are various ways in which this data may be returned. These include:

- Returning data as from a service expressed as a function call.
- Modification of data in parameters passed with a server call.
- By setting a property or variable accessible while the service is being performed or after it has completed.
- As part of a stream of data returned from a server.
- As a parameter passed by the server when raising an event or calling a call-back procedure.

The specification does not require the data to be returned by the server in a particular way as this is technology dependent.

2.1.5 Data structures, references, identifiers and raw data

Where the data required or returned by a service is expressed as a "data structure" the data may be passed between client and server in different ways. These include:

- As the raw data expressed with a syntax that allows it to be interpreted.
- As an object supported by the technical environment of the API (e.g. an instance of a COM, Java or CORBA class).
- As an internal pointer that can be passed to other API services to provide access to elements within the structure.
- In the case of a Concept, Description, Relationship or Subset, as the unique SCTID for that Component. This assumes that the API offers other services that return individual properties for an identified Component.
- As an SQL query string that returns a specified record or record set.

This specification does not require a server to use a particular approach. The chosen approach should optimize performance by balancing the impact of repeated API requests (e.g. to separately fetch each

property) with the impact of passing larger volumes of raw data across the API in response to a single request. The balance between these factors is technology dependent.

2.1.6 Asynchronous services, events and polling

Most of the terminology services described in this document need to be completed within a matter of a few milliseconds. Synchronous processing - where the calling process waits for completion of the service - is likely to meet many of these requirements.

However, there are some situations when asynchronous handling of services is essential and others in which it is useful. These include:

- Services that relate to interactions with the user interface.
  - After displaying data the server needs to allow the client application to proceed while waiting for user interaction with the displayed data.
- Services that are liable to take a prolonged or indeterminate period to complete
  - A search for Descriptions matching a common word or all the descendant Concepts of a top-level Concept may take a few seconds to complete. If the client application is forced to wait for completion it is likely to present an unacceptable delay to the user.
- Services that may return large collections of data
  - A search or hierarchy descent that returns hundreds of matching Descriptions or Concepts may generate a quantity of data that presents difficulties for a client application if returned in a single chunk.

The description of some services notes a requirement or recommendation for some type of asynchronous processing. Other services may also benefit from an asynchronous approach.

There are several ways of meeting the requirement for asynchronicity:

- In an environment that supports the raising and handling of programmable events these can be used:
  - to return data as found;
  - to intermittently report progress event when no data is ready to be returned;
  - to allow the client to cancel a process by modifying an event parameter.
- In an environment that does not support events but does allow parallel processing by client and server software a polling approach may be used:
  - the client regularly tests data returned by the server.
  - the server can test a value set by the client to allow the client to terminate the service.
- If neither of these mechanisms is applicable asynchronicity can be partially mimicked
  - by specifying pairs of services that return "first" and "next" items rather than a complete collection.
  - by including timeout and continuation parameters.

An event driven approach has many advantages:

- It places less onus on the client to specifically manage timing and processing of incomplete collections,
- It avoids the need for the server to build and pass to the client arbitrarily large internal collections.

This document does not mandate a particular approach to requirements for asynchronous services. The range of possibilities is constrained by the technical environment.

2.1.7 Actions

Many API services are only concerned with the exchange of data between the client and server. However, other actions are required by some services. These include:

- Display of data.
- Import of data from text files.
  - For example, to import Subsets and or new distribution files from SNOMED CT or from a provider of an extension.
- Export of data to files
  - For example, to share Subsets with other users within an organization.

The specification describes display related actions in general terms but does not dictate a particular appearance for data display. Import and export actions assume the use of files in the formats specified by the SNOMED CT Technical Reference Guide.

3 Reference Services

3.1 Access server and terminology system properties

3.1.1 GetServerInfo

Service Name	**GetServerInfo**	
Input Data	<none>	
Output Data	ServerInfo	s-Server
Purpose To return information about the current server. Individual properties are accessed from the ServerInfo structure.		
Alternative representations	Examples	
Method of a Server class.	Server.Info	
General function/method to return named value.	GetServerProperty(*PropertyName*) Server.Property(*PropertyName*)	
Specific function/property to return a value.	GetServerPropertyVersion Server.Version	

3.1.2 GetTerminologyInfo

Service Name	**GetTerminologyInfo**	
Input Data	<none>	This assumes a single Terminology is in use. If alternative terminologies, editions or versions are accessible through the server then a selection may need to be specified here.
Output Data	TerminologyInfo	s-TerminologyInfo
Purpose To return information about the current terminology. Individual properties are accessed from the TerminologyInfo structure.		
Alternative representations	Examples	
Method of a Server class	Server.Terminology.Info	
Method of a Terminology class.	Terminology.Info	
General function/method to return named value.	GetTerminologyProperty(*PropertyName*) Terminology.Property(*PropertyName*)	
Specific function/property to return a value.	GetTerminologyPropertyVersion Terminology.Version	

3.1.3 GetExtensionInfo

Service Name	**GetExtensionInfo**	
Input Data	Index	Index to an item from the c-Extensions collection available to the server.
Input Data (alternative)	Namespace	String or SCTID
	Version (optional)	Integer or String
	Edition (optional)	String
Output Data	Extension	s-Extension
Purpose To return information about one of the Extensions available to the server. Index is a number or key specifying a member of the collection of installed Extensions. Individual properties are accessed from the s-Extension structure.		

Alternative representations	Examples
Method of a Server class	Server.ExtensionInfos(Index) Server.ExtensionInfos(Namespace,Version,Edition)
Method of a Extension class.	Extension.Info
General function/method to return named value.	GetExtensionProperty(Index, *PropertyName*) Extension.Property(*PropertyName*)
Specific function/property to return a value.	GetExtensionPropertyVersion(Index) Extension.Version

3.1.4 Access to essential concept identifiers

The s-TerminologyInfo structure contains each of the concepts with structurally significant roles. These should be made accessible via functions or properties to allow the client to determine the appropriate identifiers.

3.1.5 GetSctIdInfo

Service Name	**GetSctIdInfo**	
Input Data	Id	String or SCTID
Output Data	SctId	s-Sctid
Description To return a structure containing information about an SCTID for a given input Id string. The structure includes information about: ❖ Whether the check-digit is valid ❖ The partition identifier (if check-digit is valid) ❖ The namespace identifier (if part of an Extension) ❖ Whether a Component with this identifier is available.		

Alternative representations	Examples
Method of a Server class	Server.SctIdInfo(Id)

Specific function/property to return a value.	SctIdIsValid(Id) SctId.IsValid

3.2 Core table access services

3.2.1 GetConcept

Service Name	**GetConcept**	
Input Data	ConceptId *OR* SnomedId *OR* Ctv3Id	String or SCTID These identifiers are lexically distinguishable. Therefore a single method/function with string input can be used irrespective of the nature of the identifier.
Output Data	Concept	s-Concept
Purpose To return a structure containing information about a Concept selected for the **ConceptId**, **SnomedId** or **Ctv3Id** supplied. These identifiers are lexically distinguishable. Therefore a single method/function with string input can be used irrespective of the nature of the identifier.		
Alternative representations	Examples	
Method of a Server class	Server.Concept(Id)	

3.2.2 GetDescriptionConcept

Service Name	**GetDescriptionConcept**	
Input Data	Description	s-Description (may be represented by DescriptionId)
Output Data	Concept	s-Concept
Purpose To return a structure containing information about the Concept represented by a specified **Description**. This may be accessed as a property of the s-Description structure instead of supporting this as a specific service.		
Alternative representations	Examples	
Method of a Description class	Description.Concept	

3.2.3 Access to properties of a concept

All relevant properties of a Concept should be accessible via function, properties or services applied to an instance of the s-Concept structure.

3.2.4 GetDescription

Service Name	**GetDescription**	
Input Data	DescriptionId	String or SCTID
Output Data	Description	s-Description
Purpose To return a structure containing information about a Description selected for the **DescriptionId** supplied.		
Alternative representations		Examples
Method of a Server class		Server.Description(Id)

3.2.5 GetDescriptions

Service Name	**GetDescriptions**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	LanguageCode (optional)	String
	DescriptionType (optional)	DescEnum
Output Data	Descriptions	c-Descriptions
Purpose To return a collection containing information about the set of Descriptions associated with a specified **Concept**. To return the Fully Specified Name, Preferred Term or Synonyms of a Concept in a particular language or dialect by applying a combination of a *LanguageCode* and *DescriptionType* or by applying a configuration containing these and other constraints.		
Alternative representations	Examples	
Method of a Concept class	Concept.Descriptions(*LanguageCode*,*DescriptionType*)	
Specific properties or methods for each possible *DescriptionTypes*.	Concept.PreferredTerm(*LanguageCode*) Concept.FullySpecifiedName(*LanguageCode*) Concept.Synonyms(*LanguageCode*)	

3.2.6 Access to properties of a description

All relevant properties of a Description should be accessible via function, properties or services applied to an instance of the s-Description structure.

3.2.7 GetRelationship

Service Name	**GetRelationship**	
Input Data	RelationshipId	String or SCTID
Output Data	Relationship	s-Relationship
Purpose To return a structure containing information about a Relationship selected for the specified **RelationshipId**. NOTE: This is not essential but is included for completeness.		
Alternative representations	Examples	
Method of a Server class	Server.Relationship(Id)	

3.2.8 GetRelationships

Service Name	**GetRelationships**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	CharacteristicType (optional)	CharEnum
	RelationshipGroup (optional)	Integer
	RelationshipType (optional)	s-Concept (may be represented by *ConceptId*)
Output Data	Relationships	c-Relationships
Purpose To return a collection containing information about the set of Relationships in which the *ConceptId1* matches a specified **Concept**. (i.e. where the Concept is the source of the Relationship). The set returned should be constrained by matching **CharacteristicType**, **RelationshipGroup** and/or **RelationshipType** where one or more of these are specified. By default the set should be returned ordered (or grouped) by *CharacteristicType*, *RelationshipGroup* and *RelationshipType*. An option to specify multiple values for *CharacteristicType* or *RelationshipType* filters may be useful.		
Alternative representations	Examples	
Method of a Concept class	Concept.Relationships(*CharacteristicType*, *RelationshipGroup*, *RelationshipType*)	

3.2.9 GetInverseRelationship

Service Name	**GetInverseRelationships**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	CharacteristicType (optional)	CharEnum
	RelationshipType (optional)	s-Concept (may be represented by *ConceptId*)
Output Data	Relationships	c-Relationships
Purpose To return a collection containing information about the set of Relationships in which the *ConceptId2* matches the specified **Concept** (i.e. where the Concept is the target of the Relationship). The set returned should be constrained by matching **CharacteristicType** and/or **RelationshipType** where either or both of these are specified. By default the set should be returned ordered (or grouped) by *CharacteristicType* and *RelationshipType*.		
Alternative representations	Examples	
Method of a Concept class	Concept.InverseRelationships(*CharacteristicType*, *RelationshipType*)	

3.2.10 Access to properties of a relationship

Individual Relationships for the c-Relationships collection should be accessible by index and/or iteration through the collection.

All relevant properties of a Relationship should be accessible via function, properties or services applied to an instance of the s-Relationship structure.

3.3 Subtype services

3.3.1 GetChildren

Service Name	**GetChildren**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	Configuration (optional)	s-Configuration
Output Data	Children	c-Concepts

Purpose

To return a collection containing information about the set of Concepts that are direct subtype or navigational children of a specified Concept. The children are target Concepts (match on *ConceptId2*) of Relationships which have the *RelationshipType* "IS A" and a source (match on *ConceptId1*) that is the specified Concept.

The Configuration setting (if specified) determines whether any Subsets are applied to restrict the set of child Concepts returned and may also specify that a Navigation Subset is applied in place of (or in addition to) the subtype Relationships to determine the set of child Concepts.

This service could be delivered by iteration through the results of a GetRelationships service using the GetConcept service to return each target Concept. However, as this service is frequently required and needs to be performed efficiently, it should therefore be made available to the client as a single optimized service.

Asynchronous implementation (recommmended)

When applied to Navigation Concept or a subtype child of Inactive Concept this return a very large collection. Therefore it is recommended an appropriate approach is employed to support asynchronous calls or that an exception is raised when more that a maximum number of children exists. Options include:

- Event oriented approaches (e.g. an Event is triggered for each child is found).
- "First"/"Next" variants of this service allowing a step through the descendants.
- A "Start"/"Continue" variants of this returning partial sets limited by size or timeout.

Alternative representations	Examples
Method of a Concept class	Concept.Children(Configuration)

3.3.2 GetParents

Service Name	**GetParents**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	Configuration (optional)	s-Configuration
Output Data	Parents	c-Concepts

Purpose

To return a collection containing information about the set of Concepts that are direct supertype or navigational parents of a specified Concept. The parents are source Concepts (match on *ConceptId1*) of Relationships which have the *RelationshipType* "IS A" and a target (match on *ConceptId2*) that is the specified Concept.

The Configuration (if specified) determines whether any Subsets are applied to restrict the set of parent Concepts returned and may also specify a Navigation Subset is applied in place of (or in addition to) the subtype Relationships to determine the set of parent Concepts.

This service could be delivered by iteration through the results of a GetRelationships service using the GetConcept service to return each target Concept. However, as this service is frequently required and needs to be performed efficiently, it should therefore be made available to the client as a single optimized service.

Alternative representations	Examples
Method of a Concept class	Concept.Parents (Configuration)

3.3.3 GetDescendants

Service Name	**GetDescendants**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
Output Data	Descendants	c-Concepts

Purpose

To return a collection containing information about the set of Concepts that are subtype descendents of a specified Concept. The descendants are the target Concepts (match on *ConceptId2*) of Relationships which have the *RelationshipType* "IS A" and a source (match on *ConceptId1*) that is either the specified Concept or another descendant of that Concept..

This service could be delivered by a tree-walk using recursive call to the GetChildren service. However, this service needs to be performed efficiently without returning duplicates, which may arise through multiple inheritance. Therefore a server should provide an optimized service to deliver this rather than requiring the client to perform the tree-walk and discard duplicates.

Asynchronous implementation (required)

When applied to a Concept which is at a high level in the subtype hierarchy this service may take a perceptible time to complete and will return a very large collection. In the extreme case all Concepts in the terminology would be returned by applying this service to the Root Concept. Therefore it is recommended that an appropriate approach is employed to support asynchronous calls. Options include:

- Event oriented approaches (e.g. an Event is triggered for each descendant found).
- "First"/"Next" variants of this service allowing a step through the descendants.
- A "Start"/"Continue" variants of this returning partial sets limited by size or timeout.

Alternative representations	Examples
Method of a Concept class	Concept.Descendants

3.3.4 GetAncestors

Service Name	**GetAncestors**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
Output Data	Ancestors	c-Concepts

Purpose

To return a collection containing information about the set of Concepts that are supertype ancestors of a specified Concept. The ancestors are the source Concepts (match on *ConceptId1*) of Relationships which have the *RelationshipType* "IS A" and a target (match on *ConceptId2*) that is either the specified Concept or another descendant of that Concept..

This service could be delivered by a tree-walk using recursive calls to the GetParents service. However, this service needs to be performed efficiently without returning duplicates, which may arise through multiple inheritance. Therefore a server should provide an optimized service to deliver this rather than requiring the client to perform the tree-walk and discard duplicates.

Asynchronous implementation (optional)

The maximum size of an ancestors collection is less than the maximum size of a descendants collection. However, for consistency it may be helpful to match this interface with GetDescendants.

Alternative representations	Examples
Method of a Concept class	Concept.Ancestors

3.3.5 GetTopLevel

Service Name	**GetTopLevel**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	SpecialFlag (optional)	Boolean
Output Data	TopLevelConcepts	c-Concepts

Purpose

To return any Top-Level Concepts that are supertypes ancestor(s) for a specified **Concept**.

If **SpecialFlag** is true then if the actual Top-Level Concept is "Special Concept" the server should return the specific subtype of Special Concept. Thus for a Special Concept when the flag is set this service returns the same result as the GetParents service (i.e. Navigation Concept, Namespace Concept or one of the subtypes of Inactive Concept).

In some cases, a Concept may have more than one Top-Level Concept. Therefore, this service is shown as returning a collection of Concepts rather than a single Concept. However, usually there will only be a single Concept in this collection.

This service could be delivered by a tree-walk using recursive calls to the GetParents service or using the GetAncestors service and discarding all Concepts that do not meet the criteria for Top-Level Concepts. Therefore a server should provide an optimized service to deliver this rather than requiring the client to perform this operation.

Alternative representations	Examples
Method of a Concept class	Concept.TopLevel

3.3.6 IsAncestor and IsDescendant

Service Name	**IsAncestor**	
	IsDescendant (alternative)	
Input Data	AncestorConcept	s-Concept (may be represented by *ConceptId*)
	DescendantConcept	s-Concept (may be represented by *ConceptId*)
	IncludeInactive (optional)	Boolean
Output Data	Result	Boolean

Purpose

To determined whether a specified **DescendantConcept** is a subtype descendant (or supertype ancestor) of the specified. **AncestorConcept**.

The **IncludeInactive** parameter is only relevant if the *DescendantConcept* is an Inactive Concept. In this case the test returns true is Concept has a Historical Relationship with a valid descendant of the putative ancestor.

The two tests IsDescendant and IsAncestor can be performed by a single service. However, optimizations may differ between situations in which several potential descendant need to be tested against a single *AncestorConcept* and other situations in which several potential ancestors are tested against a single *DescendantConcept*.

These services could be delivered by calling the GetAncestors service on the *DescendantConcept* testing the returned collection against the putative *AncestorConcept*. However, this type of test is unlikely to deliver satisfactory performance. Much more efficient methods of testing subtype subsumption are suggested in the Technical Implementation Guide [07_A-6.2.5.4]. To support access to these a specific rapid service performing this test is required.

Alternative representations	Examples
Method of a Concept class	Concept.IsAncestor(DescendantConcept, IncludeInactive) Concept.IsDescendant(AncestorConcept, IncludeInactive)

3.4 Text search services

3.4.1 GetMatchingDescriptions

Service Name	**GetMatchingDescriptions**	
Input Data	Key	String
	Mode	SearchModeEnum
	Refinements (optional)	c-SearchRefineEnums
	Configuration (optional)	s-Configuration
	SortOrder (optional)	c-SortOrderEnums
	TextFilter (optional)	String
Output Data	Descriptions	c-Descriptions

Purpose

To return a set of Descriptions that contain a word, phrase, set of words specified in the *Key*.

Key is the primary expression of words or a phrase to be searched for. The rules for parsing the *Key* prior to a search may vary according to *Mode* but should be consistent with the rules applied in generation of the SNOMED CT WordKey Tables.

Mode specifies a single primary characteristic of the search (e.g. by single word, words any order, complete phrase, phrase at start, phrase at end, etc.).

Refinements specifies additional features or constraints to be applied to the search (e.g. case sensitivity, pattern matching, exclude of multiple matches for a single Concept or for a descendants of matched Concept, allow different word forms, allow word equivalents).

A server need not support all search modes or search refinements listed in the specification. A server may support additional search modes not listed in this specification. The search modes supported should be specified in the c-Server structure and

Configuration structure should allow constraints to be applied to the search. These should include filtering by one or more of the following.

- *DescriptionType* in the specified language or dialect (i.e. in a given Language Subset)
- Whether the Description is an Active Description.
- *DescriptionStatus* and/or *ConceptStatus* of the associated Concept
- Presence in (or absence from) specified Subsets
- Whether the Concept associated with the Description is a descendant of a specified ancestor Concept.

SortOrder may be used to specify the order in which matching Descriptions are returned (e.g. alphabetical, according to term length or word count and whether a presort by *DescriptionStatus* and/or *DescriptionType* is required.

TextFilter may be used to enable specific modes or refinements to include an expression for applying additional text pattern based filtering to each the term of each Description to determined whether or not it should be returned.

Asynchronous implementation (required)

Some searches may return a very large set of matches. Some complex searches may take a significant time to complete. Therefore it is recommended that an appropriate approach is employed to support asynchronous calls to this method. Options include:

- Event oriented approaches (e.g. an Event is triggered for each Description found).
- "First"/"Next" variants of this service allowing a step through the matching Descriptions.
- "Start"/"Continue" variants of this service returning partial sets limited by size or timeout.

An additional feature which is likely to improve usability is a rapid indication of the likely maximum number of matches for a given search specification. For example, an estimate of the number of simple matches on a given *Key* before applying other constraints may allow the client to decide whether a search is likely to be tractable.

Alternative representations	Examples
As a method of a Search class	Search.MatchingDescriptions(*Key*, … other parameters as needed)

3.5 Post-coordination support services

3.5.1 GetPotentialElaborations

Service Name	**GetPotentialElaborations**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
Output Data	Elaborations	c-Elaborations

Purpose

To return sets of qualifiers and potential refinements of defining characteristics that may be applied to a specified **Concept**. The potential elaborations of a Concept do not define that Concept but may be applied to it to extend or clarify its meaning in a given context.

The potential elaborations are derived from the Relationships of the specified Concept as the union of the following:

- For each refinable defining Relationship
  - For each subtype child or the target Concept (matching on *ConceptId2*) Relationship an s-Elaboration containing the child concept as its Value and with *IsRefinable*=true.
  - This does not apply to subtype, part-of or similar hierarchical Relationships (these should be marked as "not refinable").
- For each mandatory to refine qualifying Relationship
  - For each subtype child or the target concept (matching on *ConceptId2*) of that Relationship an s-Elaboration containing the child concept as its Value and with *IsRefinable*=true.
- For each refinable qualifying Relationship
  - An instance of s-Elaboration with the same Value set to the target concept (matching *ConceptId2*) and with *IsRefinable*=true.
- For each non refinable qualifying Relationship
  - An instance of s-Elaboration with the same Value set to the target concept (matching *ConceptId2*) and with *IsRefinable*=false.

Alternative representations	Examples
Method of a Concept class	Concept.*Elaborations*

3.5.2 GetConceptCanonical

Service Name	**GetConceptCanonical**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
Output Data	CanonicalExpression	s-ConceptExpression

Purpose

To return the canonical form for a specified **Concept** based either on the distributed Canonical Table or an appropriate computation from the defining characteristics.

Alternative representations	Examples
Method of a Concept class	Concept.Canonical

3.5.3 GetExpressionCanonical

Service Name	**GetExpressionCanonical**	
Input Data	Expression	s-ConceptExpression
Output Data	CanonicalExpression	s-ConceptExpression
Purpose To return the canonical form computed from a specified post-coordinated Expression consisting of one or more concept and a set of associated elaborations. The primary use of this is to convert a post-coordinated concept expression generated by the user of client application into a form in which it can be test for equivalence and/or subsumption against the canonical form a specified pre or post-coordinated concept expression.		
Alternative representations	Examples	
Method of a Concept class	Concept.Canonical(AdditionalFocusConcepts, *Elaborations*)	
Method of a Concepts collection class	Concepts.Canonical(*Elaborations*)	
Method of a ConceptExpression class	ConceptExpression.Canonical	

3.5.4 IsDescendantExpression

Service Name	**IsDescendantExpression** or **IsAncestorExpression**	
Input Data	AncestorExpression	s-ConceptExpression
	DescendantExpression	s-ConceptExpression
Output Data	Result	Boolean
Purpose To tests the is a specified DescendantExpression is a subtype of the AncestorExpression expressions in much the same way as the IsDescendant service. However, in this case the elaborations applied to the two expressions are also considered. The service may require that the two expressions are already canonical. However, a possible enhancement would be test this and to automatically apply the GetExpressionCanonical service if required.		
Alternative representations	Examples	
Methods of a ConceptExpression class	ConceptExpression.IsDescendant(AncestorExpression) ConceptExpression.IsAncestor(DescendantExpression)	

3.6 Data retrieval support services

3.6.1 GetDescentSQL

Service Name	**GetDescentSQL**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	IncludeStatuses (optional)	c-StatusEnums
	IdTypes (optional)	c-IdTypeEnums
Output Data	SqlPredicate	String

Purpose

To return an SQL predicate statement (or fragments of such a statement) including a list of ConceptIds identifiers for the specified Concept and all its subtype descendants.

The string returned should be a comma separated list of Ids with each identifier enclosed in quotation marks. This allows the returned string to be inserted into an SQL query in a line of the form:

- WHERE Identifier IN ([returned string]).

More sophisticated development of this service to include predicates for particular post-coordinated representation may also be feasible. However, a consistent representation of such queries in a form that can be applied to a variety of different data stores for SNOMED CT encoded data has not yet been specified.

If IncludeStatuses is present, it explicitly includes Concepts of the specified *ConceptStatus* values and excludes all Concepts with other *ConceptStatus* values. For the purposes of inclusion an Inactive Concept is represented in the output if it has a Historical Relationship with a target that is a descendant of the specified Concept. If IncludeStatuses is absent, all Active Concepts are included and all Inactive Concepts are excluded.

If IdTypes is present the SQL predicate should be expressed using the type or types of identifier specified for each included Concept. Thus a predicate can be generated using legacy SnomedId and Ctv3Id values in place of or in addition to the SNOMED CT *ConceptId*. If IdTypes is absent the SNOMED CT *ConceptId* should be used.

Asynchronous implementation (required)

When applied to a Concept which is at a high level in the subtype hierarchy this service may take a perceptible time to complete and will return a very large SQL expression. In these circumstances some support for asynchronous processing or interruption of the process is required. A possible option is simply to cap the maximum number of identifiers that may be included.

Alternative representations	Examples
Method of a Concept class	Concept.DescentSQL(IncludeStatuses, IdTypes)

3.7 Subsets configuration services

3.7.1 GetSubset

Service Name	**GetSubset**	
Input Data	SubsetId	String or SCTID
	AsLatestVersion (optional)	Boolean
Output Data	Subset	s-Subset
Purpose To return a structure containing information about a Subset selected for the **SubsetId** supplied. **AsLatestVersion** when True requests the Subset with the most recent version of a Subset. That is the Subset with ❖ The highest *SubsetVersion* number ❖ and ❖ A SubsetOrginalId which either: ✧ Matches the specified *SubsetId* *or* ✧ Shares the same *SubsetOriginalId* as the Subset that matches the specified SubsetId.		
Alternative representations	Examples	
Method of a Server class	Server.Subset(Id)	

3.7.2 GetSubsets

Service Name	**GetSubsets**	
Input Data	SubsetType (optional)	String
	SubsetName (optional)	String (may be a matching pattern)
	LanguageCode (optional)	String
	RealmId (optional)	String
	ContextId (optional)	String
Output Data	Subsets	c-Subsets
Purpose To return a collection of Subsets that match a specified set of criteria.		
Alternative representations	Examples	
Method of a Server class	Server.Subsets(Criteria)	

3.7.3 Access to the properties of a Subset

All relevant properties of a Subset should be accessible via function, properties or services applied to an instance of the s-Subset structure.

3.7.4 Enabling subset configuration and selection

To configure search and navigation facilities the server should allow client selected Subsets (represented by s-Subset structures) to be added to or removed from appropriate properties of the s-Configuration structure.

3.8 Cross mapping services (optional feature)

3.8.1 GetCrossMapSet

Service Name	**GetCrossMapSet**	
Input Data	MapSetId	String or SCTID
Output Data	CrossMapSet	s-CrossMapSet
Purpose To return a structure containing information about a Cross Map Set selected for the MapSetId supplied.		
Alternative representations	Examples	
Method of a Server class	Server.CrossMapSet(Id)	

3.8.2 GetCrossMapSets

Service Name	**GetCrossMapSets**	
Input Data	MapSetType (optional)	String
	MapSetName (optional)	String (may be a matching pattern)
	MapSetSchemeId (optional)	String
	MapSetSchemeName (optional)	String (may be a matching pattern)
	RealmId (optional)	String
Output Data	Subsets	c-Subsets
Actions	<none>	
Purpose To return a collection of Cross Map Sets that match a specified set of criteria.		
Alternative representations	Examples	
Method of a Server class	Server.CrossMapSets(Criteria)	

3.8.3 Access to the properties of a CrossMapSet

All relevant properties of a Cross Map Set should be accessible via functions, properties or services applied to an instance of the s-CrossMapSet structure.

3.8.4 GetCrossMaps

Service Name	**GetCrossMaps**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	CrossMapSet	s-CrossMapSet (may be represented by MapSetId)
Output Data	CrossMaps	c-CrossMaps
Purpose To return a collection containing information about the set of Cross Maps associated with a specified **Concept** in a specified **CrossMapSet**.		
Alternative representations	Examples	
Method of a Concept class	Concept.CrossMaps(CrossMapSet)	
Method of a CrossMapSet class	CrossMapSet.CrossMaps(Concept)	

3.8.5 Access properties of a Cross Map

It should be possible to iterate through a c-CrossMaps collection in and all relevant properties of each Cross Map should be accessible via function, properties or services applied to an instance of the s-CrossMap structure.

3.8.6 GetCrossMapTarget

Service Name	**GetCrossMapTarget**	
Input Data	TargetId	String or SCTID
Output Data	CrossMapTarget	s-CrossMapTarget
Purpose To return a structure containing information about a Cross Map Target selected for the **TargetId** supplied.		
Alternative representations	Examples	
Method of a Server class	Server.CrossMapTarget(Id)	
Method of or property of a CrossMap class	CrossMap.Target	

3.8.7 Access properties of the a Cross Map Target

All relevant properties of a Cross Map Target should be accessible via functions, properties or services applied to an instance of the s-CrossMapTarget structure.

3.9 Component history services (optional feature)

3.9.1 GetHistory

Service Name	**GetHistory**	
Input Data	ComponentId	SCTID or String
Output Data	History	c-History
Purpose To return a collection the Component History information associated with a component specified by the ComponentId. The ComponentId may identify any type of SNOMED CT component (e.g. a Concept, Description, Relationship, Subset, Cross Map Set). The c-History collection should contain all Component History information about the specified component and should order this data according to the ReleaseVersion. Note that the availability of history may differ for particular type of component.		
Alternative representations	Examples	
Method of a specific component class	Concept.History Description.History Relationship.History Subset.History CrossMapSet.History	
Method of a Server or Terminology class	Server.History(ComponentId) Terminology.History(ComponentId)	

3.9.2 GetReferences

Service Name	**GetReferences**	
Input Data	ComponentId	SCTID or String
Output Data	References	c-References
Purpose To return a collection containing the References Table information associated with a component specified by the ComponentId. The ComponentId may identify any type of SNOMED CT component (e.g. a Description, Relationship, Subset or Cross Map Set). The c-Reference collection should contain all Component Reference information about the specified component and should order this data according to the ReleaseVersion. Note that the availability of References differs for particular types of component. In particular concepts have Historical Relationships rather than References.		
Alternative representations	Examples	
Method of a specific component class	Description.References	
Method of a Server or Terminology class	Server.Reference(ComponentId) Terminology.Reference(ComponentId)	

4 User Interface Services

4.1 User Interface Component Display Services

4.1.1 ShowDescription

Service Name	**ShowDescription**	
Input Data	Description	s-Description (may be represented by DescriptionId)
	DisplayOptions	d-Description
Output Data	<none>	
Action	Display the specified **Description** including properties specified by **DisplayOptions**.	
Purpose To display a specified **Description**.		
Alternative representations	Examples	
Method of a Display control class	Display.Show(Description)	
Method of a Description class	Description.Display(DisplayOptions)	

4.1.2 ShowConcept

Service Name	**ShowConcept**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	DisplayOptions	d-Concept
Output Data	<none>	
Action	Display **Concept** including properties specified by **DisplayOptions**.	
Purpose To display a specified **Concept**.		
Alternative representations	Examples	
Method of a Display control class	Display.Show(Concept)	
Method of a Concept class.	Concept.Display(DisplayOptions)	

4.2 User Interface List and Outline Displays services

4.2.1 ShowMatchingDescriptions

Service Name	**ShowMatchingDescriptions**	
Input Data Maybe pre-set properties, selection in user interface or specific input to API service call.	*Key*	String
	Mode	SearchModeEnum
	Refinements (optional)	c-SearchRefineEnums
	Configuration (optional)	s-Configuration
	SortOrder (optional)	c-SortOrderEnums
	TextFilter (optional)	String
	DisplayOptions	d-DescriptionList
Output Data	<none>	
Action	Display a list of matching Descriptions including Term and optionally also the DescriptionId and an indication of *DescriptionStatus* and *DescriptionType*.	
Purpose To display the results of a text search and then allow the user to select one or more Descriptions and their associated Concepts.		
Reference services and data structures or collections used ❖ GetMatchingDescriptions		
Asynchronous requirement Display should allow interruption of list display by user interventions such as typing a new search key or selecting an item in the displayed list. It should also be possible to programmatically interrupt the list display.		

Alternative representations	Examples
Method of a Search control class	Search.Show(*Key*, *Mode*, Refinements, Configuration, SortOrder, TextFilter)
Method of a Descriptions collection class populated with the results of a search.	Descriptions.Display(DiplayOptions)

4.2.2 ShowHierarchy

Service Name	**ShowHierarchy**	
Input Data Maybe pre-set properties, selection in user interface or specific input to API service call.	FocusConcept	s-Concept
	Configuration (optional)	s-Configuration
	DisplayOptions	d-HierarchyList
Output Data	<none>	
Action	Display an outline (tree-view) of the subtype hierarchy or the navigational hierarchy specified in Configuration. Includes the parents of FocusConcept, the FocusConcept and the children of FocusConcept. The displayed hierarchy should be constrained by Subsets and other exclusions in the Configuration and the content of each element in the hierarchy display should be determined by DisplayOptions. Display options should as a minimum allow items to be displayed as PreferrredTerms with an option to display an indication of *ConceptStatus* and/or the *ConceptId*.	
Purpose To display a subtype hierarchy or a specified navigational hierarchy. The hierarchy list once displayed should support expansion of a selected item in the tree to show the parents and children of the Concept represented by that item.		
Reference services and data structures or collections used ❖ GetParents ❖ GetChildren		
Asynchronous requirement Display should allow interruption of list display by user interventions such as selecting an item in the displayed list. It should also be possible to programmatically interrupt the list display.		
Alternative representations	Examples	
Method of a Search control class	Search.Show(*Key*, *Mode*, Refinements, Configuration, SortOrder, TextFilter, DisplayItems, FormControl	
Method of a Descriptions collection class populated with the results of a search.	Descriptions.Display(DiplayOptions)	

4.2.3 ShowConceptDetail

Service Name	**ShowConceptDetail**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	DisplayOptions	d-Concept
	Configuration (optional)	s-Configuration
	DescriptionOptions (optional)	d-Description
	CharacteristicTypes (optional)	c-CharEnums
	RelationshipTypes (optional)	c-Concepts (may be represented as collection of *RelationshipType* Ids)
	RelationshipOptions (optional)	d-Relationship
Output Data	<none>	
Action	Display Concept including properties specified by DisplayOptions together with: ❖ Descriptions associated with the Concept excluding those that do not meet the constraints of the specified Configuration. ✧ Displayed in accordance with DescriptionOptions ❖ Associated Relationships that have one of the specified CharacteristicTypes and one of the specified RelationshipTypes. ✧ Displayed in accordance with RelationshipOptions	
Purpose To display a specified Concept including associated Descriptions and or Relationships. Alternative parameters may be useful. For example, to display different Relationships in different ways. The display options for Descriptions and Relationships should support display in lists from which an individual item can be selected by the user.		
Reference services and data structures or collections used ❖ GetDescriptions ❖ GetRelationships		

Alternative representations	Examples
Method of a Display control class In practice many of the parameters shown here could be represented as properties of the control class.	Display.ShowDetail(Concept, DisplayOptions, Configuration, DescriptionOptions, *CharacteristicTypes*, *RelationshipTypes*, RelationshipOptions)
Method of a Concept class.	Concept.Display(Control, DisplayOptions, Configuration, DescriptionOptions, *CharacteristicTypes*, *RelationshipTypes*, RelationshipOptions)

4.2.4 ShowPotentialElaborations

Service Name	**ShowPotentialElaborations**	
Input Data	Concept	s-Concept (may be represented by *ConceptId*)
	DisplayOptions	d-ElaborationList
Output Data	<none>	
Action	Display the ConceptExpression either in the form expressed or in its canonical form (if AsCanonical is True). Properties are display as specified by DisplayOptions.	
Purpose To qualifying characteristics and refinable defining characteristics for a specified Concept in a form that allows the user to select appropriate values to elaborate the meaning of a select Concept.		
Reference services and data structures or collections used ❖ GetPotentialElaborations		
Alternative representations	Examples	
Method of a Display control class	Display.ShowPotentialElaborations(Concept)	
Method of a Concept class.	Concept.ShowElaborations(Concept)	

4.2.5 ShowConceptExpression

Service Name	**ShowConceptExpression**	
Input Data	ConceptExpression	s-ConceptExpression
	AsCanonical	Boolean
	DisplayOptions	d-ConceptExpression
Output Data	<none>	
Action	Display the ConceptExpression either in the form expressed or in its canonical form (if AsCanonical is True). Properties are display as specified by DisplayOptions.	
Purpose To display a specified ConceptExpression included the specified Concepts and elaborarions. Note that this service can display the canonical form of a Concept because the minimal form of ConceptExpression is a single Concept with no elaborations. The elaborations should be displayed in a list from which an individual elaboration can be selected.		
Reference services and data structures or collections used ❖ GetExpressionCanonical ❖ GetCanonical		
Alternative representations	Examples	
Method of a Display control class	Display.ShowConceptExpression(ConceptExpression, AsCanonical)	
Method of a ConceptExpression class.	ConceptExpression.Display(AsCanonical, DisplayOptions)	

4.2.6 GetSelectedItem

Service Name	**GetSelectedItem**	
Input Data	<none>	
Output Data (for a list or hierarchy of concepts)	Concept	s-Concept
Output Data (for a list of descriptions)	Description	s-Description
	Concept (optional)	s-Concept
Output Data (for a list of relationships)	Relationship	s-Relationship
	Concept1 (optional)	s-Concept
	RelationshipType (optional)	s-Concept
	Concept2 (optional)	s-Concept
Output Data (for a list of elaborations)	*Elaboration*	s-Elaboration
	RelationshipGroup (optional)	Integer
	Attribute (optional)	s-Concept
	Value (optional)	s-Concept
Output Data (for a list of subsets)	Subset	s-Subset
Output Data (for a list of CrossMapSets)	CrossMapSet	s-CrossMapSet
Output Data (for a list of CrossMaps)	CrossMap	s-CrossMap
	CrossMapTarget	s-CrossMapTarget
Purpose To return information about an item selected by a user in an UI-control such as a list or hierarchy tree.		
Alternative representations	Examples	
Method of a user interface control class	Control.GetSelectedItem	
Event triggered by user selection in a given control.	Control.SelectionChanged(SelectedComponents)	

4.3 User Interface Subset services

4.3.1 ShowSubsets

Display list showing the Subsets in a collection of Subsets. This may be the collection of available Subsets (optionally filtered by *SubsetType*, *LanguageCode* and/or Realm) or a collection of Subsets currently in use in a particular Configuration.

4.3.2 ShowSubset

Display details of a specified Subset.

4.4 User Interface cross mapping services

4.4.1 ShowCrossMapSets

Display list of available Cross Map Sets.

4.4.2 ShowCrossMapSet

Display details of a specified Cross Map Set.

4.4.3 ShowConceptCrossMaps

Display list of Cross Maps for a selected Concept in a specified Cross Map Set.

4.4.4 ShowCrossMap

Display details of a selected Cross Map and its associated Cross Map Target.

4.4.5 ShowCrossMapTarget

Display details of a specified Cross Map Target.

5 Data Definitions

5.1 Data Types

5.1.1 Basic Data Types

	Description	**String representation**	**Alternative representations**
String	String of UTF-8 encoded characters.	Length stated where relevant but otherwise variable/unlimited.	UTF-16
Boolean	Zero=False Any other values=True	True "1" (or "-1") False "0"	Boolean or Int (any size)
Integer	Signed 32-bit integer. May be restricted to 8 or 16 bit in some data structures.	String of 1 to 10 digits Optional preceding "-" Maximum length may be reduced in some structures.	Int(32)
SCTID	Unsigned 64-bit integer.	String of 6 to 18 digits	Int * 64
Enum	An integer representing a value from a set of enumerated types. See specific enumerated types listed below		

5.1.2 Enumerated Types detailed in SNOMED CT release documentation

StatusEnum	An enumerated type representing ComponentStatus values.	String of 1 to 2 digits	Char or Int (any size). (See TIG / TRG for explanation of values)
DescEnum	An enumerated type representing *DescriptionType* values.	String of 1 digit	
CharEnum	An enumerated type representing *CharacteristicType* values.	String of 1 digit	
RefineEnum	An enumerated type representing Refinability values.	String of 1 digit	

SubsetEnum	An enumerated type representing *SubsetType* values..	String of 1 digit	
ChangeEnum	An enumerated type representing ChangeType values..	String of 1 digit	
ReferEnum	An enumerated type representing ReferenceType values.	String of 1 digit	
MapEnum	An enumerated type representing CrossMapSetType values.	String of 1 digit	
RuleEnum	A specific enumerated type representing CrossMapRuleType values.	String of 1 or 3 digits	Char or Int (16 or 32). TO BE DEFINED

5.1.3 Enumerated Types defined in this document

FeatureEnum	An enumerated type representing server feature values.	String of 1 or 3 digits	Char or Int (16 or 32). TO BE DEFINED
SearchModeEnum	An enumerated type representing different types of text search mode. 1 = single word search 2 = words any order search 3 = words in order 4 = complete term match 5 = phrase match anywhere in term 6 = phrase match at start of term 7 = phrase match at end of term 8 to 31 reserved 32 and above server specific values	String of 1 to 3 digits	Char or Int (16 or 32) AS DEFINED HERE
SearchRefineEnum	An enumerated type representing different types of text search refinements. 1 = case dependent 2 = whole words only 3 = allow pattern match 4 = allow alternative word forms 5 = allow word equivalents 6 = exclude duplicate hits on a concept 7 = exclude matching descendants 8 to 31 reserved 32 and above server specific values	String of 1 to 3 digits	Char or Int (16 or 32) AS DEFINED HERE
SortOrderEnum	An enumerated type representing different sort orders to be applied to a	String of 1 to 3 digits	Char or Int (16 or 32) AS DEFINED

	collection of Descriptions (e.g. the results of a search). 1 = alphabetical (ascending) 2 = character count in term (lowest first) 3 = word count in term (lowest first) 4 = status sort (active before inactive) 5 = type sort (preferred, synonym, fully specified)		HERE
IdTypeEnum	An enumerated type indicating which type or types of identifier should be shown or exported to represent a Concept. 1 = SNOMED CT *ConceptId* 2 = SNOMED CT DescriptionId 3 = Legacy SnomedId 4 = Legacy Ctv3Id (Read Code)	String of 1 to 3 digits	Char or Int (16 or 32) AS DEFINED HERE

5.2 Data Structures

5.2.1 Introduction

This sections outlines the content of data structures used to express some of the API requirements. Many of these are directly equivalent to the structures of the SNOMED CT release tables. However, in some cases the structures contain additional derived properties (e.g. the s-Concept structure contains an "IsActive" property derived from *ConceptStatus*).

Some optional properties are included in the structures. In most cases, these are relatively complex collections of properties derived by traversing logical joins between two or more release tables. These are regarded as optional because the same information can be accessed in other ways using one or more of the services defined in this specification. However, in an object oriented implementation these may usefully be implemented as properties of classes that represent these structures.

5.2.2 Information about the server and available terminology components

5.2.2.1 Information about the SNOMED CT enabled server

s-Server	Type	READ ONLY
Name	String (255)	The name of the terminology server.
Supplier	String (255)	The name of the owner or developer.
Version	String (16)	A version number applicable to the server.
SupportedFeatures	c-FeatureEnums (collection)	The set of features supported by this server
SearchModes	c-SearchModeEnums (collection)	The set of search modes supported by this server
SearchRefinements	c-SearchRefineEnums (collection)	The set of search modes supported by this server
Optional additional properties		
Terminology	s-TerminologyInfo	The current loaded instance of SNOMED CT
ExtensionInfos	c-Extensions (collection)	The set of currently available Extensions.
Subsets	c-Subsets (collection)	The set of currently available Subsets.
CrossMapSets	c-CrossMapSets (collection)	The set of currently available CrossMapSets.
Configurations	c-Configurations (collection)	The set of available configurations in a current instance of the server.

5.2.2.2 Information about the available version of the main body of SNOMED CT

s-TerminologyInfo	Type	READ ONLY
ReleaseVersion	Integer	Version as ISO formatted release date (E.g. "20020731")
ReleaseEdition	String (16)	E.g: "US", "UK", "ES"
ReleaseName	String (255)	E.g: SNOMED CT July 2002 Release: 20020731 [R]
ReleaseStatus	String (1)	E.g: "R" for release "D" for developmental etc.
ReleaseNote	String (255)	Additional note on the release.
CodeSystemOid	String (255)	An ISO OID identifying SNOMED CT or this release of SNOMED CT.
RootConcept	s-Concept	The Root Concept of the current release of the terminology.
IsaConcept	s-Concept	The Concept that represents the Relationship Type of the Relationship "IS A" (i.e. the subtype relationship).
AttributeConcept	s-Concept	The Top-Level Concept for all Relationship Types (also known as attributes).
ValueConcept	s-Concept	The Top-Level Concept for all Concepts that may only be used as values of specified attributes.
SpecialConcept	s-Concept	The Top-Level Concept for all special Concepts.
NavigationConcept	s-Concept	The Special Concept supertype of Navigation Concepts.
NamespaceConcept	s-Concept	The Special Concept supertype of Namespace Concepts.
InactiveConcept	s-Concept	The Special Concept supertype of Inactive Concepts.
SameAsConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "SAME AS".
MayBeAConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "MAY BE A".
ReplacedByConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "REPLACED BY".
WasAConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "WAS A".
MovedToConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "MOVED TO".
MovedFromConcept	s-Concept	The Concept that represents the Relationship Type of the Historical Relationship "MOVED FROM".

Optional additional properties		
DataInstallerVersion	String (16)	E.g: Version number of the server application that loaded this release into a server accessible form.
DataInstallationTime	String (255)	E.g: Time of loading data into a server accessible form.

5.2.2.3 Information about a SNOMED CT Extension available to the server

s-Extension	Type	READ ONLY
ExtensionNamespace	SCTID	The *ConceptId* of the namespace in which this Extension is managed.
ReleaseVersion	Integer	Version as ISO formatted release date (E.g. "20020731")
ReleaseEdition	String (16)	E.g. UK_GP.
ExtensionName	String (255)	E.g. UK Administration Extension GP release.
ReleaseStatus	String (1)	E.g. "R" for release "D" for developmental etc.
ReleaseNote	String (255)	E.g. Maintained by UK NHS IA.

5.2.2.4 Information about a configuration setting applicable to various services

s-Configuration	Type	CREATABLE AND UPDATABLE The precise details of this structure are open to extension and variation. This set of properties is suggested as a starting point for development
ConfigurationCode	String (16)	An arbitrary code defined by the server that can be used to refer to a configuration.
LanguageCode	String (16)	The default language code to be assumed while this configuration is when searching Descriptions.
AncestorConcept	s-Concept	A Concept which restricts the results of a search so that descendants of the Concept are included in the search result.
InclusionSubsets	c-Subsets (collection)	A collection of Subsets whose members are to be explicitly included or prioritised by a search.
ExclusionSubsets	c-Subsets (collection)	A collection of Subsets whose members are to be explicitly excluded by a search.
ExclusionStatuses	c-StatusEnums (collection)	Components with status values in this collection are excluded from searches.
ExclusionTypes	c-DescEnums (collection)	Descriptions with *DescriptionTypes* in this collection are exclude from searches.
Optional additional properties		
NavigationSubset	s-Subset	A Subset used to determine the content of a navigation hierarchy.
SearchMode	s-SearchModeEnum	The active/default text search mode.
SearchRefinements	c-SearchRefineEnums	The set of refinements to be applied to a text search.
SortOrder	c-SortOrderEnums	The sort order specification to be applied to a text searches.

5.2.3 Structures representing core SNOMED CT components and resources

5.2.3.1 Information about an SCTID

s-Sctid	Type	Read Only
Id	SCTID or String	The full identifier.
ComponentType	Enum	Indicates if this is a Concept (0), Description(1), Relationship (2), Subset (3), CrossMapSet (4) CrossMapTarget (5). Not known or invalid (-1).
NamespaceId	String (7)	Blank string if this is in the main body of SNOMED CT. Namespace-identifier part of the SCTID if this is part of an Extension.
IsValid	Boolean	True if the check-digit computation indicates as valid SCTID.
Exists	Boolean	True if a component with this SCTID exists in the terminology or Extensions available to the server.

5.2.3.2 Information about a Concept

s-Concept	Type	READ ONLY
ConceptId	SCTID	Derived directly from the SNOMED CT Concepts Table release data fields with the same names.
ConceptStatus	StatusEnum	
CTV3ID	String (5)	
SNOMEDID	String (8)	
IsPrimitive	Boolean	
IsActive	Boolean	Indicates that the *ConceptStatus* is one of the active values.
IsSpecial	Boolean	Indicates that the top-level ancestor is Special Concept.
Optional additional properties		This information can be derived from services specified in this document but direct access as properties of a class may be appropriate in some environments.
TopLevelType	s-Concept	If IsSpecial is false - the top-level ancestor. If IsSpecial is true - the Special Concept subtype which is this Concepts supertype parent.
Descriptions	c-Descriptions (collection)	The collection of Descriptions that contain a *ConceptId* that matches this Concept.
Relationships	c-Relationships (collection)	The collection of Relationships that contain a *ConceptId1* that matches this Concept.
InverseRelationships	c-Relationships (collection)	The collection of Relationships that contain a *ConceptId2* that matches this Concept.

CrossMaps	c-CrossMaps (collection)	The collection of Cross Maps that contain a Map*ConceptId* that matches this Concept
History	c-History (collection)	The collection of History that contain a ComponentId that matches this Concept.

5.2.3.3 Information about a Description

s-Description	Type	READ ONLY
DescriptionId	SCTID	Derived directly from the SNOMED CT Descriptions Table release data fields with the same names.
DescriptionStatus	StatusEnum	
Term	String (255)	
InitialCapitalStatus	Boolean	
Concept	s-Concept	The Concept that matches the *ConceptId* field of this Description.
DescriptionTypes	c-DescEnums (collection)	The *DescriptionType* value set by each of the available Language Subsets.
Optional additional properties		This information can be derived from services specified in this document but direct access as properties of a class may be appropriate in some environments.
History	c-History (collection)	The collection of Component History that contain a ComponentId that matches this Description.
References	c-References (collection)	The collection of References that contain a ComponentId that matches this Description.

5.2.3.4 Information about a Relationship

s-Relationship	Type	READ ONLY
RelationshipId	SCTID	Derived directly from the SNOMED CT Relationships Table release data fields with the same names.
CharacteristicType	CharEnum	
Refinability	RefineEnum	
RelationshipGroup	Integer	
Concept1	s-Concept	The Concept that matches the *ConceptId1* field of this Relationship.
RelationshipType	s-Concept	The Concept that matches the *RelationshipType* field of this Relationship.
Concept2	s-Concept	The Concept that matches the *ConceptId2* field of this Relationship.
Optional additional properties		This information can be derived from services specified in this document but direct access as properties of a class may be appropriate in some environments.

History	c-History (collection)	The collection of Component History that contain a ComponentId that matches this Relationship
References	c-References (collection)	The collection of References that contain a ComponentId that matches this Relationship.

5.2.4 Structures representing Subsets

5.2.4.1 Information about a Subset

s-Subset	Type	READ ONLY
SubsetId	SCTID	Derived directly from the SNOMED CT Subsets Table release data fields with the same names.
SubsetOriginalId	SCTID	
SubsetVersion	Integer	
SubsetName	String (255)	
SubsetType	SubsetEnum	
LanguageCode	String (8)	
RealmId	String (24)	
ContextId	String (18)	
Optional additional properties		This information can be derived from services specified in this document but direct access as properties of a class may be appropriate in some environments.
SubsetMembers	c-SubsetMembers (collection)	The collection of Subset Members that contain a SubsetId that matches this Subset.
History	c-History (collection)	The collection of Component History that contain a ComponentId that matches this Subset.
References	c-References (collection)	The collection of References that contain a ComponentId that matches this Subset.

5.2.4.2 Information about a Subset Member

s-SubsetMember	Type	READ ONLY
SubsetId	SCTID	Derived directly from the SNOMED CT Subset Members Table release data fields with the same names. Note that the LinkedId is only included for use when it is not an SCTID. In other cases LinkedComponent should be supported.
MemberStatus	Integer	
LinkedId	String (18)	
Member	s-Concept OR s-Description	The Component that matches the MemberId.
LinkedComponent	s-Concept OR s-Description	The Component that matches the LinkedId.

5.2.5 Structures representing history and reference information

5.2.5.1 Information about the history of a Component

s-History	Type	READ ONLY
ComponentId	SCTID	Derived directly from the SNOMED CT Component History Table release data fields with the same names.
ReleaseVersion	Integer	
ChangeType	ChangeEnum	
Status	StatusEnum	
Reason	String (255)	

5.2.5.2 Information about the reference from an inactive Component an active Component

s-Reference	Type	READ ONLY
ComponentId	SCTID	Derived directly from the SNOMED CT References Table release data fields with the same names.
ReferenceType	ReferEnum	
ReferComponent	s-Description OR s-Relationship OR s-Subset OR s-CrossMapSet	The Component that matches the ReferencedId.

5.2.6 Structures representing cross mapping information

5.2.6.1 Information about a Cross Map Set

s-CrossMapSet	Type	READ ONLY
MapSetId	SCTID	Derived directly from the SNOMED CT Component History Table release data fields with the same names.
MapSetName	String (255)	
MapSetType	MapEnum	
MapSetSchemeId	String (64)	
MapSetSchemeName	String (255)	
MapSetSchemeVersion	String (12)	
MapSetRealmId	String (24)	
MapSetSeparator	String (1)	
MapSetRuleType	RuleEnum	
Optional additional properties		This information can be derived from services specified in this document but direct access as properties of a class may be appropriate in some environments.
CrossMaps	c-CrossMaps (collection)	The collection of Cross Maps that contain a MapSetId that matches this Subset.
History	c-History (collection)	The collection of Component History that contain a ComponentId that matches this Cross Map Set.
References	c-References (collection)	The collection of References that contain a ComponentId that matches this Cross Map Set.

5.2.6.2 Information about a Cross Map Target

s-CrossMapTarget	Type	READ ONLY
TargetId	SCTID	Derived directly from the SNOMED CT Cross Map Targets Table release data fields with the same names.
TargetSchemeId	String (64)	
TargetCodes	String (255)	
TargetRule	String (255)	
TargetAdvice	String (255)	

5.2.6.3 Information about a Cross Map

s-CrossMap	Type	READ ONLY
MapSetId	SCTID	Derived directly from the SNOMED CT Cross Maps Table release data fields with the same names.
MapOption	Integer	
MapPriority	Integer	
MapTargetId	SCTID	

MapRule	String (255)	
MapAdvice	String (255)	
MapConcept	s-Concept	The Concept that matches the Map*ConceptId* of this Cross Map.

5.2.7 Structures representing post-coordinated aspects of SNOMED CT

5.2.7.1 Information about a post-coordinated representation of a Concept

s-ConceptExpression	Type	CREATABLE AND UPDATABLE
FocusConcepts	c-Concepts	A collection of one or more Concepts which together with the associated elaborations form a post-coordinated representation of Concept.
Elaborations	c-Elaborations (collection)	A collection of elaborations which when combined with the *FocusConcepts* form a post-coordinated representation of a Concept.

5.2.7.2 Information about the actual or potential post-coordinated elaboration of a Concept

s-Elaboration	Type	CREATABLE AND UPDATABLE
RelationshipGroup	Integer	The *RelationshipGroup* as specified in the Relationship from which this elaboration is derived. The default value (0) means ungrouped.
Attribute	s-Concept	The Concept that represents the *RelationshipType* elaborated (or potentially elaborated) by the associated value.
Value	s-Concept	The Concept that represents the value applies to elaborate the associated attribute.
IsRefinable	Boolean	This is only applicable in situations where the s-Elaboration structure is used to represent a potential rather than actual elaboration. If *IsRefinable* is false then the permitted actual elaboration must contain the specified value. If *IsRefinable* is false then the permitted actual elaboration may contain either the specified value or a subtype of that value. A qualifying Relationship which is specified as mandatory to refine is represented by a set of potential elaborations in which each subtype child value is expressed.

5.3 Data Collections

5.3.1 Collections of structures

Collection name	Type	Structures or services from which this type of collection may be derived (and type of access)	Filterable to sub collections by
c-Configurations	s-Configuration	Client Created (allow Create, Add & Remove)	
c-Extensions	s-Extension	Server (read only)	ExtensionNamespace
c-Concepts	s-Concept	Terminology (read only) Subtype Services (read only) Relationship Services (read only) Client Created (allow Create, Add & Remove)	
c-Descriptions	s-Description	Concepts (read only) Search services (allow Add & Remove)	*DescriptionStatus* *DescriptionType* *LanguageCode*
c-Relationships	s-Relationship	Concept.Relationships (read only) Concept.InverseRelationships (read only)	*CharacteristicType* *RelationshipType* *RelationshipGroup*
c-Subsets	s-Subset	Server (read only) Configuration.ActiveSubsets (allow Add & Remove) Configuration.ExclusionSubsets (allow Add & Remove)	*SubsetOriginalId* *SubsetType* *LanguageCode* Realm
c-SubsetMembers	s-SubsetMember	Subsets (read only)	MemberStatus LinkedId OR LinkedComponent
c-History	s-History	Concept, Description, Relationship, Subset or CrossMapSet. (read only)	
c-References	s-Reference	From Description, Relationship, Subset or CrossMapSet. (read only)	
c-CrossMapSets	s-CrossMapSet	Server (read only)	MapSetSchemeId MapSet*RealmId*
c-CrossMaps	s-CrossMap	Concept (read only)	MapSetSchemeId MapSet*RealmId*
		CrossMapSet (read only)	Map*ConceptId*

c-ConceptExpressions	s-ConceptExpression	Server UI actions (allow Add & Remove) Client created (allow Create, Add & Remove)	
c-Elaborations	s-Elaboration	GetPotentialElaborations (read only) Server UI actions (allow Add & Remove) Client created (allow Create, Add & Remove)	*RelationshipType* *RelationshipGroup*

5.3.2 Property display options

A collection containing options each of which identifies a property of one of the structures in this specification and determines whether, and if so in what form, it is to be displayed by a particular user interface service.

In its simplest form, the members of this collection may simply indicate that a particular property is to be displayed. However, in more sophisticated user interfaces this may also identify the relevant display control, position or format of display for that property.

This type of collection referred to in the specification as a place marker for some appropriate control on user-interface display characteristics. However, the content and range of flexibility is outside the scope of this specification.

d-Description	Property display options governing the display of the properties of a Description.
d-DescriptionList	Property display options governing the display of the properties of a Description in a list.
d-Concept	Property display options governing the display of the properties of a Concept.
d-Relationship	Property display options governing the display of the properties of a Relationship.
d-RelationshipList	Property display options governing the display of the properties of a Relationship in a list.
d-ElaborationList	Property display options governing the display of the properties of an *Elaboration* in a list.
d-HierarchyList	Property display options governing the display of the properties of a Concept in a hierarchical list.
d-ConceptExpression	Property display options governing the display of the properties of a post coordinated concept expression.
d-Subset	Property display options governing the display of the properties of a Subset.
d-SubsetList	Property display options governing the display of the properties of a Subset in a list.
d-CrossMapSet	Property display options governing the display of the properties of a Property display options governing the display of the properties of a Cross Map Set.
d-CrossMapSetList	Property display options governing the display of the properties of a Property display options governing the display of the properties of a Cross Map Set in list.
d-CrossMap	Property display options governing the display of the properties of a Cross Map.
d-CrossMapTarget	Property display options governing the display of the properties of a Cross Map Target.

5.3.3 Collections of enumerated values

The collection of enumerated values differ from other collections used in this specification in that they are not ordered and each possible value can only occur once in such a collection. Thus in any given collection each possible value either occurs or does not occur. Therefore these collections can be represented as a set of Boolean flags. This means that a collection of enumerated values can stored or exchanged as a single integer with the presence or absence of particular flags set or tested using binary AND, OR and NOT operations.

Collection name	Data Type of collection members	Structures or services from which this type of collection may be derived (and type of access)
c-DescEnums	DescEnum	Configuration (allow Add & Remove) Description (read only)
c-StatusEnums	StatusEnum	Configuration (allow Add & Remove)
c-CharEnums	CharEnum	Configuration (allow Add & Remove)
c-FeatureEnums	FeatureEnum	Server (read only)
c-SearchModeEnums	SearchModeEnum	Server (read only) Configuration (allow Add & Remove)
c-SearchRefineEnums	SearchRefineEnum	Server (read only) Configuration (allow Add & Remove)
c-SortOrderEnums	SortOrderEnum	Server (read only) Configuration (allow Add & Remove)
c-IdTypeEnums	IdTypeEnum	Parameter (allow Add & Remove)

6 Check-list of SNOMED CT Server API functions

The list of services in this section is based on the more detailed descriptions of requirements in the Technical Implementation Guide. It was originally published in this form in the Technical White Paper on " Review of SNOMED CT API Requirements and the proposed HL7 CTS" presented to the SNOMED International Editorial Board October 2002 meeting. It is included here in its original form as a check list for the services and/or data definitions.

6.1 Introduction

Reference services either "return" a value or Component or "allow" some type of modification of other services.
User-Interface services allow "display" of values or Components, "allow" some user action to alter the display or "return" a selected value or Component.
User-Interface services are prefixed by "UI-service". All other services listed in this section are Reference Services. In general the User-Interface services assume the availability of the associated to Reference Services to populate displays.

6.2 Access server and terminology system properties

6.2.1 Access to information about the server

Return properties of current server. For example, name, version, type and extended capabilities.
Supported by: GetServerInfo [3.1.1]

6.2.2 Access to release information

Read the value of any specified property of the current SNOMED CT release information.
Supported by: GetTerminologyInfo [3.1.2]

6.2.3 Access to essential concept identifiers

Return the ConceptIds that represent concepts with structurally significant roles.

- *The root concept*
- *The subtype (IS A) relationship*
- *Top-level concepts with structural roles*
  - Attribute (Relationship Types)
  - Qualifier value
  - Special concept
- Any specified subtype of Special Concept.

Supported by: Access to essential concept identifiers [3.1.4]

6.2.4 Validation of SCTIDs

Return one or more of the following items of information for a specified string intended to represent an SCTID:

- Whether the check-digit is valid;
- The partition identifier (if check-digit is valid);
- The namespace identifier (if partition-identifier indicates that the SCTID is part of an Extension);
- Whether a Component with this identifier is available.

Supported by: GetSctIdInfo [3.1.5]

6.3 Core table access services

6.3.1 Access to concepts

Return a Concept any of the following:

- ConceptId
- SNOMEDID
- CTV3ID

Supported by: GetConcept [3.2.1]

6.3.2 Access to properties of a concept

Return the value of any specified distributed property of a selected Concept.

Supported by: Access to properties of a concept [3.2.3]

6.3.3 Access to descriptions

Return a Description or set of Descriptions by any of the following:

- DescriptionId
- ConceptId
- ConceptId and DescriptionType

Supported by: GetDescription [3.2.4] and GetDescriptions [3.2.5]

6.3.4 Access to properties of a description

Return the value of any specified distributed property of a selected Description.

Supported by: Access to properties of a description [3.2.6]

6.3.5 Access to relationships

Return a Relationship or set of Relationships by any of the following:

- ConceptId1
- ConceptId1, CharacteristicType, and Relationship Type
- ConceptId1, CharacteristicType, Relationship group and Relationship Type
- ConceptId2
- ConceptId2, CharacteristicType, and Relationship Type

Supported by: GetRelationship [3.2.7], GetRelationships [3.2.8] and GetInverseRelationship [3.2.9]

6.3.6 Access to properties of a relationship

Return the value of any specified distributed property of a selected Relationship.

Supported by: Access to properties of a relationship [3.2.10]

6.4 Subtype services

6.4.1 Traversing subtype relationships

Return a set of Concepts that are related to a specified Concept as:

- Subtype children
- Subtype descendents
- Supertype parents
- Supertype ancestors

Supported by: GetChildren [3.3.1], GetDescendants [3.3.3], GetParents [3.3.2] and GetAncestors [3.3.4]

6.4.2 Identifying top-level supertypes

Return the top-level concept that is the supertype ancestor for any specified Concept.

Supported by: GetTopLevel [3.3.5]

6.4.3 Testing subtype descendants

Return an indication of whether any specified Concept is a descendant subtype of another specified Concept.

Supported by: IsDescendant [3.3.6]

6.4.4 Testing for special concept types

Return an indication of whether a specified Concept is a Navigation Concept.

Supported by: GetTopLevel [3.3.5]

6.5 Text search services

6.5.1 General features of text searching

Return a set of Descriptions that contain a specified word, phrase, set of words or text pattern.

Supported by: GetMatchingDescriptions [3.4.1]

6.5.2 Selection of text search modes

Allow selection of alternative search modes that are more or less sensitive to word order, case other variants in the precision of matching.

Supported by: GetMatchingDescriptions [3.4.1]

6.5.3 Filtering of text searches

Allow selective filtering or sorting of search result by any combination of the following:

- *DescriptionType* (in a specified Language Subset)
- *DescriptionStatus*
- Subset membership of Description or associated Concept
- *ConceptStatus* of associated Concept
- Supertype ancestry of associated Concept.

Supported by: GetMatchingDescriptions [3.4.1]

6.5.4 Rationalization of searches

Allow rationalization of searches which return multiple Descriptions that apply to the same Concepts or to closely related Concepts.

Supported by: GetMatchingDescriptions [3.4.1]

6.5.5 Displaying search results

UI-service: Display list of search results from a text search.

Supported by: ShowMatchingDescriptions [4.2.1]

6.5.6 Displaying selected descriptions

UI-service: Allow selection of Description from the list of search results to provide more detailed display of the Description or associated Concept.

Supported by: ShowDescription [4.1.1]

6.5.7 Returning search selections

UI-service: Return a Description selected in the search result list.

Supported by: GetSelectedItem [4.2.6]

6.6 Hierarchical navigation services

6.6.1 Following hierarchical relationships

Return a set of Concepts that are related to a specified Concept by hierarchical Relationships including:

- "Is a" (subtype children and supertype parents)
- "Part-of"

Supported by: GetRelationships [3.2.8]

6.6.2 Following navigation subsets

Return set of Concepts linked to a Concept by a specified Navigation Subset.

Allow selective filtering or sorting of set of hierarchical or navigationally linked Concepts by any combination of the following:

- Subset membership.
- *ConceptStatus*

Supported by: GetChildren [3.3.1] and GetParents [3.3.2]

6.6.3 Displaying relationship based hierarchies

UI-service: Display a hierarchical list of Concepts linked by "is a" or other hierarchical Relationships.

Supported by: ShowHierarchy [4.2.2]

6.6.4 Displaying navigational hierarchies

UI-service: Display a hierarchical list of Concepts following Navigation Subset links.

Supported by: ShowHierarchy [4.2.2]

6.6.5 Expanding and collapsing hierarchical displays

UI-service: Allow selection of an item in a hierarchical list to expand or collapse the display of child or parents defined by the chosen Relationship Type or Navigation Subset.
Supported by: ShowHierarchy [4.2.2]

6.6.6 Displaying selected concepts

UI-service: Allow selection of a Concept from a hierarchical list to provide more detailed display of the Concept and/or its associated Descriptions.
Supported by: ShowConcept [4.1.2] and ShowConceptDetail [4.2.3]

6.6.7 Returning a selected concept

UI-service: Return a Concept selected in the hierarchical list.
Supported by: GetSelectedItem [4.2.6]

6.7 Post-coordination support services

6.7.1 Access to possible qualifiers or refinements

Return sets of qualifiers and refinable defining characteristics for a specified Concept.
Supported by: GetPotentialElaborations [3.5.1]

6.7.2 Displaying qualifiers or refinements

UI-service: Display lists of qualifiers and refinable defining characteristics for specified Concept.
Supported by: ShowPotentialElaborations [4.2.4]

6.7.3 Expanding and collapsing lists of possible refinements

UI-service: Allow selection of a refinable item in a list of qualifiers or refinable defining characteristics to expand or collapse the display of its subtype children.
Supported by: ShowPotentialElaborations [4.2.4]

6.7.4 Returning selected qualifiers or refinements

UI-service: Return the attribute (Relationship Type) and value (*ConceptId2*) Concepts for a selected item in a list of qualifiers or refinable defining characteristics.
Supported by: GetSelectedItem [4.2.6]

6.7.5 Deriving the canonical forms for a concept

Return the canonical form for a specified Concept based either on the distributed Canonical Table or an appropriate computation of the defining characteristics.
Supported by: GetConceptCanonical [3.5.2]

6.7.6 Deriving the canonical form for a post-coordinated expression

Return the canonical form based on a post-coordinated represented involving two or more SNOMED CT Concepts.
Supported by: GetExpressionCanonical [3.5.3]

6.7.7 Displaying canonical forms

UI-service: Display a canonical form for a pre- or post-coordinated Concept.
Supported by: ShowConceptExpression [4.2.5]

6.8 Component display services

6.8.1 Displaying a details of a concept

UI-service: Display the attributes, associated Descriptions and Relationships of a specified Concept.
Supported by: ShowConceptDetail [4.2.3]

6.8.2 Displaying details of a description

UI-service: Display the attributes and the associated Concept of a specified Description.
Supported by: ShowDescription [4.1.1] and ShowConcept [4.1.2]

6.9 Data retrieval support services

6.9.1 Creating query statements for SNOMED CT data

Return SQL predicate statements (or fragments of such statements) including lists of ConceptIds in a form suitable for use in a query selective retrieving all instances of a specified Concept or any subsumed or equivalent concept representation.
Supported by: GetDescentSQL [3.6.1]

6.9.2 Creating query statements for legacy data

Return SQL predicate statements (or fragments of such statements) including lists of legacy codes (e.g. SNOMEDIDs or CTV3IDs) in a form suitable for use in a query selective retrieving all instances of a specified Concept or any subsumed or equivalent concept representation.
Supported by: GetDescentSQL [3.6.1]

6.10 Subsets configuration services

6.10.1 Locate the available Subsets

Return a Subset or set of Subsets selected according by criteria involving one or more properties of the Subsets Table.
Supported by: FindSubsets [3.7.2]

6.10.2 Access properties of a Subset

Return the value of any specified distributed property of a selected Subset.
Supported by: GetSubset [3.7.1]

6.10.3 Enable subset configuration and selection

Allow selection of Subsets in any or all of the following ways

- As a configuration activity applicable to an installation or session
- As a user selectable preference
- As a dynamically changeable filter determined manually or automatically by the record context.

Supported by: Configuration structure [5.2.2.4]

6.11 Cross mapping services

6.11.1 Locate the available Cross Map Sets

Return a Cross Map Set or set of Cross Map Sets selected according by criteria involving one or more properties of the Cross Map Sets Table.
Supported by: GetCrossMapSets [3.8.2]

6.11.2 Access properties of a Cross Map Set

Return the value of any specified distributed property of a selected Cross Map Set.
Supported by: GetCrossMapSet [3.8.1]

6.11.3 Locate the Cross Maps for a concept

Return the set of Cross Maps for a specified Concept in a specified Cross Map Set.
Supported by: GetCrossMaps [3.8.4]

6.11.4 Access properties of a Cross Map

Return the value of any specified distributed property of a selected Cross Map.
Supported by: Access properties of a Cross Map [3.8.5]

6.11.5 Locate the Cross Map Target referred to be a particular Cross Map

Return the Cross Map Target referred to by the TargetId of a Cross Map.
Supported by: GetCrossMapTarget [3.8.6]

6.11.6 Access properties of the a Cross Map Target

Return the value of any specified distributed property of a selected Cross Map Target.
Supported by: Access properties of the a Cross Map Target [3.8.7]

6.11.7 Display Cross Maps for a concept

UI-server: Display the Cross Maps for a selected Concept in a specified Cross Map Set.
Supported by: ShowConceptCrossMaps [4.4.3]

6.11.8 Display details of a selected Cross Map and associated Cross Map Target

UI-server: Allow selection of an item from a displayed list of Cross Maps and display the details of the Cross Map and its associated Cross Map Target.
Supported by: ShowCrossMap [4.4.4] and ShowCrossMapTarget [4.4.5]

6.11.9 Return a selected Cross Map and associated Cross Map Target

UI-server: Return a Cross Map selected from a displayed list of Cross Maps and its associated Cross Map Target.
Supported by: GetSelectedItem [4.2.6]

Index

Glossary

This glossary is intended to be an informative source of reference relating to various terms used in this and other documents connected with SNOMED Clinical Terms.

Active Concept	A Concept that is intended for active use. This is determined by the ConceptStatus. Concepts with status value “Current” (0) or “Limited” (6) are always regarded as active. Concepts with status “Pending Move” (11) are regarded as active if the Concept is not yet accessible in the new target Namespace. (see also Inactive Concept)
Active Description	A Description that is intended for active use. This is determined by a combination of the DescriptionStatus, the DescriptionType in the Language or Dialect in use and the ConceptStatus of the associated Concept. Descriptions are active when the following conditions apply Associated with an Active Concept, AND DescriptionStatus “Current” (0), “Limited” (6) or “Pending Move” (8) AND DescriptionType not unspecified (0) in a chosen Language/Dialect (see also Inactive Description)
Base Subset	A Subset used as a starting point the Subset Definition of another Subset.
Canonical form	A table that contains the Canonical form expressions for all SNOMED CT Concepts. This table contains some (but not all) of the attributes present in the Relationships Table. It only contains the Defining characteristics required to distinguish a Concept from its most proximate Primitive supertype Concepts. The set of “is a” subtype Relationships excludes Relationships to Fully defined (non-primitive) Concepts and includes instead appropriate additional Relationships to the most proximate supertypes.
Canonical Table	A table that contains the Canonical form expressions for all SNOMED CT Concepts. This table contains some (but not all) of the attributes present in the Relationships Table. It only contains the Defining characteristics required to distinguish a Concept from its most proximate Primitive supertype Concepts. The set of “is a” subtype Relationships excludes Relationships to Fully defined (non-primitive) Concepts and includes instead appropriate additional Relationships to the most proximate supertypes.
ChangeType	A field in the Component History Table that indicates the nature of a change to a Component made in a specified release of SNOMED CT.
CharacteristicType	A field in the Relationships Table that indicates whether a Relationship specifies a Defining characteristic, a qualifying characteristic or a context-specific characteristic.

Check-digit	A digit used to check the validity of an SCTID. The check-digit is the final (right most) digit of the SCTID and it is calculated using the algorithm described in Appendix F.
Child/Children	See Subtype.
Clinical Terms Version 3	See CTV3.
Component	An identifiable item in the main body of SNOMED CT, or in an authorized Extension. Each Component is a uniquely identifiable instance of one of the following: ❖ Concept ❖ Description ❖ Relationship ❖ Subset ❖ Subset Member ❖ Cross Map Set ❖ Cross Map Target ❖ History Component
Component History	A record of an addition or change in the status of a SNOMED CT Component in a particular Release Version. Each item of Component History is represented by a row in the Component History Table.
Component History Table	A data table consisting of rows each of which represents an item of Component History. The Component History Table is part of the History Mechanism. See specification of Component History Table [6.2.2].
ComponentId	A general term used to refer to the primary identifier of any SNOMED CT Component. ComponentIds include ConceptIds, DescriptionIds, RelationshipIds, SubsetIds, CrossMapSetIds and TargetIds. All ComponentIds follow the form of the SCTID specification.
Concept	A clinical idea to which a unique ConceptId has been assigned. Each Concept is represented by a row in the Concepts Table.
Concept equivalence	Concept equivalence occurs when a post-coordinated expression has the same meaning as a pre-coordinated Concept or another post-coordinated expression.
ConceptId	A SNOMED Clinical Terms Identifier that uniquely identifies a Concept.
ConceptId1	A field in the Relationships Table and Canonical Table that refers to the first of two related Concepts. The first Concept is defined or qualified by a Relationship to the second Concept.
ConceptId2	A field in the Relationships Table and Canonical Table that refers to the second of two related Concepts. The second Concept defines or qualifies the first Concept.

Concepts Table	A data table consisting of rows, each of which represents a Concept. See also the specification of the Concepts Table.
ConceptStatus	A field in the Concepts Table that specifies whether a Concept is in current use. Values include “current”, “duplicate”, “erroneous” “ambiguous” and “limited”.
Context Concept Subset	A Subset used to specify the Concepts that form part of a context-domain.
Context Description Subset	A Subset used to specify the Descriptions that form part of a context-domain.
Context Domain	A context-domain is a set of values that are, or may be, used in an identifiable logical setting in an application, protocol, query or communication specification. A context-domain may be very broad (e.g. procedures or diagnoses) or very narrow (e.g. procedures performed by a specialty or possible values for a field in specific message).
ContextId	A field in the Subsets Table that specifies the context within which a Context Concept Subset or Context Description Subset is valid.
Context-specific characteristic	A Relationship to a target Concept that provides information about the source Concept that is true at a particular time or within a particular country or organization. Contrast with Defining characteristic and Qualifying characteristic. Referred to in CTV3 as a ‘Fact’.
Core Tables	See SNOMED CT Core Tables.
Cross Map	A reference from a Concept to a Cross Map Target. A Cross Map is part of a Cross Map Set A Concept may have a single Cross Map or a set of alternative Cross Maps. Each Cross Map is represented as a row in the Cross Maps Table.
Cross Map Set	A set of Cross Maps that together provide a valid way of mapping some or all SNOMED CT Concepts to a specified Target Scheme. Alternative Cross Map Sets may exist for the same Target Scheme, if business rules or guidelines alter the appropriateness of particular mappings to that scheme. Each Cross Map Set is represented as a row in the Cross Map Sets Table.
Cross Map Sets Table	A data table consisting of rows each of which represents a Cross Map Set. See the specification of the Cross Map Sets Table.

Cross Map Target	A code or set of codes in a Target Scheme that together represent an appropriate mapping from a clinical statement expressed using SNOMED CT. Some Cross Map Targets may be derived from two or more associated statements and in these cases the combination can be expressed as a set of associated rules. Each Cross Map Target is represented as a row in the Cross Map Targets Table.
Cross Map Targets Table	A data table consisting of rows each of which represents a Cross Map Set. See the specification of the Cross Map Targets Table.
Cross Maps Table	A data table consisting of rows each of which represents a Cross Map. See the specification of the Cross Maps Table.
CTV3 Clinical Terms Version 3	One of the source terminologies, along with SNOMED RT, used to develop SNOMED CT. CTV3 is UK Crown Copyright, distributed by the United Kingdom National Health Service Information Authority, and integrated into SNOMED CT. Also known as “Version 3 of the Read Codes.” See also Read Code.
CTV3ID	A field in the Concepts Table that contains the Clinical Terms Version 3 identifier (Read Code) for that Concept.
Data migration	Steps taken to enable legacy data to be accessible as part of a system that uses SNOMED CT. Options for Data migration include actual conversion of the data or provision of methods for accessing the data in its original form.
Defining characteristic	A Relationship to a target Concept that is always true from any case of the source Concept. Example: “‘Topography (site)’ = ‘Liver’” is a Defining characteristic of the Concept ‘Liver biopsy’. Contrast with qualifying characteristic and context-specific characteristic. Referred to in CTV3 as an ‘Atom’.
Descendants	All subtypes of a concept, including subtypes of subtypes. For example, if a concept has four children, then descendants = those children plus all the concepts that are descended from those four children. See also Subtype.
Description	A row in the Descriptions Table. Each Description is assigned a unique DescriptionId and connects a Term and a Concept.
DescriptionId	A SNOMED CT Identifier that uniquely identifies a Description.
Descriptions Table	A data table consisting of rows, each of which represents a Description. See the specification of the Descriptions Table [B.2].
DescriptionStatus	A field in the Descriptions Table that specifies whether a Description is in current use. Values include “current”, “duplicate”, “erroneous” “inappropriate” and “limited”.

DescriptionType	A field in the Descriptions Table that specifies whether a Description is a Fully Specified Name, Preferred Term, or Synonym. The DescriptionType is language dependent and may be changed by applying a Language Subset. It may be “undefined” in the released Descriptions Table in which case the Description is not used unless an appropriate Language Subset is applied.
Dialect	A language modified by the vocabulary and grammatical conventions applied to the language of a particular geographical or cultural environment.
Dualkey	A key used to facilitate textual searches of SNOMED CT that consists of the first three letters of a pair of words in a Description. All possible pairs of words in each Description may be paired irrespective of their relative position in the Description. Dualkeys are represented as a row in the Dualkeys Table.
Dualkey Table	A table in which each row represents a Dualkey. See specifications of the DescDualKey Table [B.7.3.1] and ConcDualKey Table [B.7.3.2]
Duplicate Term	A Term that occurs in several Active Descriptions. Duplicate Terms are valid in SNOMED CT since the intention is to provide natural terms used by clinicians rather than to apply formalized phraseology. The formalized form is provided by the Fully Specified Name and these are not permitted to be duplicated. Although Duplicate Terms can be identified by string matching, a Duplicate Terms Subset is specified to indicate the presence and likely priority of duplicates when undertaking a search.
Duplicate Terms Subset	A Subset Type that identifies Duplicate Terms and allows a priority to be specified between these for use in searches.
Equivalence	See Word Equivalents, Phrase equivalence and Concept equivalence.
Excluded Word	A word that in a given language is so frequently used, or has so poor a discriminating power, that it is suggested for exclusion from the indices used to support textual searches of SNOMED CT. Excluded Words are represented as a row in the Excluded Words Table
Excluded Words Table	A data table in which each row represents an Excluded Word. See specification of the Excluded Words Table [B.7.1].

Expression	A collection of references to one or more concepts used to express an instance of a clinical idea. An expression containing a single concept identifier is referred to as a *pre-coordinated expression*. An expression that contains two or more concept identifiers is a *post-coordinated expression.* The concept identifiers within a *post-coordinated expression* are related to one another in accordance rules expressed in the SNOMED CT Concept Model. These rules allow concepts to be: • combined to represent clinical ideas which are subtypes of all the referenced concepts, e.g. "tuberculosis" + "lung infection" • applied as refinements to specified attributes of a more general concept, e.g. "asthma" : "severity" = "severe" *Notes:* See "SNOMED CT Guide - Abstract Models and Representational Forms". This has been released as an External Draft in July 2005.
Extension	A data table or set of data tables that is created in accordance with the structures and authoring guidelines applicable to SNOMED CT but which may not be edited, maintained and distributed by the IHTSDO. SNOMED CT Components are identified using SCTIDs, which are structured to ensure that Extensions are recognizable and can be traced to an authorized originator.
Extra- Relationship	A Relationship between two Concepts distributed as part of an Extension. An Extra-Relationship may relate SCT Concepts and/or Extra-Concepts.
Extra-Concept	A Concept distributed as part of an Extension.
Extra-Description	A Description which is distributed as part of an Extension. An Extra-Description may apply a Term to an SCT Concept or to an Extra-Concept.
Extra-Subset	A Subset distributed as part of an Extension. The members of an Extra-Subset may include SNOMED CT-Concepts, Extra-Concepts, SNOMED CT- Descriptions and Extra-Descriptions.
Fully defined	A Concept is Fully defined if its Defining characteristics are sufficient to differentiate a concept relative to its immediate supertype(s). A Concept which is not Fully defined is Primitive. For example, if the Concept “Red car” is defined as [is a=*car*] + [color=*red*] this is Fully defined but the same definition applied to the Concept “Red sports car” is Primitive.
Fully Specified Name	A phrase that describes a Concept uniquely and in a manner that is intended to be unambiguous. This phrase is in the FullySpecifiedName field of the Concepts Table and is also in the Descriptions Table.

Historical Relationship	A Relationship that refers from an Inactive Concept to an Active Concept that duplicates, corrects, replaces or disambiguates it. Note that Historical Relationships are used in a way similar to References for Descriptions. However, as part of the Relationships Table they are more readily accessible for computation and retrieval of legacy data.
History Mechanism	A SNOMED CT distribution table that contains information about the history of changes to one of the core SNOMED CT tables. The History Mechanism is supported by two distribution tables: ❖ Component History Table References Table (Future Use)
Inactive Concept	A Concept that is not intended to be actively used. This is determined by the ConceptStatus. Concepts with status values other than “Current” (0), “Limited” (6) and “Pending Move” (11) are regarded as inactive. Concepts with status “Pending Move” (11) are regarded as inactive if the Concept is accessible in the new target Namespace as of the release date. Inactive Concepts remain in SNOMED CT to support legacy data recorded when these Concepts were in active use. (see also Active Concept)
Inactive Description	A Description that is not intended to be actively used. . This is determined by a combination of the DescriptionStatus, the DescriptionType in the Language or Dialect in use and the ConceptStatus of the associated Concept. Descriptions are inactive if one or more of the following apply Associated with an Inactive Concept, OR DescriptionStatus not “Current” (0), “Limited” (6) or “Pending Move” (8) OR DescriptionType unspecified (0) in a chosen Language/Dialect Inactive Descriptions remain in SNOMED CT to support legacy data recorded when these Descriptions were in active use. (see also Active Description)
InitialCapitalStatus	A field in the Descriptions Table that specifies whether the capitalization of the first character is significant. If the value of this field is “1” then the first character should remain either in upper or lower case as released. Otherwise the case of the first character may be changed to suit its context in a sentence.
International Release	The required international components of the SNOMED CT terminology, along with related works and resources, maintained and distributed by the IHTSDO.
is a ISA “is a” relationship	The RelationshipType that defines a supertype-subtype Relationship between two Concepts. Usually expressed as subtype “is a” supertype - - For Example, Blister with infection IS A Infection of skin.

IsPrimitive	A field in the Concepts Table that indicates whether a Concept is Primitive or Fully defined.
Keyword	A field containing a potential search text in one of the WordKey Tables or a word excluded for key generation in the Excluded Words Table.
Kind-of-Value	The nature of a value that may be associated with a Concept. For example, the concept "systolic blood pressure reading" can label a numeric value. The Kind-of-Value that it labels is a pressure.
Language	A vocabulary and grammatical form that has been allocated an ISO639-1 language code. See also Dialect.
Language Subset	A Subset that specifies the various Terms according to a language or dialect and the DescriptionType for each Term.
LanguageCode	A field that indicates the Language and, optionally, Dialect applicable to a row in the Subsets Table, Descriptions Table or to an Excluded Words Table.
LinkedId	A field in the Subsets Table
MapAdvice	A field in the Cross Maps Table, may contain human-readable advice on mapping.
MapConceptId	A field in the Cross Maps Table containing the identifier of the Concept that is the subject of the map.
MapOption	A field in the Cross Maps Table, which specifies the order in which Cross Maps are tested for automated processing of cross mapping rules.
Mapping Mechanism	A mechanism for mapping to other terminologies and classifications. The Mapping Mechanism is supported by three distribution tables: ❖ Cross Map Sets Table ❖ Cross Maps Table ❖ Cross Map Targets Table
MapPriority	A field in the Cross Maps Table that specifies which Cross Maps are most likely to apply to the associated Concept. The value 0 indicates a default map.
MapRule	A field in the Cross Maps Table that may contain a computer processable representation of a rule that determines when this map should be used.
MapSetId	A SNOMED CT Identifier that uniquely identifies a Cross Map Set.
MapSetName	A field in the Cross Map Sets Table that names that Cross Map Set.
MapSetRealmId	A field in the Cross Map Sets Table that indicates the Realm in which a set of Cross Maps is applicable.

MapSetRuleType	A field in the Cross Map Sets Table that indicates whether any computer processable rules are present in the associated Cross Maps or Cross Map Targets and, if so, what form of expression is used to represent these rules.
MapSetSchemeId	A field in the Cross Map Sets Table, which identifies the classification or coding-scheme that is the target of a Cross Map Set.
MapSetSchemeName	A field in the Cross Map Sets Table that contains the plain text name of the classification or coding-scheme that is the target of a Cross Map Set.
MapSetSchemeVersion	A field in the Cross Map Sets Table that identifies the version of the classification or coding-scheme that is the target of a Cross Map Set.
MapSetSeparator	A field in the Cross Map Sets Table that contains a character that acts as a separator between target codes in the Cross Map Targets Table.
MapSetType	A field in the Cross Map Sets Table that indicates whether the Cross Maps associated with this Cross Map Set are all simple one to one maps or include, one to many and/or choices of alternative maps.
MapTargetId	A field in the Cross Maps Table, which refers to the TargetId of a row in the Cross Map Targets Table that contains a target scheme mapping for a specified Concept.
MemberId	A field in the Subset Members Table, which refers to the ComponentId of the Concept, Description or other Component that is a member of a specified Subset.
MemberStatus	A field in the Subset Members Table, which indicates the inclusion, exclusion, priority or order of an identified Subset Member in a specified Subset.
Migration	See Operational migration, Data migration and Predicate migration.
Moved Elsewhere	A Status value applicable to a Component that has been moved to another Namespace. Concepts or Descriptions may be moved from an Extension to the SNOMED CT core, from the core to an Extension or between one Extension and another. Moves occur if responsibility for supporting the Concepts changes to another organization.
Namespace Namespace-identifier	A block of identifiers allocated for use by an organization creating Extensions to SNOMED CT. The Namespace-identifier is part of the SCTID and its structure is described in Appendix C.

Namespace Concept	A Concept that exists to represent a SNOMED CT Namespace-identifier. All Namespace Concepts are direct subtypes of the Concept “Namespace Concept” which is a subtype of the Top-Level Concept “Special Concept”. Namespace Concepts are used as the target of Historical Relationships and References when a Component is moved from one Namespace to another.
Navigation	The process of locating a Concept by traversing Relationships or Navigational links. For example, moving from a supertype Concept to more refined Concepts, from a specific Concept to a more general Concept or from a Concept to its Defining characteristics. Navigation Links allow navigation to follow intuitive routes through SNOMED CT even where there are no direct supertype or subtype Relationships.
Navigation Concept	A Concept that exists only to support Navigation. A Navigation Concept is not suitable for recording or aggregating information. All Navigation Concepts: ❖ Are direct subtypes of the concept “Navigational Concept” ❖ Have not other supertype or subtype Relationships ❖ Are linked to other Concepts only by Navigational Links
Navigation Link	An association between two Concepts that supports Navigation between Concepts. Navigation Links generate a hierarchy which has three distinct differences from the subtype hierarchy defined by “IS A” Relationships this hierarchy: Does not effect the semantic definitions of Concepts; Specifies a display order Concepts within a set of Concepts linked to a common parent. Navigation Links are distributed as a Navigation Subset. Alternative Navigation Subsets may be specified and applied to vary the navigational hierarchy to meet the needs of particular groups of users.
Navigation Subset	A Subset that specifies sets of Navigation Links between Concepts.
NHS Information Authority NHS IA	Located in the United Kingdom, the National Health Service Information Authority partnered with the College of American Pathologists in the development of SNOMED CT.
Operational migration	Steps taken to enable an organization that either used a previous coding scheme (or no clinical coding scheme) to make use of SNOMED CT.
Partition-identifier	A pair of digits that indicate whether an SCTID identifies a Concept, Description, Relationship, Subset, History, or Extension component. The partition-identifier consists of the second and third digits from the right of the SCTID.

Pending Move	A Status value applicable to a Component that is thought to belong in a different Namespace but which is maintained with its current SCTID while awaiting addition to the new Namespace. A new Concept and associated Descriptions may be added with this Status where a missing SNOMED CT Concept is urgently required to support the needs of a particular Extension. Existing Concepts can also use this status when it is recognized that they should be moved to another namespace (organization) or to the core namespace. See also Moved Elsewhere.
Phrase equivalence	Two words or phrases with a similar meaning. For example, “renal calculus” and “kidney stone”. See Word Equivalents.
Post-coordination	Representation of a clinical idea using a combination of two or more concept identifiers. A combination of concept identifiers used to represent a single clinical idea is referred to as a post-coordinated expression (see expression). Many clinical ideas can also be represented using a single SNOMED CT concept identifier (see pre-coordination). Some clinical ideas may be represented in several different ways. SNOMED CT technical specifications include guidance of logical transformations that reduce equivalent *expressions* to a common *canonical form*. Example: SNOMED CT includes the following concepts: *Fracture of bone (conceptId= 125605004)* *Finding site (conceptId= 363698007)* *Bone structure of femur (conceptId= 181255000)* SNOMED CT also includes a pre-coordinated concept for this procedure *Fracture of femur (conceptId= 71620000)* It is possible to represent "fracture of femur" in different ways: *71620000* (pre-coordinated expression) *125605004* : *363698007* = *181255000* (post-coordinated expression). *Note:* In an HL7 representation a SNOMED CT expression is represented in a single HL7 attribute using the HL7 CD (Concept Descriptor) data type.

Pre-coordination	Representation of a clinical idea using a single concept identifier. A single concept identifier used to represent a specific meaning is referred to as a pre-coordinated expression (see expression). SNOMED CT also allows the use of post-coordinated expressions (see *post-coordination*) to represent a meaning using a combination of two or more concept identifiers. However, including commonly used concepts in a pre-coordinated form makes the terminology easier to use. For examples see post-coordination.
Predicate migration	Steps taken to enable pre-existing data retrieval predicates (including queries, standard reports and decision support protocols) to be converted or utilized in a system using SNOMED CT.
Preferred Term	The Term that is deemed to be the most clinically appropriate way of expressing a Concept in a clinical record. Preferred Term is one of the three types of terms that can be indicated by the DescriptionType field.
Primitive	A Concept is Primitive if its Defining characteristics are insufficient to define it relative to its immediate supertype(s). For example, if the Concept “Red sports car” is defined as [is a=*car*] + [color=*red*] this is Primitive but the same definition applied to the Concept “Red car” is Fully defined.
Qualifying characteristic	A Relationship to a target Concept that specifies a possible qualification of the source Concept. Example: “‘Revision status’ = ‘Conversion from other type of arthroplasty’” is a possible qualifying characteristic of ‘Hip replacement’ Contrast with defining characteristic and context-specific characteristic. Referred to in CTV3 as a ‘Qualifier’.
Read Code	A five-character code allocated to a concept of term in CTV3. Note that codes allocated in Read Codes Version 2 and the Read Codes 4-Byte Set are also included in CTV3. In the case of 4-byte codes the original code is prefixed by a full stop (‘.’).
Read Codes 4-Byte Set	The first version of the clinical coding scheme developed by Dr James Read. The Read Codes 4-Byte Set is UK Crown Copyright distributed by NHS IA, and integrated into SNOMED CT.
Read Codes Version 2	The second version of the clinical coding scheme developed by Dr. James Read. Read Codes Version 2 is UK Crown Copyright distributed by NHS IA, and integrated into SNOMED CT.
Realm	A sphere of authority, expertise or preference that influences the range of Concepts and Descriptions required, or the frequency with which they are used. A Realm may be a nation, organization, professional discipline, specialty or individual user. See also Realm Concept Subset.

Realm Concept Subset	A Subset of Concepts applicable to a particular Realm.
Realm Description Subset	A Subset of Descriptions applicable to a particular Realm.
Realm Relationship Subset	A Subset of Relationships applicable to a particular Realm (for future use).
RealmId	A field in the Subsets Table that identifies the Realm within which the specified Subset is applicable.
Reason	A field in the Component History Table which provides a text description of the reason for a change in the Status of a Component.
Reference	An association between a non-current SNOMED CT Component and a current Component that duplicates it, replaces it or is related to it. Each Reference is represented by a row in the References Table.
ReferencedId	A field in the References Table that identifies the current Component which replaces it or is duplicated by a non-current Component.
References Table	A data table consisting of rows each of which represents a Reference. The References Table is part of the History Mechanism.
ReferenceType	A field in the References Table that indicates whether a specified non-current Component was replaced by, duplicated by, similar to or an alternative form of the referenced current Component.
Refinability	A field in the Relationships Table, which indicates whether it is permissible to refine the value of a Defining characteristic or qualifying characteristic to represent a more refined Concept.
Relationship	An association between two Concepts (each identified by a ConceptId). The nature of the association is indicated by a RelationshipType Each Relationship is represented by a row in the Relationships Table.
Relationship group	Relationships, for a concept that are logically associated with each other. The RelationshipGroup field in the Relationships Table is used to group these rows together for a concept. For example, where a particular type of prosthesis is inserted a joint, the Defining characteristics describing the prosthesis type would be in one group whereas those describing the location or laterality of the joint would be in another group.
Relationship Type Type of Relationship	The nature of a Relationship between two Concepts. Relationship Types are represented in SNOMED CT by Concepts. The RelationshipType field indicates the nature of the Relationship by referring to the appropriate ConceptId.
RelationshipId	A SNOMED CT Identifier that uniquely identifies a Relationship.
Relationships Table	A data table consisting of rows, each of which represents a Relationship.

Release Version	An identifiable set of SNOMED CT tables distributed on or after a particular date for use in SCT Enabled Applications. Each Release Version is referred to by the ISO format date of which this set of files was distributed (or was scheduled for distribution). Thus release version "20030131" refers to the version released on January 31, 2003.
ReleaseVersion	A field in the Component History Table which indicates the SNOMED CT release in which a Component was added or changed.
Retired Concept	Concepts that are no longer considered current are called “non current” or “inactive” rather than “retired.”
Root Concept	A single special Concept that represents the root of the entire content of SNOMED CT. All other Concepts are related to this Concept via a least one series of Relationships of the Relationship Type "ISA" (i.e. all other Concepts are regarded as subtypes of the Root Concept).
SCT Enabled Application	A software application designed to support the use of SNOMED CT.
SCT-Concept SCT-Description SCT-Relationship SCT-Subset SCT-History	A Concept, Description, Relationship, or Subset distributed by the IHTSDO as part of SNOMED CT. The prefix "*SCT*-" is used only where it is necessary to distinguish clearly between elements distributed by the IHTSDO and Extensions (denoted by the prefix "*Extra*-"). For further information refer to the glossary entries for the unprefixed names.
SNOMED®	The **S**ystematized **No**menclature of **Med**icine
SNOMED Clinical Terms SNOMED CT®	The clinical terminology maintained and distributed by the IHTSDO.
SNOMED CT Core Tables	Refers to the SNOMED CT Concept, Relationship and Description Tables.
SNOMED CT Derivative	Documentation, subsets, cross-mappings, extensions, and other files.
SNOMED CT Identifier SCTID	A unique identifier applied to each Concept, Description or Subset. The SCTID includes a check-digit and a partition-digit. See specification of SCTID in Appendix C.
SNOMED International	SNOMED International is the version of SNOMED® that was first released in 1993 and which, as version 3.5 released in 1998, was the immediate predecessor of SNOMED RT. SNOMED International was also the name of the organization (now SNOMED Terminology Solutions) within the College of American Pathologists that was responsible for SNOMED CT until the transfer of the terminology to the IHTSDO in 2007.

SNOMED Reference Terminology SNOMED RT	The latest version of SNOMED® prior to the collaborative effort to develop SNOMED Clinical Terms. One of the source terminologies, along with CTV3, used to develop SNOMED CT. SNOMED RT was sponsored by the College of American Pathologists (CAP).
SNOMEDID	A field in the Concepts Table that contains the SNOMED RT identifier for the Concept.
Status	The Status of a Component indicates whether it is in current use and, if not, provides a general indication of the reason that it is not recommended for current use. The Status of a Concept is referred to as ConceptStatus and the Status of a Description is referred to as DescriptionStatus.
Subset	Subsets represent groups of Components that share specified characteristics that affect the way they are displayed or otherwise accessible within a particular realm, specialty, application or context.
Subset Definition	A series of clauses that specifies the nature of a Subset and determines its membership. A Subset Definition is an alternative form of representation applicable to many Subsets. See also Subset Definition File.
Subset Definition Clause	A statement that refers directly or indirectly to one or more Concepts, Descriptions or Relationships and indicates whether the referenced item(s) should be removed from or added to a specified Subset. In the case of additions to a Subset the clause also specifies the MemberStatus to be assigned to those additions.
Subset Definition File	A file containing a series of clauses that define the nature and membership of a Subset. The Subset Definition File is an XML document that contains the Subset Definition for a single Subset. A Subset Definition File can be used to generate the appropriate Subsets Table row and Subset Members Table rows to represent a Subset. Therefore applications need not support this alternative form of Subset representation.
Subset Member	A Concept, Description, Relationship or another Subset that is part of a specified Subset.
Subset Members Table	A data table consisting of rows each of which refers to a single Subset Member.
Subset Type Type of Subset	An indication of the type of component that may be a member of a Subset.
Subset Version	A version number assigned to a particular release of a Subset.
SubsetId	A SNOMED CT Identifier that uniquely identifies a Subset.
SubsetName	A field in the Subsets Table that contains a human readable name for the Subset.

SubsetOriginalId	A field in the Subsets Table that identifies the first version of the Subset on which this Subset is based. For the first version of a Subset the SubsetOriginalId and SubsetId fields contain the same value. For each subsequent version the Subset Version is incremented and a new SubsetId is allocated but the SubsetOriginalId field retains the same value in all versions.
Subsets Table	A data table consisting of rows each of which represents a Subset.
Subtype	A specialization of a concept, sharing all the definitional attributes of the parent concept, with additional granularity. For example, bacterial infectious disease is a subtype of infectious disease. Bacterial septicemia, bacteremia, bacterial peritonitis, etc. are subtypes of bacterial infectious disease (and infectious disease as well). In short, the concepts in a hierarchy that are directly related to a parent concept via the “IS A” relationship. Distinguished from “Descendants” which explicitly includes subtypes of subtypes. Subtype is usually intended to refer to only the concepts that are immediately under the subject concept – that is, one level down in the hierarchy.
Synonym	A Term which is an acceptable alternative to the Preferred Term as a way of expressing a Concept.
Target Code	A code or other identifier within a Target Scheme.
Target Scheme	A terminology, coding scheme or classification to which some or all SNOMED CT Concepts are cross-mapped. Mappings from SNOMED CT to a Target Scheme are represented by one or more Cross Map Sets.
TargetAdvice	A field in the Cross Map Targets Table that may contain human readable advice on the circumstances in which this Cross Map Target is applicable.
TargetCodes	A field in the Cross Map Targets Table that contains one or more codes or identifiers in the target scheme or classification. If there is more than one Target Code they are separated by a separator specified in the associated row of the Cross Map Sets Table
TargetId	A SNOMED CT Identifier that uniquely identifies a Cross Map Target.
TargetRule	A field in the Cross Map Targets Table that may contain computer processable rules specifying the circumstances in which this Cross Map Target is applicable.
TargetSchemeId	A field in the Cross Map Targets Table that identifies the target coding scheme or classification to which this Cross Map Target applies.
Term	A text string that represents the Concept. The Term is part of the Description. There are multiple descriptions per Concept.

Terminology server	Software that provides access to SNOMED CT (and/or to other terminologies). A Terminology server typically supports searches and Navigation through Concepts. A server may provide a user interface (e.g. a browser or set of screen controls) or may provide low-level software services to support access to the terminology by other applications. See the SNOMED CT Technical Implementation Guide.
Top-Level Concept	A Concept that is directly related to the Root Concept by a single Relationship of the Relationship Type "ISA". All other Concepts are descended from one Top-Level Concept via at least one series of Relationships of the Relationship Type "ISA" (i.e. all other Concepts are subtypes of one Top-Level Concept).
Unicode	A standard character set, which represents most of the characters used in the world using a 16-bit encoding. Unicode can be encoded in using UTF-8 to more efficiently store the most common ASCII characters.
UTF-16	A standard method of directly encoding Unicode using two bytes for every character. See also UTF-8.
UTF-8	A standard method of encoding Unicode characters in a way optimized for the ASCII character set. UTF-8 is described in [Appendix E].
Word Equivalent	A word or abbreviation that is stated to be equivalent to one or more other words, phrases or abbreviations for the purposes of textual searches of SNOMED CT. Word Equivalents and Phrase equivalents are represented as rows in the Word Equivalents Table.
Word Equivalents Table	A data table in which each row represents a Word Equivalent.
WordBlockNumber	A field in the Word Equivalents Table, which links together several rows which have an identical or similar meaning.
WordKey Table	A data table relating each word used in SNOMED CT (other than Excluded Words) to the Descriptions.
WordRole	A field in the Word Equivalents Table, which specifies the usual usage of this word, abbreviation or phrase, or the usage in which it has a similar meaning to the text in one or more other rows of the table that share a common WordBlockNumber.
WordText	A field in the Word Equivalents Table, which contains a word, phrase, acronym or abbreviation that is considered to be similar in meaning to the text in one or more other rows of the table that share a common WordBlockNumber.
WordType	A field in the Word Equivalents Table, which specifies whether this row contains a word, phrase, acronym or abbreviation